Implementación del Soporte Básico de Comunicaciones para μCronOS

Víctor D. Castillo Díaz
Lab. de Sistemas de Información

VictorCastilloEn@yahoo.com.mx
www.geocities.com/ViDaSoftwareInc/

Rolando Menchaca Méndez
Jefe del Lab. de Sistemas de Información

rmen@cic.ipn.mx

Centro de Investigación en Computación-IPN
Unidad Profesional Adolfo Lopez Mateos, C.P. 07730, México D.F.

Resumen: En este escrito se presentan los detalles de implementación del Soporte Básico de Comunicaciones (SBC) para el núcleo β CronOS, basado en el controlador PCI Ethernet "*Realtek RTL8029AS*" y compatibles con el estándar NE2000 (Holtek HT80229, HT80232, Via VT86C926, Winbond 89c940). Este soporte nos brinda la oportunidad de realizar comunicaciones de datos entre diversos equipos de forma eficiente, al no haber ningún protocolo se provee un ancho de banda casi igual al ofrecido por la tarjeta, con todas las ventajas que esto implica y la posibilidad de definir los protocolos en las capas de software superiores acorde a las necesidades.

Palabras clave: Soporte Básico de Comunicaciones, NIC, RTL8029AS, Controlador Ethernet, NE2000, Escritura de Driver.

1 Introducción

El ancho de banda de las redes a sido mejorado con una frecuencia impresionante, incrementando por lo menos 10 veces cada 5 años. Sin embargo existe un gran hueco entre el ancho de banda ofrecido por el dispositivo y lo que el usuario puede obtener realmente.

$$\textit{Ancho de banda efectivo} = \frac{\textit{Longitud del mensaje}}{\textit{Tiempo de comunicación}}$$

Muchos factores afectan el desempeño del subsistema de comunicaciones. Los más importantes se listan a continuación.

- El hardware de comunicaciones, incluyendo la cantidad de memoria y arquitectura de E/S del equipo, la interfaz de red y la red.
- El software de comunicaciones, incluyendo la estructura y algoritmos de los protocolos.

- El ambiente del usuario, incluyendo si este soporta multiusuarios, multiprocesamiento, multiprogramación, etc.
- Los servicios de comunicación, incluyendo envió de mensajes, control de flujo, control de errores, protección, etc.

Se han propuesto diferentes arquitecturas para la NIC (Tarjeta de Interfaz de Comunicaciones) y han surgido las siguientes características comunes.

- La NIC debe tener funcionalidad DMA (Acceso Directo a Memoria). La E/S programada produce sobrecarga, mayor latencia y bajo ancho de banda.
- La NIC debe incluir un procesador, el cual es necesario para la inicialización DMA, empaque y desempaque de datos, verificaciones de seguridad, etc.
- Todas las operaciones DMA deben ser inicializadas por el procesador del NIC, no el procesador del sistema. Esto permite mover datos sin intervención del procesador central; el procesador central esta *desacoplado* de la red.
- La NIC debe tener una memoria local para almacenar código del NIC y guardar mensajes, etc. Esta memoria puede ser accedida a través de E/S programada. La NIC debe tener también colas FIFO para guardar paquetes.

Sin embargo, la sobrecarga del software domina el tiempo de comunicación en los sistemas actuales. De esto que el tiempo de comunicaciones no pueda reducirse aunque se mejore la tecnología de la red y los NIC.

La sobrecarga por el software se produce principalmente por 3 causas:

- Se necesita atravesar varias capas de protocolos. Esto puede minimizarse, simplificando los protocolos.
- La comunicación suele necesitar múltiples copias de datos, las cuales tienen un costo y seria deseable requerir el mínimo número de ellas.
- Se necesita saltar la protección de los espacios de memoria varias veces, por lo cual se están usando protocolos del nivel de usuario, en donde no existen tantas restricciones.

2 Estado del Arte

La conocida interfaz socket es ampliamente utilizada por sistemas de aplicación y distribuidos como sistemas de archivo en red (NFS), llamadas a procedimientos remotos (RPC), ftp, telnet, http, etc.

