Instalación de Debian en un MVME2434 sin disco duro

P. de Vicente, G. Paubert (IRAM)

Informe Técnico IT-OAN 2005-6

Índice

1. Introducción

El OAN dispone de un autocorrelador de 1024 canales, fabricado en IRAM que se controlará empleando una tarjeta VME modelo 2434 de Motorola. La tarjeta autocorreladora y la del microprocesador están pinchadas en un bus VME alimentado con 5 y 12 V.

El programa de control del autocorrelador está basado en el que se emplea en **Wilma**, el autocorrelador de IRAM de última generación. Este programa corre en un sistema operativo Linux.

Este informe describe el procedimiento a seguir para que la tarjeta MVME2434 (**Pebbles** a partir de ahora) cargue Linux en su RAM utilizando la red ethernet. Para llevar a cabo con éxito esta operación son necesarias las siguientes condiciones:

- La tarjeta debe disponer de una dirección IP e información sobre la red de área loca. Esta información la puede obtener durante el inicio o la puede almacenar de modo estático.
- El firmware de la tarjeta MVME debe descargar el núcleo de Linux a través de la red local.
- Debe existir un servidor TFTP que entregue el fichero con el núcleo de Linux a través de la red local.
- El microprocesador debe resituar en memoria dicho archivo y ejecutarlo.
- Linux se debe iniciar en la tarjeta MVME y montar el sistema de 'root' mediante NFS.
- Debe existir un servidor de NFS root en la red que haga las veces de disco duro del MVME2434
- Una vez se haya arrancado Linux es necesario liberar el puerto serie para que las aplicaciones del usuario puedan utilizarlo.

En las siguientes secciones se describen las tareas realizadas para cumplir cada uno de los requisitos mencionados anteriormente.

2. La tarjeta microprocesadora PowerPC

La tarjeta microprocesadora que se usa en el OAN se puede observar en la fotografía de la figura 1. Dispone de un procesador PowerPC a 350 MHz con arquitectura PReP. La tarjeta tiene una conector de red 10/100 UTP en el frontal, y un conector RJ45 para la línea serie. También dispone de dos puertos tipo PCI Mezzazine que, al no emplearse, se desmontaron durante el tiempo en el que se realizaron las operaciones que se describen en este informe. En el frontal la tarjeta dispone de dos pulsadores: *ABT* y *RST*. El primero permite abortar cualquier comando permitiendo al usuario el acceso inmediato al firmware y el segundo reinicia la tarjeta. La tarjeta se describe con detalle en [1]

La tarjeta dispone de una PROM y 256 Mb de memoria RAM. En la memoria ROM reside el firmware PPC4 que proporciona un conjunto de comandos para controlar la tarjeta. Los comandos se lanzan desde el "prompt" PPC4-Bug y se ejecutan contestando interactivamente las

