Hospedaje Linux de multicomputadores DSP

Irala Veloso, Isidro; Díaz Martín, Juan Carlos;
Rodríguez García, José Manuel; García Zapata, Juan Luis;
Matías Martín, Jorge;
{juancarl, iirala, jmrodri, jgzapata}@unex.es, j_martin@teleline.es
Escuela Politécnica, Avda. Universidad, s/n, 10071, Cáceres, Spain
Fax: 34 27 257202,Phone: 34 27 257265

11 de noviembre de 2000

Esta ponencia forma parte del III Congreso Hispalinux[1] virtual, celebrado en Noviembre de 2000. Puedes participar en él en la sección de eventos de BarraPunto[2]

Resumen

Este trabajo presenta, hasta donde conocemos, el primer driver Linux de libre distribución para un multicomputador comercial DSP de propósito general. Ha sido desarrollado con el fin de facilitar la construcción de aplicaciones distribuidas de procesado de voz sobre una red DSP Blue-Waveä QPC-C40S. Describimos las aplicaciones objetivo, la interfaz hardware PC-QPC y la estructura y funcionalidad del driver.

1. Introducción

El procesado y reconocimiento de voz es un campo del proceso digital de la señal (DSP) que está experimentando una expansión rápida y prometedora. Estas aplicaciones DSP ([ALV00]) imponen una carga computacional que supera los recursos de los microprocesadores actuales. Así, se hace necesario la construcción de motores paralelos DSP que exhiban dos propiedades, primero, la escalabilidad y, segundo la accesibilidad como servicio remoto de cómputo DSP ([DIA00a], [DIA00b]), según muestra la Fig. 1. Uno de estos motores DSP es la tarjeta Blue-Wave QPC-C40S ([LOU96]), un multicomputador DSP.

[1] `http://congreso.hispalinux.es`
[2] `http://barrapunto.com/eventos`

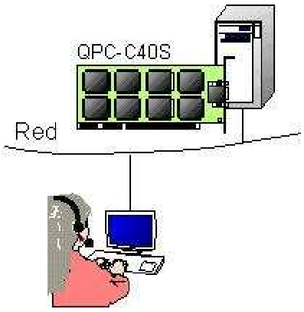

Fig. 1. Computación DSP como servicio remoto

Tradicionalmente, los fabricantes de hardware DSP proporcionan paquetes de soporte para sus productos para los sistemas operativos Windows y BlueWave no es una excepción. Así, a pesar de la robustez y estabilidad atribuida a Linux, este ha sido sistemáticamente ignorado en el mercado de plataformas DSP.

Este trabajo presenta, hasta donde conocemos, el primer driver Linux de libre distribución para un multicomputador comercial DSP de propósito general. Este driver se utiliza regularmente en proyectos de investigación sobre paralelización de algoritmos de reconocimiento y procesado de voz. El resto del artículo se estructura como sigue. La sección 2 relaciona nuestro trabajo con el trabajo de otros. La sección 3 esboza el multicomputador DSP QPC-C40S, basado en componentes Texas Instrumentsä. La sección 4 esboza el perfil de las aplicaciones DSP sobre el multicomputador objeto de investigación. La sección 5 describe con algún detalle el hardware de interface PC-QPC. La sección 6 presenta el uso del driver en las aplicaciones de usuario. La sección 7 muestra las conclusiones del trabajo.

2. Trabajo relacionado

Hunt Engineering, compañía especializada en proceso de señal en tiempo real y procesado de imagen, proporciona soporte Linux para algunos de sus productos DSP desde abril del 2000. Así, aparentemente, el desarrollo de su driver Linux parece coincidir en el tiempo con el nuestro. Ahora bien, es un software comercial distribuido sin fuente.

3. £Qué es un multicomputador DSP?

Cada placa QPC-C40S0 admite hasta cuatro módulos TIM, cada uno de ellos con dos procesadores DSP C44 ([TEX93]), según muestra la Fig. 2. El procesador C44 es uno de los más utilizados en sistemas paralelos DSP. Proporciona un rendimiento de 60 MFLOPS y dispone del soporte hardware necesario para configurarse como nodo de una red paralela con arquitecturas SIMD o MIMD.