Dichas aplicaciones pueden basarse en sockets convencionales, enviando mensajes que se dirigen hacia las capas inferiores, pasando por la pila de protocolos TCP/IP, hasta llegar a los controladores de dispositivo, los cuales acceden al hardware de comunicaciones. Al recibir un mensaje, se recorre el mismo camino pero en orden inverso. Durante estos recorridos ganan funcionalidad, a la vez que generan sobrecarga.

Cabe señalar que actualmente los sockets pueden también ser implementados sobre una *Capa de Comunicaciones Básica* (BCL) de bajo nivel, evitando pasar por la pila de protocolos TCP/IP. Con lo cual ciertamente pierden funcionalidad pero prácticamente no existe sobrecarga en las comunicaciones, pudiendo entonces aprovechar un ancho de banda mayor para necesidades especificas de las aplicaciones.
Adicionalmente se están empleando *protocolos de cero copias*, que permita el envió directo de información sin necesidad de copias, eliminando la sobrecarga impuesta por el uso de estas.

La estructura común de sockets y BCL (Capa Básica de Comunicaciones) se muestra en la figura 1.

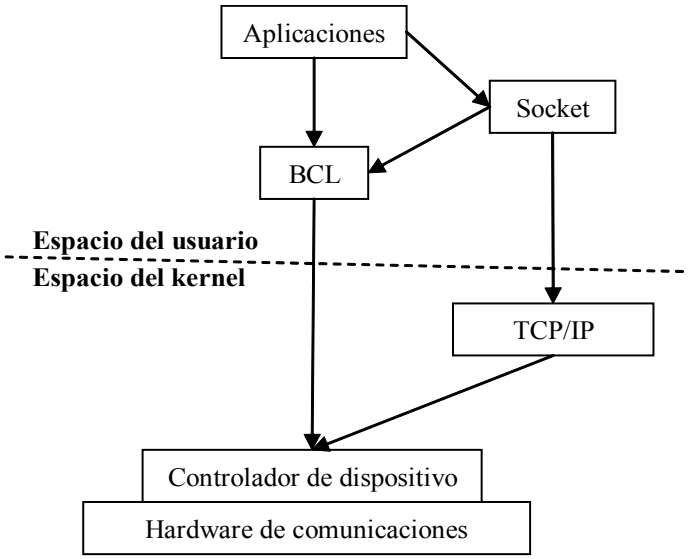

Figura 1 *Estructuras de comunicaciones*

El principal objetivo de BCL es proporcionar a los usuarios un ancho de banda efectivo muy cercano al proporcionado por el hardware.

3 El Soporte Básico de Comunicaciones

Ahora se describirán los detalles de implementación del soporte básico de comunicaciones basado en la NIC "Realtek RTL8029AS" y su uso eficiente, por medio de una ligera capa BCL que realice accesos más directos al dispositivo a favor de los procesos del sistema.

3.1 Procedimiento de Inicialización

La NIC debe ser inicializada antes de la transmisión o recepción de paquetes provenientes de la red.

Secuencia de inicialización necesaria

1) Programar Registro de Comandos, seleccionando página 0 (CR = 21h)
2) Inicializar Registro de Configuración de Datos (DCR)
3) Limpiar Registros Contadores de Byte Remoto (RBCR0, RBCR1)
4) Inicializar Registro de Configuración de la Recepción (RCR)
5) Establecer la NIC en modo LoopBack 1 o 2 (TCR = 02h o 04h)
6) Inicializar Buffer de Recepción Circular: Apuntador Limite (BNDRY), Pagina Inicial (PSTART) y Página Final (PSTOP)
7) Limpiar Registro de Estado de Interrupción (ISR), escribiendo 0ffh en el.
8) Inicializar Registro de Mascara de Interrupción (IMR)
9) Programar Registro de Comandos, seleccionando la página 1 (CR = 61h)
    i) Inicializar Registros de Dirección Física (PAR0-PAR5)
    ii) Inicializar Registros de Dirección Multicast (MAR0-MAR7)
    iii) Inicializar Registro Apuntador Actual (CURR)
10) Establecer NIC en modo INICIAR (CR = 22h). La DMA de recepción local no esta activo aun, debido a que la NIC se encuentra en modo LoopBack.
11) Inicializar el Registro de Configuración de la Transmisión con un valor razonable, El NIC se encuentra ahora listo para la transmisión y recepción.