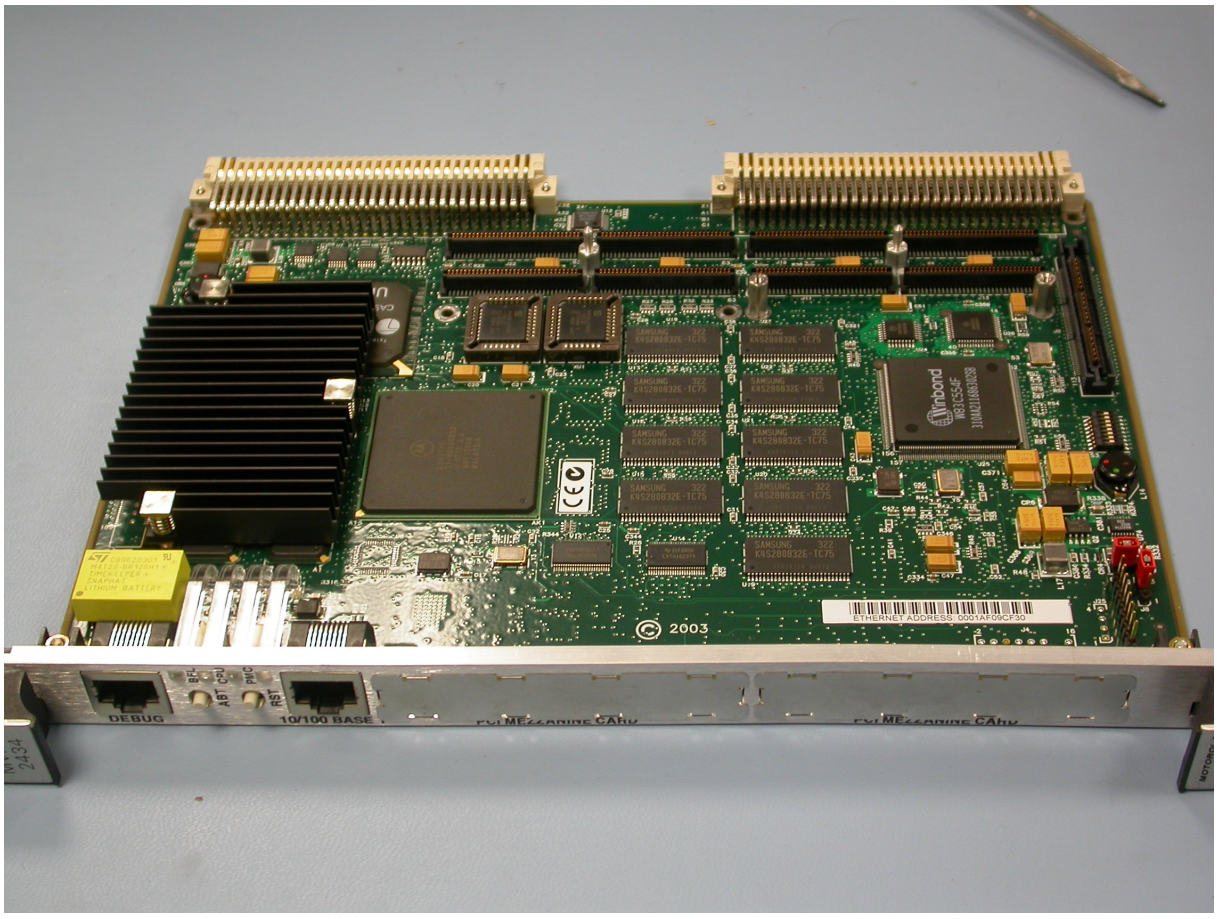

Figura 1: *Fotografía de la placa MVME2434. En el momento de la foto ya se habían desmontado los puertos auxiliares PCI Mezzazine*

preguntas que se realizan. El comando básico de configuración es **ENV**, y se describe junto con los otros comandos en [4]

El control de la tarjeta se realiza mediante un terminal externo conectado al puerto serie. La velocidad predefinida es 9600 baudios y la comunicación usa 1 bit de stop y 1 bit de paridad. Se construyó un cable serie con conector RJ11 en un extremo y DB9 en el otro empleando las siguientes conexiones:

Número de pin RJ45	Nombre	Descripción	Número de pin DB9
1	DSR/RI	Data set Ready/ring indicator	
2	DCD	Data Carrier Detect	
3	DTR	Data Terminal Ready	5
4	SGND	Signal Ground	5
5	TD	Transmit Data	3
6	RD	Receive Data	2
7	CTS	Clear to Send	
8	RTS	Request to Send	

Cuadro 1: *Conexiones entre el puerto DB9 y el puerto serie RJ45 de la tarjeta MVME*

Las conexiones de la tabla 1, aunque similares, no siguen el estándar de conexión RJ45-DB9 para puertos serie que se describe en [3].

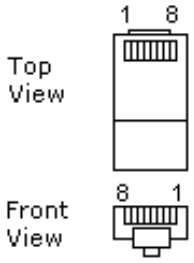

Figura 2: *Vista del conector RJ45 para puerto serie.*

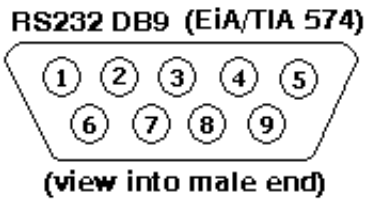

Figura 3: *Vista de un conector macho DB9 para puerto serie, con los contactos numerados.*

La configuración de la tarjeta MVME se realizó utilizando *kermit* desde un portátil con Linux. Es posible guardar un registro de todos los comandos y respuestas en un archivo histórico. La instalación, y uso de *kermit* en el portátil se resume en los siguientes comandos:

```
apt-get install ckermit
kermit
C-kermit> log session vme.log
C-Kermit> set port /dev/ttyS0
C-Kermit> set baud 9600
C-Kermit> set carrier-watch off
C-Kermit> c
```