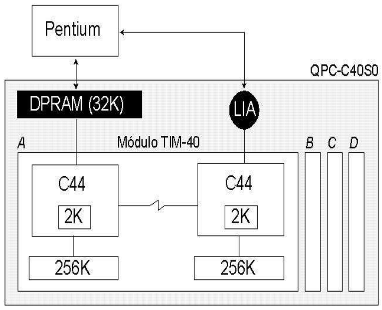

Fig. 2. Plataforma paralela DSP QPC-C40S

La comunicación entre el PC huésped y la red QPC se realiza mediante dos vías, primero, un enlace serie de 160 Mbits/s al segundo C44 del primer módulo TIM denominado LIA y, segundo, un bloque de memoria de doble puerto de 32K octetos mapeada en la memoria del PC y del primer C44 del primer módulo TIM, denominado DPRAM.

Un servicio DSP como el de la Fig. 1 puede ser escalado fácilmente, primero, incorporando nuevos módulos TIM a una placa y, segundo, incorporando nuevas placas al bus del PC. La escalabilidad puede continuar en una tercera dimensión, fuera del estándar TIM, disponiendo más de un PC huésped en la red local.

4. Perfil y evolución de la aplicación objetivo

Nuestro trabajo de investigación está dirigido hacia la paralelización de aplicaciones de procesado de voz sobre redes DSP. La Fig. 3 muestra un sistema de reconocimiento de voz de palabras aisladas, en la cual se pueden identificar cinco etapas de proceso dispuestas en pipeline: Cancelador de ruido, extractor de características, codificación de plantillas y parsing de Markov.

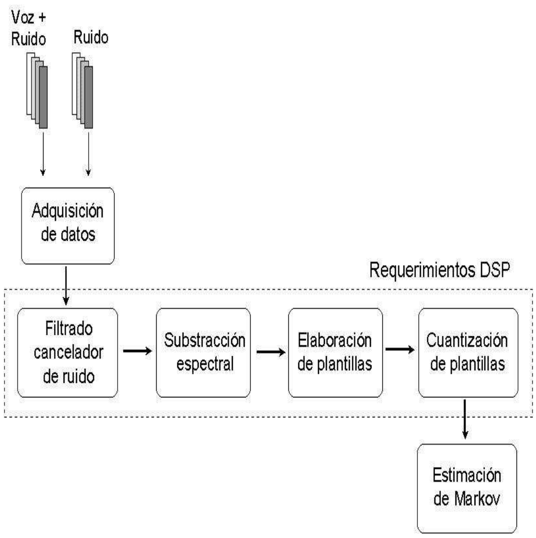

Fig. 3. Reconocimiento de voz como pipeline 1D

La Fig. 4 muestra una implementación experimental de la Fig. 3. El hardware DSP consiste en dos tarjetas ISA, cada una de ellas con un procesador DSP Texas Instruments C32 ([GOM98]) gobernadas por el sistema operativo DOS.

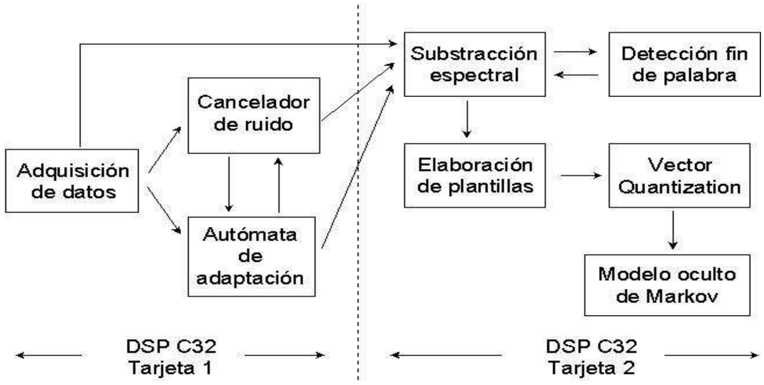

Fig. 4. Una implementación experimental DSP de reconocimiento de voz

La implementación de la Fig. 4, concebida como un coprocesador DSP de bajo costo para un PC de sobremesa, es excesivamente rígida para incorporar nuevas características y hace difícil extenderla como un servicio de red. Es preciso que el servico sea escalable, tanto en hardware como en software para adoptar sin traumas los algoritmos y aplicaciones procesado de voz presentes y futuros. Así, desarrollos más ambiciosos exigen hardware multicomputador y software abierto más sofisticado en el PC huésped para proporcionar:

1.Un sevicio de red como el de la Fig. 1.