3.1.1 Implementación del procedimiento de inicialización

A continuación se muestra el código necesario para la inicialización, este código es valido para chips compatibles con el estándar NE2000.

```
// seleccionar pagina 0, detener NIC

outp(PuertoES, 0x21);

// FIFO, comando enviar ejecutado, operación normal, DMA 16 bits, big endian, word

outp(PuertoES + 0xe, 0x59);                    // configuración de datos

// Establecer registros del contador de bytes de datos remotos = 0

outp(PuertoES + 0xa, 0x0);
outp(PuertoES + 0xb, 0x0);
```

```
// Establecer registro de configuración de la recepción (RCR)
// Modo monitor con buffers en memoria, todos los paquetes con dirección destino física son aceptados,
// paquetes multicast son aceptados, paquetes broadcast son aceptados, paquetes < 64 bytes son rechazados
// paquetes con errores son rechazados

outp(PuertoES + 0xc, 0x1c);

// TPSR - Registro Inicio de Pagina para Transmisión
// Establece la dirección de inicio de la pagina del paquete a trasmitir

outp(PuertoES + 0x4, 0x7a);                    // tx pagina

// TCR - Registro de Configuración de la Transmisión)
// generar y verificar CRC en modo normal, loopback interno, operación normal (sin auto transmisión),
// desplazamiento de transmisión deshabilitado

outp(PuertoES + 0xd, 0x2);

// PSTART - Registro Inicio de Pagina
// Establece apuntador a la primera página del buffer de recepción circular

outp(PuertoES+1,0x40);

// BNRY - Registro de Limite
// establece apuntador a la ultima página que la PC a leído del buffer de recepción

outp(PuertoES+3,0x40);

// PSTOP - Registro Fin de Pagina
// establece apuntador a la ultima página del buffer circular de recepción

outp(PuertoES+2,0x79);

// Registro de Comandos 0x0
// activar página de registro 1, detener NIC

outp(PuertoES,0x61);

// CURR - Registro de Pagina Actual
// Este apunta a la  página del buffer de recepción donde puede guardarse el siguiente paquete

outp(PuertoES+7,0x41);

// Registro de Comandos 0x0
// seleccionar pagina de registros 0, Activar DMA remota, detener NIC

outp(PuertoES,0x21);

// ISR - Registro de Estado de Interrupciones
// NIC en estado reset, operación remota finalizada, byte mas significativo de contadores establecido,
// buffer de recepción desbordado, muchas colisiones, error varios, transmisión-recepción sin errores

outp(PuertoES+7,0xff);

// IMR - Registro de Mascara de Interrupciones
// 1 - Habilita las interrupciones descritas en el registro ISR

outp(PuertoES+0xf,imr);
```

```
// TCR - Registro de Configuración de la Trasmisión 0xd
// generar y verificar CRC en modo normal, operación normal(no loopback), operación normal (sin
// autotrasmisión), desplazamiento de transmisión deshabilitado

outp(PuertoES+0xd,0);

// RC - Registro de Comandos 0x0
// seleccionar página de registros 1, detener NIC

outp(PuertoES,0x61);

// MAR0-7 - Registros de Dirección Multicast

outp(PuertoES+8,0xff);
outp(PuertoES+0x9,0xff);
outp(PuertoES+0xa,0xff);
outp(PuertoES+0xb,0xff);
outp(PuertoES+0xc,0xff);
outp(PuertoES+0xd,0xff);
outp(PuertoES+0xe,0xff);
outp(PuertoES+0xf,0xff);

// RC - Registro de Comandos 0x0
// seleccionar página de registros 0, detener NIC

outp(PuertoES,0x21);

// activar la IRQ asociada a la tarjeta de red

outp(picport, picval);

// seleccionar  página de registro 0, Activar DMA remota, activar NIC

outp(PuertoES,0x22);
```

3.2 Transmisión de Paquetes

El controlador de transmisión generalmente se encuentra dividido en 2 partes. La primera parte (*EnviarPaquete*) comienza una transmisión siempre que el software del nivel superior le pase un paquete al driver. EnviarPaquete verifica si la NIC puede transmitir el paquete, leyendo el registro CR (bit TXP = 0).