Desde este momento obtendremos en la pantalla del portátil el “prompt” de PPC4, y podremos introducir los comandos que deseemos. Para volver al portátil es necesario pulsar la siguiente combinación de teclas (*kermit* avisa de ello al conectarse):

```
Ctrl-] c
```

Al iniciar el VME obtendremos información sobre el equipo:

```
Copyright Motorola Inc. 1988 - 1999, All Rights Reserved

PPC4 Debugger/Diagnostics Release Version 1.2 - 02/15/99 RM01
COLD Start

Local Memory Found =10000000 (&268435456)

MPU Clock Speed =350Mhz

BUS Clock Speed =100Mhz

Reset Vector Location  : ROM Bank B
Mezzanine Configuration: Single-MPU
Current 60X-Bus Master : MPU0
Idle MPU(s)            : NONE

Initializing System Memory (DRAM)...

System Memory: 256MB, ECC Enabled (ECC-Memory Detected)
L2Cache:       1024KB, 140Mhz

SelfTest/Boots about to Begin... Press <BREAK> at anytime to Abort ALL
PPC4-Bug>
```

Para que la tarjeta arranque desde red es necesario habilitar el autoarranque por red. Al ejecutar el comando **ENV** debemos mantener las opciones predefinidas excepto algunas de ellas. A continuación mostramos la configuración adecuada:

```
PPC4-Bug>env
Bug or System environment [B/S] = B?
Field Service Menu Enable [Y/N] = N?
Remote Start Method Switch [G/M/B/N] = B?
```

```
Probe System for Supported I/O Controllers [Y/N] = Y?
Auto-Initialize of NVRAM Header Enable [Y/N] = Y?
Network PReP-Boot Mode Enable [Y/N] = Y?
Negate VMEbus SYSFAIL* Always [Y/N] = N?
SCSI Bus Reset on Debugger Startup [Y/N]   = N?
Primary SCSI Bus Negotiations Type [A/S/N] = A?
Primary SCSI Data Bus Width [W/N]          = N?
Secondary SCSI Identifier                  = "07"?
NVRAM Bootlist (GEV.fw-boot-path) Boot Enable [Y/N]           = N?
NVRAM Bootlist (GEV.fw-boot-path) Boot at power-up only [Y/N] = N?
NVRAM Bootlist (GEV.fw-boot-path) Boot Abort Delay            = 5?
Auto Boot Enable [Y/N]           = N?
Auto Boot at power-up only [Y/N] = N?
Auto Boot Scan Enable [Y/N]      = N?
Auto Boot Scan Device Type List  = HDISK?
Auto Boot Controller LUN   = 00?
Auto Boot Device LUN       = 00?
Auto Boot Partition Number = 00?
Auto Boot Abort Delay      = 7?
Auto Boot Default String [NULL for an empty string] = ?
ROM Boot Enable [Y/N]            = N?
ROM Boot at power-up only [Y/N]  = Y?
ROM Boot Enable search of VMEbus [Y/N] = N?
ROM Boot Abort Delay             = 5?
ROM Boot Direct Starting Address = FFF00000?
ROM Boot Direct Ending Address   = FFFFFFFC?
Network Auto Boot Enable [Y/N]           = Y?
Network Auto Boot at power-up only [Y/N] = N?
Network Auto Boot Controller LUN = 00?
Network Auto Boot Device LUN     = 00?
Network Auto Boot Abort Delay    = 1?
Network Auto Boot Configuration Parameters Offset (NVRAM) = 00001000?
Memory Size Enable [Y/N]         = Y?
Memory Size Starting Address     = 00000000?
Memory Size Ending Address       = 10000000?
DRAM Speed in NANO Seconds       = 10?
ROM First Access Length (0 - 31) = 31?
ROM Next Access Length  (0 - 15) = 15?
DRAM Parity Enable [On-Detection/Always/Never - O/A/N]    = O?
L2Cache Parity Enable [On-Detection/Always/Never - O/A/N] = O?
PCI Interrupts Route Control Registers (PIRQ0/1/2/3) = 0A0B0E0F?
Serial Startup Code Master Enable [Y/N] = N?
Serial Startup Code LF Enable [Y/N] =     N?
VME3PCI Master Master Enable [Y/N] = N?
PCI Slave Image 0 Control                = 00000000?
PCI Slave Image 0 Base Address Register  = 00000000?
PCI Slave Image 0 Bound Address Register = 00000000?
```

```
PCI Slave Image 0 Translation Offset     = 00000000?
PCI Slave Image 1 Control                = 00000000?
PCI Slave Image 1 Base Address Register  = 00000000?
PCI Slave Image 1 Bound Address Register = 00000000?
PCI Slave Image 1 Translation Offset     = 00000000?
PCI Slave Image 2 Control                = 00000000?
PCI Slave Image 2 Base Address Register  = 00000000?
PCI Slave Image 2 Bound Address Register = 00000000?
PCI Slave Image 2 Translation Offset     = 00000000?
PCI Slave Image 3 Control                = 00000000?
PCI Slave Image 3 Base Address Register  = 00000000?
PCI Slave Image 3 Bound Address Register = 00000000?
PCI Slave Image 3 Translation Offset     = 00000000?
VMEbus Slave Image 0 Control                = E0F20000?
VMEbus Slave Image 0 Base Address Register  = 00000000?
VMEbus Slave Image 0 Bound Address Register = 10000000?
VMEbus Slave Image 0 Translation Offset     = 00000000?
VMEbus Slave Image 1 Control                = 00000000?
VMEbus Slave Image 1 Base Address Register  = 00000000?
VMEbus Slave Image 1 Bound Address Register = 00000000?
VMEbus Slave Image 1 Translation Offset     = 00000000?
VMEbus Slave Image 2 Control                = 00000000?
VMEbus Slave Image 2 Base Address Register  = 00000000?
VMEbus Slave Image 2 Bound Address Register = 00000000?
VMEbus Slave Image 2 Translation Offset     = 00000000?
VMEbus Slave Image 3 Control                = 00000000?
VMEbus Slave Image 3 Base Address Register  = 00000000?
VMEbus Slave Image 3 Bound Address Register = 00000000?
VMEbus Slave Image 3 Translation Offset     = 00000000?
VMEbus Register Access Image Control Register = 00000000?
VMEbus Register Access Image Base Address Register = 00000000?
PCI Miscellaneous Register       = 10000000?
Special PCI Slave Image Register = 00000000?
Master Control Register          = 80C00000?
Miscellaneous Control Register   = 52060000?
User AM Codes                    = 00000000?
```