2.Un entorno de desarrollo flexible

Linux es nuestra elección. El servidor se estructura entorno a un objeto multihilo denominado sesión, que ilustra la Fig. 5. Cada sesión gestiona una conexión bidereccional por cada cliente de la Fig. 1.

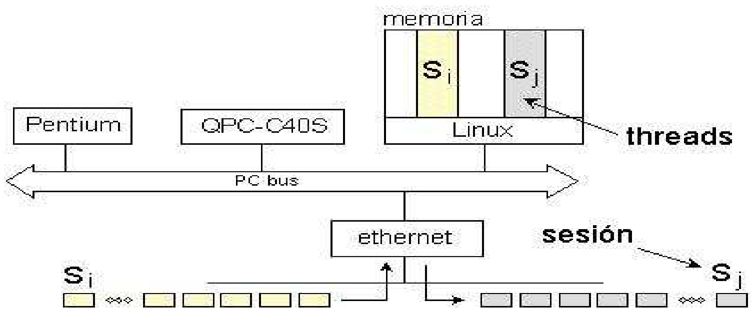

Fig. 5 El concepto de sesión

Así, la Fig. 6 muestra la paralelización de la aplicación sobre la red QPC-C40S. A medida que son aceptadas, las sesiones ocupan los pipelines disponibles hasta que estos se agotan. Entonces entran en concurrencia en cada C44. Para soportar la concurrencia se está desarrollado ViTReOS ([DIA00c]), un sistema operativo de tiempo real conforme POSIX 1003.13 ([IEE98]) para la arquitectura C44.

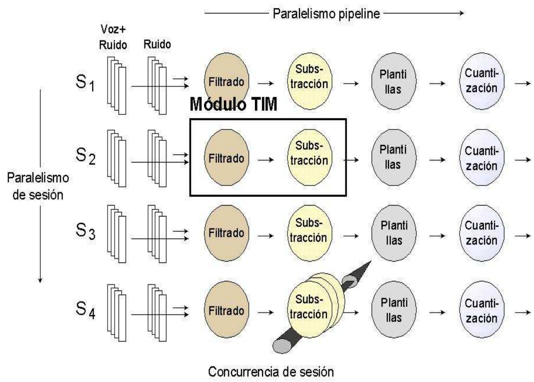

Fig. 6. Partición paralela de un algoritmos de reconocimiento de voz robusto

Todo el multicomputador QPC ha sido modelado como un dispositivo que extrae los centroides que alimentan el modelo de Markov según muestra la Fig. 7.

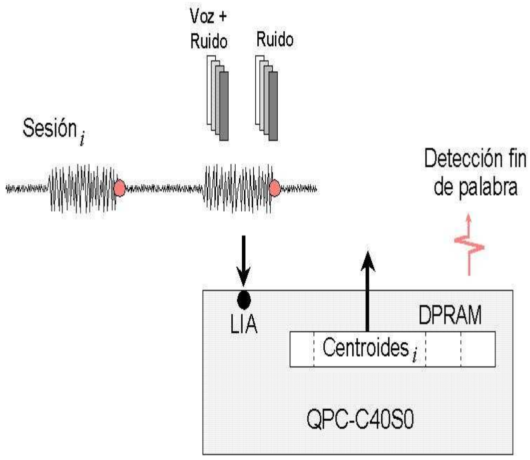

Fig. 7. La red QPC como un extractor de formantes.

5. El hardware de interfaz PC-QPC

La Fig. 8 muestra un esquema general de la interfaz PC huésped-QPC/C40S.