Si esta listo, EnviarPaquete usa el canal DMA remoto para transmitir de la PC a la memoria local de la NIC, emite el comando enviar y retorna inmediatamente. De otra forma si el transmisor está ocupado (bit TXP = 1) el paquete se encola en un buffer de transmisiones pendientes. Después de inicializar el envió o encolar el paquete, *EnviarPaquete* retorna.

EnviarPaquete opera junto con una rutina de servicio de interrupción (*RutinaDeServicioDeInterrupción*). Después de completar la transmisión, la NIC interrumpe al CPU para señalar el final de la transmisión e indicar información de estado en el registro de Estado de la Transmisión que la Rutina de Servicio de Interrupción lee y luego transmite el siguiente paquete pendiente en la cola, si lo hay.

Así, cuando sucede una interrupción por transmisión, la rutina de servicio de interrupción realiza los siguientes pasos:

1. Restablece el bit PTX en el registro de Estado de Interrupción.
2. Verificar la correcta transmisión por medio del Registro Estado de la Transmisión.
3. Si hay más paquetes en la cola de transmisiones pendientes, transmitir el siguiente paquete; de lo contrario ir al paso 4.
4. Leer Registro Estado de Interrupción por alguna interrupción pendiente.

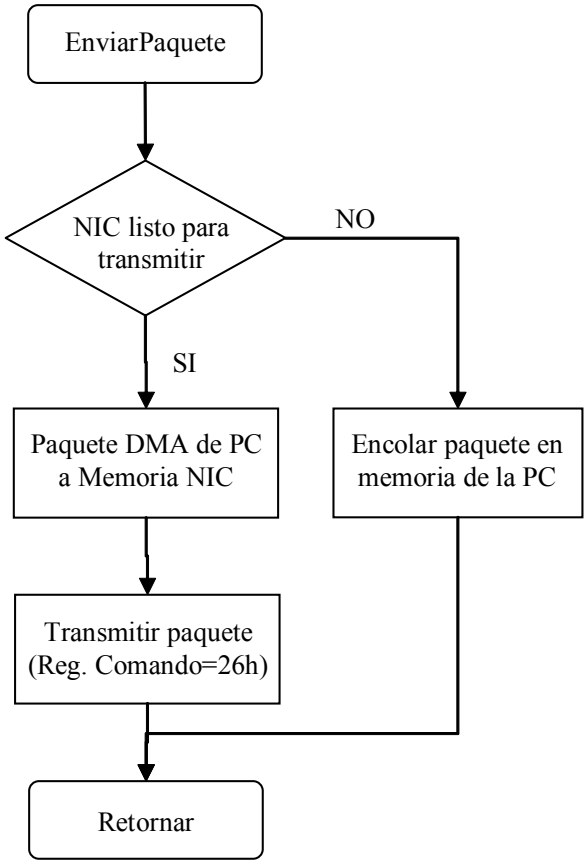

Figura 2 *Rutina Enviar Paquete*

3.2.1 Implementación del procedimiento de transmisión

```
// Enviar

if (enviar && (inp(PuertoES) != 0x26))     // si se desea transmitir y no hay otra transmisión e progreso
{       asm  cli                           // desactivar interrupciones

        // Dividir el paquete en boques de 256 bytes y enviar

        for (i=0; i<(LongPaqueteTrasmitir / 256); i++)
        {       PCaNIC(0x7a00+(i << 8),i << 8);
        }

        // Establece apuntador a pagina del paquete a trasmitir

        asm  mov dx,PuertoES
        asm  add dx,4
        asm  mov al,0x7a
        asm  out dx,al

        // Establecer contador de bytes a trasmitir

        asm  inc dx
        asm  mov ax, LongPaqueteTrasmitir
        asm  out dx,al
        asm  inc dx
        asm  mov al,ah
        asm  out dx,al

        // Comando enviar

        asm  sub dx,6
        asm  mov al,0x26
        asm  out dx,al

        asm  sti                           // Activar interrupciones
}


// Escribe 256 bytes del buffer de envío del sistema a partir de la posición indicada, al
// buffer interno de la tarjeta (en la pagina especificada) y establece la terminación de
// la lectura DMA en el registro ISR de la NIC.

void PCaNIC(unsigned int Pagina, unsigned int PosicionEnBuffer)
{       asm mov dx,PuertoES
        asm add dx,8                       // RSAR0,1 - Registro de Dirección de Inicio Remota