donde las siguientes opciones fueron modificadas para que el equipo arranque a través de la red:

```
Network PReP-Boot Mode Enable [Y/N] = Y?
Network Auto Boot Enable [Y/N]           = Y?
```

También modificamos la opción

```
ROM Boot Abort Delay             = 5?
```

para poder depurar con mayor rapidez los problemas originados durante la descarga de ficheros por la red.

El manual “MVME2400 Series. VME Processor Module. Installation and Use” de Motorola ([2]) describe pormenorizadamente el significado de las opciones de configuración de los diferentes parámetros.

Para acceder a la red Ethernet es preciso configurar previamente la fecha y hora de la tarjeta. Para verificar ambas se debe ejecutar el comando **TIME** que mostrará la fecha y hora actualizando el último campo cada segundo. Para terminar este comando, pulsar sobre el pulsador *ABT* en el panel frontal de la tarjeta. Si la hora y la fecha no estuvieran configuradas o fueran incorrectas se pueden modificar ejecutando el comando adecuado de *PPC4*.

La tarjeta puede realizar una petición BOOTP para obtener de un servidor en la red de área local una dirección IP a partir de la dirección MAC de la tarjeta. También es posible configurar todas las características de la red, es decir, dirección IP, puerta de acceso, máscara, dirección de difusión, subred y dirección del servidor, de modo estático. El comando **NIOT** permite configurar estos campos junto con el archivo que deseamos descargar empleando TFTP. A continuación se muestra un ejemplo de configuración, en el que ya se había configurado previamente la tarjeta y en el que se busca la imagen `zImage.prep` en la ruta predefinida del servidor TFTP:

```
PPC4-Bug>NIOT
Controller LUN =00?
Device LUN     =00?
Node Control Memory Address =0FF9E000?
Client IP Address      =193.146.252.58?
Server IP Address      =193.146.252.30?
Subnet IP Address Mask =255.255.255.128?
Broadcast IP Address   =193.146.252.127?
Gateway IP Address     =193.146.252.1?
Boot File Name ("NULL" for None)     =zImage.prep?
Argument File Name ("NULL" for None) =?
Boot File Load Address         =001F0000?
Boot File Execution Address    =001F0000?
Boot File Execution Delay      =00000000?
Boot File Length               =00000000?
Boot File Byte Offset          =00000000?
BOOTP/RARP Request Retry       =10?
TFTP/ARP Request Retry         =10?
Trace Character Buffer Address =00000000?
BOOTP/RARP Request Control: Always/When-Needed (A/W)=W?
BOOTP/RARP Reply Update Control: Yes/No (Y/N)       =Y?

Update Non-Volatile RAM (Y/N)? Y
```

Para que los cambios sean permanentes es necesario modificar la ROM. Esto se consigue contestando afirmativamente a la última pregunta que se acaba de mostrar. Si no se han realizado cambios no aparece la última pregunta.