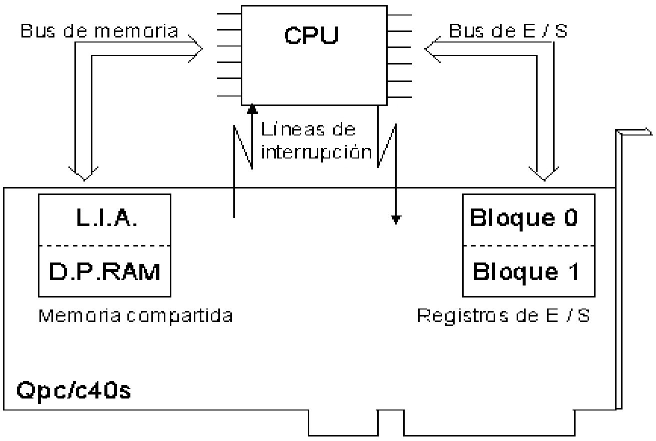

Fig. 8. Esquema de canales de comunicación PC-QPC/C40S

La interfaz entre el PC y la tarjeta consiste en tres recursos de comunicación:

- Dos bancos de registros de entrada salida. Cada banco es un conjunto de 16 registros con direcciones consecutivas. Hay registros de sólo escritura para configurar la tarjeta y registros de sólo lectura para conocer su estado. En la arquitectura PC el acceso a los registros de entrada salida se hace a través del bus de entrada salida.
- Una zona de memoria compartida (DPRAM y LIA). Es una memoria de 128 KB que físicamente reside en la tarjeta pero que se mapea en la memoria del PC.
- Líneas de interrupción en ambos sentidos entre el PC y la tarjeta. Este recurso permite al driver implementar operaciones de E/S bloqueantes.

La QPC-C40S es un adaptador que conecta en un slot ISA, de modo que el mapeo se realiza entre los 640KB y el primer Mega de la memoria física del PC. Las posibles direcciones de comienzo de esta zona de memoria son:

0A 0000h, 0B 0000h, 0C 0000h y 0D 0000h.

En el manual de la QPC este valor es denominado Memory Base o MBASE. La dirección elegida se escribe en el registro Base Address Register del bloque 0 (junto con la dirección del bloque 1 de registros). El mapeo se raliza automáticamente por hardware. La organización interna de esta memoria es la siguiente:

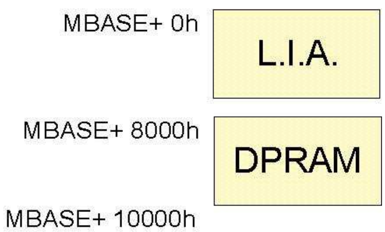

Fig. 9 División interna de la memoria compartida.

Hay que tener en cuenta que un proceso de usuario maneja direcciones virtuales de su espacio de direccionamiento, y no direcciones físicas del PC. Por eso para acceder a esta zona de memoria física hay que hacerlo desde un módulo añadido al sistema operativo o mapear esa memoria en un buffer de usuario con la llamada mmap. En este caso hemos hecho lo primero, el driver ofrece operaciones de lectura y escritura (read y write) sobre la memoria compartida, para que los procesos de usuario puedan utilizarla.

5.1 Link Interface Adapter (L.I.A.)

El C44 dispone de 4 canales de comunicación que lo conectan con el exterior (el procesador C40 tiene 6) y hacen del mismo un procesador idóneo para el proceso paralelo. Estos puertos interconectan dos procesadores C44 o bien comunican al procesador con un puerto externo del módulo TIM o de la tarjeta. Están mapeados en la memoria del C40, pudiéndose leer o escribir libremente ya que están controlados por una unidad de arbitraje que regula la comunicación bidireccional.

La interfaz LIA ocupa los primeros 32 KB de la memoria compartida (Fig. 9). Puede verse como un medio para intercambiar datos entre el PC y un C44, donde la comunicación es bidireccional y la unidad de intercambio son 4 octetos (Fig. 10).

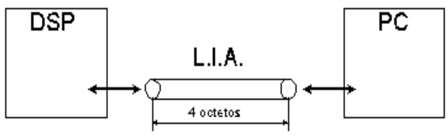

Fig. 10 Esquema de operación de LIA.

Desde el punto de vista del procesador DSP, LIA es un puerto más de comunicación, sin ninguna diferencia especial. Del lado del PC, la interfaz LIA es el resultado de mapear un puerto de un C44 en 4 Bytes de la memoria compartida del PC. Esta interfaz está indicada para el intercambio de flujos de datos.