        asm mov ax,Pagina                  // Dirección de inicio de la DMA remota
        asm out dx,ax                      // Establecer la dirección de inicio de la DMA remota

        asm add dx,2                       // RBCR0,1 - Registro Contador de Bytes de la DMA Remota
        asm mov ax,256                     // contador = 256
        asm out dx,ax                      // Establecer contador de bytes de la DMA remota
        asm sub dx,0xa                     // CR - Registro de comandos
        asm mov al,012h                    // pagina 0, escritura remota, activar NIC
        asm out dx,al                      // enviar comando

        asm mov cx,128                     // 128 datos de 16 bits
```

```
        asm add dx,0x10              // puerto de DMA remota
        asm mov si,DirSendBuffer     // Dirección de inicio del buffer de envío
        asm mov bx,PosicionEnBuffer
        asm add si,bx                // agregar a dirección del búfer la posición deseada

        asm rep outsw                // escribir datos del buffer de envío a la NIC

        asm sub dx,9                 // ISR - Registro de Estado de Interrupciones
        asm mov al,0x40              // Se completo la operación DMA remota
        asm out dx,al                // establecer registro
}
```

3.3 Recepción de Paquetes

La responsabilidad de controlador de recepción es, transferir datos del Buffer de Recepción Circular a la memoria de la PC. Idealmente, este proceso debe de ser realizado tan rápido como sea posible para eliminar cualquier cuello de botella que pudiera deberse al driver.

Dado que el controlador de recepciones es muy dependiente de las interrupciones, este reside completamente dentro de la rutina de servicio de interrupción. Cuando la interrupción de recepción sucede, uno o más paquetes pueden haber sido guardados en el buffer de recepción circular por la NIC. La rutina de servicio de interrupción mueve paquetes del buffer de recepción circular a la memoria de la PC por medio del comando "enviar paquete", los paquetes son removidos hasta que no que de ninguno, que es cuando los registros CURRENT y BOUNDARY son iguales.

La secuencia de la rutina de recepción es la siguiente:

1. Restablecer el bit PRX del Registro de Estado de Interrupciones.
2. Remover el siguiente paquete del buffer de recepción, usando el comando"enviar paquete".
3. Verificar si el buffer de recepción se encuentra vació: registro BOUNDARY = registro CURRENT.
4. Si el buffer de recepción no esta vació, ir al paso 1; de otra forma, leer el Registro de Estado de Interrupciones por alguna interrupción pendiente.

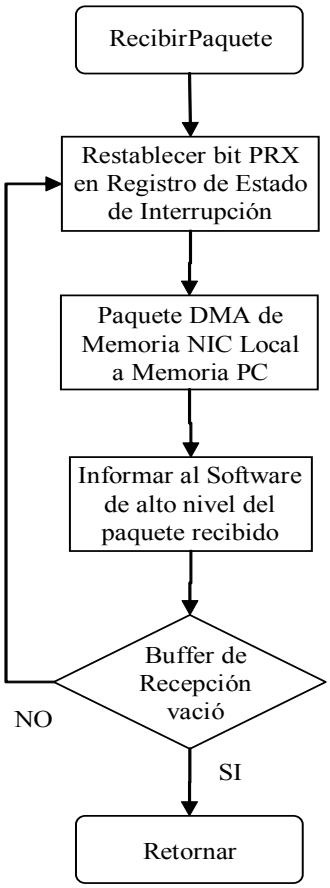

Figura 2 *Rutina Recibir Paquete*

3.3.1 Implementación del procedimiento de recepción

```
void interrupt isr(__CPPARGS)
{           // Código de inicio para control de interrupciones común

            EstadoDeNIC = inp(PuertoES+7);          // leer el estado de la NIC

            // Si se detecto error en el paquete recibido

            if ((EstadoDeNIC & 4) == 4)
            {           outp(PuertoES+7,4);             // desactivar bit indicador de error en transmisión
                        ContadorErroresRecepcion++;     // incrementar contador de errores

                        // Leer Registro de Estado de la Recepción

                        RegistroEstadoDeRecepcion = inp(PuertoES+0xc);
```

```
		// Verificar si se trata de un error en CRC

		if ((RegistroEstadoDeRecepcion & 2) == 2)
		{		ContadorErroresCRC++;
		}

		// Verificar error de alineación del marco

		if ((RegistroEstadoDeRecepcion & 4) ==4)
		{		ContadorErroresAlineacion++;
		}

		// Verificar error de desbordamiento FIFO

		if ((RegistroEstadoDeRecepcion & 8) == 8)
		{		ContadorErroresDesbordamientoFIFO++;
		}

		// Verificar si hubo portadora o colision
		if ((RegistroEstadoDeRecepcion & 0x80) == 0x80)
		{		ContadorErroresPortadora++;
		}
}

// Si se detecta desbordamiento del buffer de recepción

if ((EstadoDeNIC & 0x10) == 0x10)
{		ContadorDesbordamientoBuffer++;

		outp(PuertoES,0x21);		// pagina 0, Activar DMA remota, detener NIC
		delay(2);			// esperar un momento

		outp(PuertoES+0xa,0);		// establecer contador de datos DMA remotos a 0
		outp(PuertoES+0xb,0);

		// verificar CRC en modo normal, loopback interno, operación normal(sin
		// autotransmisión), desplazamiento de transmisión deshabilitado

		outp(PuertoES+0xd,2);

		outp(PuertoES,0x22);		// pagina 0, Activar DMA remota, activar NIC

		NICaPC(SiguientePaquete,0);	// Recibir datos

		SiguientePaquete = BufferDeRecepcion[1] << 8;

		// obtener puntero a donde el siguiente paquete puede ser guardado

		btemp=BufferDeRecepcion[1]-1;

		// ajustar pagina inicial de lectura de la NIC

		if (btemp < 0x40)
		{		btemp = 0x79 - 1;
		}

		outp(PuertoES+0x3,btemp);	// establecer apuntador a la ultima pagina leída
		outp(PuertoES+0x7,0x10);	// desactivar bit indicador desbordamiento de buffer
```

```
            // generar y verificar CRC en modo normal, operación // normal, operación normal (sin
            // auto transmisión), desplazamiento de transmisión deshabilitado

            outp(PuertoES+0xd,0);
}

// Si se trasmitió paquete sin errores

if ((EstadoDeNIC & 2) == 2)
{           outp(PuertoES+0x7,2);          // desactivar bit indicador de transmisión sin errores

            ContadorEnvioOK++;

            // leer registro de estado de la transmisión

            RegistroEstadoDeRecepcion=inp(PuertoES+0x4);

            // Verificar si hubo colisión del paquete trasmitido

            if ((RegistroEstadoDeRecepcion & 4)==4)
            {           // Obtener la cantidad de colisiones que se detectaron durante el
                        // envió del paquete y agregar al contador de colisiones global

                        ContadorDeColisiones=ContadorDeColisiones+(long) inp(PuertoES+0x5);
            }

            // Verificar si hubo perdida de portadora durante la trasmisión del paquete

            if ((RegistroEstadoDeRecepcion & 0x10) == 0x10)
            {           ContadorPerdidaPortadora++;
            }

            // Verificar si hubo colisión

            if ((RegistroEstadoDeRecepcion & 0x80) == 0x80)
            {           ContadorDeColisiones++;
            }
}

// Si hubo error de transmisión, debido a demasiadas colisiones

if ((EstadoDeNIC & 8) == 8)
{           outp(PuertoES+7,8);            // desactivar bit indicador de error de transmisión
            ContadorErroresEnvio++;

            // leer registro de estado de la trasmision

            RegistroEstadoDeRecepcion = inp(PuertoES+0x4);

            // Si se interrumpio trasmision debido a demasiadas colisiones