Los puertos de comunicación de los C44 son rutados por hardware. Cada uno se conecta a otro procesador o dispositivo, dependiendo de la decisión de los fabricantes del módulo TIM y de la tarjeta. En nuestra QPC-C40S el puerto número 2 del módulo A (A2) es rutado hacia el módulo B, LIA está conectada al puerto 5 del módulo A, y dicho puerto está conectado al puerto 5 del procesador II. Esto significa que LIA comunica al PC con el puerto 5 del procesador II del módulo A.

Vamos a ocuparnos ahora de la interfaz LIA en el lado del PC. Aunque LIA ocupa 32 KB de la memoria compartida, sólo se utilizan los cuatro primeros Bytes. LIA se configura internamente como sigue:

M_BASE	1ž octeto / 2ž octeto
M_BASE + 2	3ž octeto / 4ž octeto
	(Sin uso)

Esos cuatro octetos forman la palabra de 32 bits a leer o escribir en cada momento. En el manual de la tarjeta no se informa de cómo se escriben y se leen los datos, ni el sitio concreto donde hay que hacerlo (dentro de la memoria destinada a LIA). El registro LIA Status / Interrupt del primer bloque contiene dos bits que informan de cuándo LIA está lista para ser leída, y cuándo está lista para ser escrita. Lo que significa que antes de usar LIA hay que leer esos bits del registro LIA Status/Interrupt. Aparentemente se puede leer y escribir en cualquier punto del rango dedicado a LIA, y el resultado es el mismo. Nuestro driver lee y escribe en los 4 primeros Bytes (como muestra la tabla anterior).

5.2 Dual Port RAM (DPRAM)

El segundo bloque de 32 KB de memoria compartida (Fig. 9) los ocupa DPRAM, un auténtico buffer compartido entre un C44 y el PC huésped, según muestra la Fig. 11. No hay ninguna restricción sobre cómo debe usarse esta memoria; todo depende del acuerdo entre los procesadores.

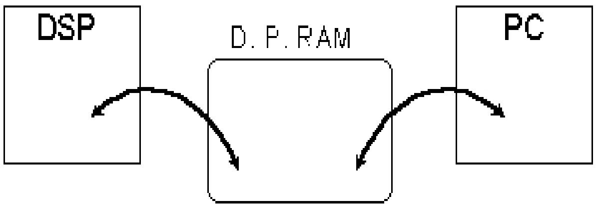

Fig. 11 Esquema de operación de DPRAM.

Como se puede apreciar esta interfaz está más indicada para intercambiar estructuras de datos, que pueden haber sido construidas en la propia memoria compartida.

Sólo el módulo TIM A tiene acceso a DPRAM, y dentro de este módulo la memoria compartida sólo está mapeada en la memoria del procesador I. Así, la interfaz DPRAM puede concretarse con la Fig.12.

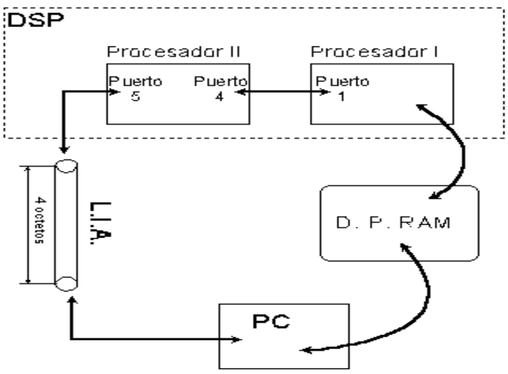

Fig. 12 Conexión de LIA y DPRAM al módulo TIM A.

Como se trata de una memoria compartida entre el PC y un C44, la gestión de este recurso queda en manos de los programadores. Los fabricantes de la tarjeta sugieren que se implemente un protocolo de semáforo, para evitar acceso simultáneo a los datos. Hay que tener en cuenta que el C44 accede a esta memoria en unidades de 32 bits, mientras que el PC accede en unidades de 8 bits.

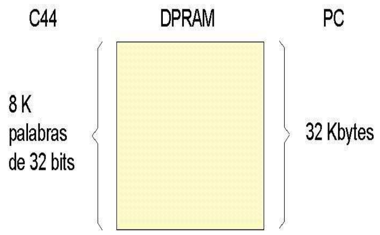

Fig. 13 Comparación en el acceso a DPRAM entre el PC y el procesador I.