            if ((RegistroEstadoDeRecepcion & 8) == 8)
            {           ContadorDeColisiones = ContadorDeColisiones + 16;
            }
```

```
		// si hubo colisión del paquete trasmitido

		if ((RegistroEstadoDeRecepcion & 4) == 4)
		{		// Obtener la cantidad de colisiones que se detectaron durante el
				// envio del paquete y agregar al contador de colisiones global

				ContadorDeColisiones = ContadorDeColisiones + (long) inp(PuertoES+5);
		}

		// Verificar si hubo colisión

		if ((RegistroEstadoDeRecepcion & 0x80) == 0x80)
		{		ContadorDeColisiones++;
		}

		// Verificar si hubo perdida de portadora durante la transmisión del paquete

		if ((RegistroEstadoDeRecepcion & 0x10) == 0x10)
		{		ContadorPerdidaPortadora++;
		}
}

// Si se recibió paquete de forma correcta

if((EstadoDeNIC & 1)==1)
{		outp(PuertoES+7,1);			// limpiar bit recibido
		buff = 0;
		ContadorRecepcionOK++;
		NICaPC(SiguientePaquete,0);		//  recibir paquete

		btemp=inp(PuertoES+3);			// actualizar limite; liberar espacio del buffer
		btemp++;

		if (btemp > 0x78)
		{		btemp = 0x40;
		}

		outp(PuertoES+3,btemp);
		contador = 256;

		while(contador<(unsigned int)(BufferDeRecepcion[3] << 8 + BufferDeRecepcion[2]) )
		{		buff = buff+256;
				SiguientePaquete=SiguientePaquete+256;

				// verificar limites del buffer

				if (SiguientePaquete > 0x7800)
				{		SiguientePaquete = 0x4000;
				}

				NICaPC(SiguientePaquete,buff);
				contador = contador + 256;
				btemp = inp(PuertoES+3);	// actualizar limites; liberar espacio del buffer
				btemp++;

				if (btemp > 0x78)
				{		btemp = 0x40;
				}
```

```
                    outp(PuertoES+3,btemp);
                }

                SiguientePaquete = BufferDeRecepcion[1] << 8;
                btemp = BufferDeRecepcion[1]-1;

                if (btemp < 0x40)
                {       btemp = 0x78;
                }

                outp(PuertoES+3,btemp);                 // mover apuntador de limite
                asm mov NuevoPaqueteRecibido,TRUE       // activar bandera de recepción de paquete
        }

        // Código de salida común
}


// Envía 256 bytes del buffer interno de la tarjeta (en la pagina especificada) al buffer de recepción del
// sistema en la posición indicada y establece la terminación  de la lectura DMA en el registro ISR de la NIC.

void NICaPC(unsigned int pagina, unsigned int PosicionEnBuffer)
{       asm mov dx,PuertoES
        asm add dx,8                            // RSAR0,1 - Registro de Dirección de Inicio Remota
        asm mov ax,pagina                       // Dirección de inicio de la DMA remota
        asm out dx,ax                           // Establecer la dirección de inicio de la DMA remota

        asm add dx,2                            // RBCR0,1 - Registro Contador de Bytes de la DMA
                                                // Remota
        asm mov ax,256                          // contador = 256
        asm out dx,ax                           // Establecer contador de bytes de la DMA remota

        asm sub dx,0xa                          // CR - Registro de comandos
        asm mov al,0ah                          // pagina 0, lectura remota, activar NIC
        asm out dx,al                           // enviar comando

        asm mov cx,128                          // 128 datos de 16 bits
        asm add dx,0x10                         // puerto de DMA remota
        asm mov di,DirBuffer                    // Dirección de inicio del buffer de recepción
        asm mov bx,PosicionEnBuffer
        asm add di,bx                           // agregar a dirección del buffer la posición deseada

        asm mov ax,ds
        asm mov es,ax

        asm rep insw                            // leer datos de la NIC al buffer de recepción

        asm sub dx,9                            // ISR - Registro de Estado de Interrupciones
        asm mov al,0x40                         // Se completo la operación DMA remota
        asm out dx,al                           // establecer registro
}
```