En el procesador I del módulo TIM A, la memoria compartida comienza en la posición A0100000h y se extiende sobre 8 KB, teniendo en cuenta que en un C44 un Byte son 32 bits. En el PC la memoria compartida comienza 32KB (8000h) por debajo de MBASE, que es lo que ocupa LIA. En la Fig. 13, la dirección de comienzo es D8000h, porque se supone que MBASE ha sido configurado al valor 0D0000h. DPRAM se extiende en el PC 32 KB a partir de esa dirección de comienzo.

6. Uso del driver

El driver, exaustivamente documentado en [IRA00], es un módulo cargable ([RUB98]) denominado Módulo QPC/C40S, según muestra la Fig. 14.

Fig. 14. Ubicación del driver en Linux 2.0.x.

6.1. Definción de interfaz

Nuestro driver ofrece una interfaz a los procesos de usuario basada en dispositivos para gestinar los recursos de la QPC-C40S. Soporta los siguientes:

- `/dev/lia`
- `/dev/dpram0`
- `/dev/dpram1`
- `/dev/dpram2`
- `/dev/dpram3`

La interfaz /dev/lia permite realizar operaciones de lectura y escritura con un tamaño de datos múltiplo de 4 octetos. El resto de los dispositivos sirven para acceder a la memoria DPRAM, que se ha modelado como un disco RAM. Hay cuatro dispositivos dedicados a la memoria de intercambio, porque el driver realiza una división lógica de esta memoria en tres regiones, cada una de ellas es manejado por el dispositivo dpramX correspondiente sin solapamiento; excepto dpram0, que sirve para acceder a toda la memoria DPRAM en modo crudo. El número de divisiones lógicas de la memoria DPRAM y su tamaño puede variar, dependiendo de las necesidades de las aplicaciones que se ejecuten.

Las operaciones de lectura y escritura en estos dispositivos hacen accesible la memoria de intercambio a los procesos de usuario. El tamaño de los datos a intercambiar está en función del tamaño de cada partición. Las llamadas al sistema que soporta el driver son las siguientes:

- `lseek`
- `read`
- `write`
- `ioctl`
- `open`
- `close`

La llamada lseek sólo es aplicable a los dispositivos DPRAM, cada uno de ellos puede desplazar el puntero virtual en todo su rango lógico. Dpram0 , sin embargo, puede desplazar su puntero virtual en todo el rango de la memoria DPRAM.

Las operaciones read y write se pueden utilizar con todos los dispositivos. El tamaño de los datos puede ser cualquier número de octetos, excepto para LIA, que ha de ser múltiplo de cuatro. Estas llamadas son bloqueantes tanto para LIA como para DPRAM, excepto en dpram0. En LIA el proceso invocante es bloqueado si los datos que pide no están disponibles (en una operación de lectura), o cuando el dispositivo no está vacío a fin de recibir datos (en las operaciones de escritura).

El criterio para bloquear los procesos en los dispositivos dpramX se implementa con semáforos, que informan sobre quién es el propietario temporal de la memoria correspondiente. La propiedad de la memoria alterna entre el DSP y el PC cuando uno escribe y el otro lee. Por ejemplo, el procesador I del DSP escribe una cadena de caracteres en /dev/dpram1, entonces dicho procesador no podrá volver a escribir hasta que alguien (lo más normal es que sea el PC) lea de /dev/dpram1 momento en el que se cambia el semáforo.

La llamada ioctl está indicada para realizar operaciones de configuración sobre un dispositivo. Estas operaciones son dos, el reset y la inicialización de la tarjeta, las cuales configuran la tarjeta usando los registros de entrada salida. Por ejemplo:

```
/*Reset de la tarjeta.*/
if(ioctl(LIA, QPCC40S_IOCTL_RESET) != 0){
  printf("Imposible reset QPC\n");
  exit (1);
}

/*Inicializar la tarjeta.*/
if(ioctl(LIA, QPCC40S_IOCTL_INIT) != 0){
```

```
  printf("Imposible inicializar QPC\n");
    exit (1);
}
```

El parámetro LIA es un descriptor del dispositivo LIA, el segundo parámetro especifica la operación deseada. La llamada ioctl también se aprovecha para que los procesos de usuario puedan leer y escribir cualquier registro de entrada salida de la tarjeta. Para esto se utiliza el tercer parámetro de la llamada ioctl. Con él se accede a una estructura con dos campos: el número de registro y su valor. Por ejemplo, con las siguientes instrucciones se puede leer el registro de estado de la QPC:

```
DatosIoctl.NumReg = QPC_R_STATUS;
if(ioctl(descriptor,
     QPCC40S_IOCTL_LEER_REGISTRO,
     &DatosIoctl) != 0)          {
  printf("Imposible leer Reg.  Estado\n");
  exit (1);
}
```

La variable descriptor es un descriptor de cualquier dispositivo atendido por el driver. La variable DatosIoctl es una estructura de la siguiente forma:

```
struct Ioctl_Estructura{
        unsigned short ÑumReg;
        unsigned short  ValorReg;
};
```

Cuando retorna, la llamada ioctl el campo ValorReg de la estructura contendrá el valor del registro de estado de la tarjeta.

Las macros

```
QPCC40S_IOCTL_LEER_REGISTRO,
QPCC40S_IOCTL_RESET,
QPC_R_STATUS...
```

y la estructura `Ioctl_Estructura` están definidas en el fichero de cabecera del driver: `qpcc40s.h`, por lo que dicho fichero debe ser incluido en los programas de usuario.

6.2. La biblioteca de usuario QpcLib

Hemos desarrollado una biblioteca de usuario que implementa utilidades de gestión de la QPC-C40S, según muestra la Fig. 15. De momento, la utilidad más importante es el manejo de los ficheros ejecutables en formato COFF ([GIN98]), a fin cargarlos en los procesadores C44. Se espera que con el tiempo esta biblioteca vaya aumentando sus utilidades, a medida que ganemos experiencia en este trabajo.

Fig. 15. La biblioteca QpcLib facilita el manejo de la tarjeta a los procesos de usuario.

6.2.1 Gestión de errores

Cuando una función de la biblioteca encuentra un error devuelve un valor negativo, su valor absoluto especifica el tipo de error producido. Esta clasificación de los errores se ha hecho imitando la biblioteca NetAPI de Blue Wave. Cada error tiene una macro asociada en el fichero de cabecera de la biblioteca (qpclib.h), y se ha implementado una función que a partir de ese código devuelve una cadena de texto explicativo del error:

```
char* QpcLib_TextoDeError(int);
```

Por ejemplo, si se produce un error al cargar el programa COFF en el C44, se usa la función anterior para dar una información más ampliada al usuario:

```
/* Cargar el programa C40 deseado en el
 * procesador I                        */
if((ret = QpcLib_CargarPrograma(LIA,
           argv[1])) != QPCLIB_E_NOERR){
  printf("Error al cargar %s en LIA: ",
            argv[1]);
  printf("\n%s\n",
            QpcLib_TextoDeError(ret));
  exit(1);
}
```

6.2.2 Manipulación del formato COFF C40

El compilador de Texas Instruments para la arquitectura C40 genera los programas en formato COFF. Esto nos obliga a comprender este formato de ficheros objeto a la hora de usar el driver para cargar programas en los procesadores C44. No se trata de comunicar el fichero COFF, tal cual, al procesador, sino interpretar dicho fichero y comunicar la información precisa. La biblioteca QpcLib ofrece la siguiente función para realizar este trabajo:

```
int QpcLib_CargarPrograma (int, char *);
```

El primer parámetro es un descriptor del dispositivo /dev/lia, la función supone que ya está abierto y lo utiliza para escribir la información extraída del fichero COFF. El segundo parámetro es el camino para encontrar el fichero COFF en el sistema. Devuelve un código de error de la biblioteca.

Esta forma de cargar programas en un C40 obliga a tener un sistema operativo con sistema de ficheros. Puede ocurrir que el PC huésped de la tarjeta ejecute un sistema de tiempo real sin sistema de ficheros. Para solucionar esto, las rutinas que interpretan el formato COFF (fichero coff.c de la biblioteca) están controladas por una directiva de compilación que, en vez de escribir el ejecutable C40 sobre el dispositivo LIA, lo escriben en un fichero de texto, en forma de secuencia de números, de esta forma:

```
unsigned long ArrayCoff[] = {
  796360720L   /* 0x2F778010 */,
  1038852176L  /* 0x3DEBA050 */,
  1L             /* 0x1        */,
  ......       ........
```

```
  3145728L      /* 0x300000    */
};
```