4 Resultados Experimentales

Hemos desarrollado un software que nos ofrece soporte básico de comunicaciones siguiendo los lineamientos expuestos anteriormente y basados en los detalles de las especificaciones publicadas por los fabricantes de la NIC *Realtek RTL8029AS* [2] y *Nacional Semiconductor DP8390D*[3]. Aunque no se cuenta con el material suficiente para realizar otros tipos de prueba, con la única NIC con que se cuenta se han probado su modo de envío Broadcast (Envío hacia todos) y el modo de recepción múltiple (promiscuo), en el cual la NIC acepta todos los paquetes que le lleguen, sin importar que no vayan dirigidos a ella específicamente.

También se recolecta información estadística sobre la cantidad de paquetes enviados y recibidos, así como sus respectivos errores por colisión, perdida de portadora, desbordamiento de buffer, CRC, etc.

A continuación un listado de los resultados de la ejecución:

NE2000 en puerto de E/S **ec00** IRQ **11**
Dirección EEPROM **0485481531b**

Tx Total OK	: 2094	---A MAC----	--De MAC----	--Tipo---	Long
Rx Total OK	: 1324	ffffffffffff	0c090fdf61b8	IP	fd
Tx Total de Errores	: 9	ffffffffffff	0c090fdf61b8	IP	60
Rx Total de Errores	: 4	ffffffffffff	0c090fdf61b8	IP	60
Tx Colisiones	: 0	ffffffffffff	0c090fdf61b8	IP	60
Tx Perdida de portadora	: 1	ffffffffffff	0c090fdf61b8	IP	5a
Rx Sobreflujo de Buffer	: 3				
Rx Error de CRC	: 0				
Rx Error en Frame	: 1				
Tx Sobreflujo de FIFO	: 0				
Rx Desbordamiento FIFO	: 0				

Ultimo paquete de datos extraído de la memoria NIC

```
21 5e 5a 01 ff ff ff ff ff ff 00 c0
9f 0d f6 1b 08 00 45 00 00 e5 00 8e
00 00 80 11 7d f7 a9 fe 67 86 a9 fe
ff ff 00 8a 00 d1 7b 00 00 00 00 00
45 00 00 e5 00 8e 00 00 e5 64 fe ef
00 80 11 7d f7 11 7d f7 a9 fe 67 86
f7 a9 fe 67 00 d1 45 00 00 d1 7b 00
```

5 Conclusiones y Trabajos Futuros

Se a visto que muchas veces se nos provee de funcionalidad que realmente no necesitamos en la mayoría de los casos, sin embargo esta funcionalidad a veces no necesaria, si esta consumiendo recursos en todo momento, obviamente las abstracciones de alto nivel tienen sus ventajas pero muchas veces nos encontramos con que se esta subutilizando la capacidad real de nuestros equipos debido a esta excesiva cantidad de funcionalidad, es por eso que pensamos que en el futuro predominaran los sistemas dinámicamente extensibles que serán capaces de utilizar solo lo que necesiten para llevar a cabo sus actividades.

Y al fin nuestros equipos podrán liberarse de la pesada armadura que pone sobre ellos el software actualmente. Por eso nuestro interés en desarrollar más adelante una Capa Básica de Comunicaciones (BCL), basada en el soporte básico de comunicaciones aquí expuesto en cuanto a detalles de implementación y posiblemente en otros soportes físicos.

6 Referencias

[1] Hwang Kai y Xu Zhiwei
Scalable Parallel Computing: Technology, architecture and Programming
1rst. edition, McGraw-Hill

[2] REALTEK SEMI-CONDUCTOR CO., LTD.
"RTL8029AS" Realtek PCI Full-Duplex Ethernet Controller with built-in SRAM

[3] National Semiconductor Corporation.
DP8390D/NS32490D NIC Network Interface Controller
National Semiconductor Corporation, Julio 1995