Los números se escriben formando la declaración de un vector, con la intención de que el fichero resultante pueda ser incluido en los fuentes de un programa. Ese programa, a la hora de cargar el ejecutable en el C40, no usaría la función QpcLib_CargarPrograma, sino que escribiría en el dispositivo LIA cada uno de los números de este vector. De esta forma no se necesita sistema de ficheros para cargar un ejecutable en los procesadores DSP. Para ello hay que compilar la biblioteca QpcLib con la opción `coff_to_array`:

```
make coff_to_array
```

De esta forma, el fichero coff.c de la biblioteca se transforma en un ejecutable llamado coff_to_array, que admite los siguientes parámetros:

```
coff_to_array Nombre_Fichero_COFF [Nombre_Fichero_Destino]
```

El primer parámetro es el fichero objeto en formato COFF, tal como lo deja el enlazador de Texas Instruments. El segundo parámetro es el nombre del fichero donde escribir el vector de números. Si se omite, se escribe en un fichero llamado `array.hh`.

7. Conclusiones

El driver Linux que describe este artículo ha reemplazado con éxito a la utilidad Windows NetAPI que proporciona el constructor Blue Wave para el multicomputador DSP QPC-C40S. Se ha desarrollado una herramienta que permite al diseñador de aplicaciones DSP trabajar con mayor autonomía y le ofrece la posibilidad de realizar desarrollos y servicios DSP más imaginativos. El driver introduce Linux y el software de libre distribución en general en un mundo de software propietario y caro, a saber, el sistema de desarrollo en el entorno huésped -generalmente sobre Solaris para estaciones de trabajo y Windows en PC-, que el fabricante, en la práctica, impone al comprador de tecnología DSP.

Referencias

[1] [ALV00] Álvarez-Marquina, A. et al, "Speech Enhancement Through Binaural Negative Filtering", to appear in Proc. of X European Signal Processing Conference (EUSIPCO 2000), Tampere, Finland, 5-8 September 2000.

[2] [DIA00a] Díaz Martín, J.C. et al, "Distributing voice processing over DSP multicomputers. The role of POSIX 1003.13", Simposio Español de Informática Distribuida , Ourense 25-26-27 de Septiembre, 2000. http://www.ei.uvigo.es/seid2000/

[3] [DIA00b] Díaz Martín, J.C. et al, "Building predictable remote speech recognition services", The Third European DSP Education and Research Conference , Noisy le Grand, Paris, 20th & 21st September2000.

[4] [DIA00c] Díaz Martín, J.C., Irala Veloso, I. y Rodríguez García, J.M., "ViTReOS: Hacia la portabilidad de las aplicaciones paralelas DSP", XI Jornadas de Paralelismo , 11-13 de septiembre, Universidad de Granada, España.

[5] [GIN88] Gintaras R. Gircys, "Understanding and using COFF", O'Reilly & Associates, Inc. 1988.

[6] [GOM98] Gómez, P., et al, "A DSP-based Modular Architecture for Noise Cancellation and Speech Recognition", Proc. of the 1998 IEEE Int. Symp. on Circuits and Systems, ISCAS'98, Monterey, CA, USA, pp. V.178-181.

[7] [IEE98] IEEE, "Standardized Application Environment Profile - Realtime Application Support (AEP)", IEEE Std 1003.13, IEEE (1998).

[8] [IRA00] Irala Veloso, I. "Driver para la tarjeta QPC/C40S", Informe interno driver_qpc1-1.doc-26/1/-v1.0, Depto. de Informática, Univ. of Extremadura, Spain, 2000.

[9] [LOU96] Loughborough Sound Images plc., 'QPC/C40S Technical Reference Manual', Loughborough Sound Images plc, Loughborough Park, Loughborough, Leics, LE11 3NE, UK.

[10] [LSI97] Loughborough sound images, "User Documentation QPC/C40S Quad TIM-40 Carrier Board & Development System", Feb. 1997.

[11] [RUB98] Alessandro Rubini, "Linux device drivers", O'Reilly & Associates, Inc. 1998.

[12] [TEX93] Texas Instruments Inc., 'TMS320C4x User's Guide', Texas Instruments Inc., 1993. http://www.ti.com/europe/docs/univ/docs/info.htm