Una vez hecha la modificación es necesario reiniciar la tarjeta pulsando *RST* en el panel frontal.

En la configuración definitiva optamos por eliminar los parámetros estáticos y los sustituimos por una petición BOOTP, ya que esta última opción es más flexible.

```
PPC4-Bug>NIOT
Controller LUN =00?
Device LUN     =00?
Node Control Memory Address =0FF9E000?
Client IP Address      =193.146.252.58?0.0.0.0
Server IP Address      =193.146.252.30?0.0.0.0
Subnet IP Address Mask =255.255.255.128?0.0.0.0
Broadcast IP Address   =193.146.252.127?0.0.0.0
Gateway IP Address     =193.146.252.1?0.0.0.0
Boot File Name ("NULL" for None)     =zImage.prep?NULL
Argument File Name ("NULL" for None) =?
Boot File Load Address         =001F0000?
Boot File Execution Address    =001F0000?
Boot File Execution Delay      =00000000?
Boot File Length               =00000000?
Boot File Byte Offset          =00000000?
BOOTP/RARP Request Retry       =10?
TFTP/ARP Request Retry         =10?
Trace Character Buffer Address =00000000?
BOOTP/RARP Request Control: Always/When-Needed (A/W)=W?A
BOOTP/RARP Reply Update Control: Yes/No (Y/N)       =Y?

Update Non-Volatile RAM (Y/N)? Y
```

Como se puede observar se ponen todos los campos a “NULL” o ‘0’ y se fuerza que se realice una petición BOOTP/RARP siempre que se inicie el procesador MVME.

3. Servidores de BOOTP, TFTP

3.1. BOOTP

BOOTP es un protocolo que permite asignar a un ordenador una dirección IP a partir de la dirección MAC de su tarjeta. Todas las tarjetas de red tienen un número MAC que las identifica unívocamente y BOOTP hace uso de esa propiedad para realizar esta asignación. BOOTP propociona información adicional para el arranque del sistema.

Los ordenadores que no disponen de disco duro habitualmente tienen una PROM con firmware del fabricante que permite establecer de donde deseamos arrancar el ordenador. Habitualmente se puede arrancar desde CD, disco duro o desde la red. Si el firmware es muy sencillo estas opciones deberían bastar. En el caso de arranque por red, el ordenador hace una petición general utilizando el protocolo BOOTP solicitando a la red local que le proporcione una dirección IP y, si este fuera el caso, otro tipo de información adicional como por ejemplo la dirección de la

puerta de entrada, la máscara de la red o el nombre del archivo que debe envíar al cliente que lo solicita para arrancar el sistema operativo.

En Debian se puede montar un servidor BOOTP empleando cualquiera de los siguiente paquetes:

- bootp - server for the bootp protocol with DHCP support
- bootparamd - Boot parameter server.
- dhcp - DHCP server for automatic IP address assignment
- dhcp3-server - DHCP server for automatic IP address assignment
- dnsmasq - A small caching DNS proxy and DHCP server.
- udhcpd - very small DHCP server

En el Centro Astronómico de Yebes ya hay montado un servidor DHCP3 para la asignación dinámica de direcciones IP y por tanto se puede utilizar como servidor BOOTP. Sin embargo en lugar de emplear el servidor DHCP3 hemos preferido instalar un servidor BOOTP (paquete bootp) en el mismo ordenador que sirve los discos del sistema empleando NFS y que sirve el fichero de arranque empleando TFTP ya que se trata de una solución más compacta y autocontenida. El servidor BOOTP se inicia desde *inetd*, cuando el sistema recibe una petición en el puerto 67. El demonio *bootpd* se pone en marcha y lee el archivo de configuración de `/etc/bootptab`. Este archivo contiene las opciones de configuración adecuadas para cada ordenador que arranque por la red. En este caso se muestra el perfil de configuración de `pebbles`.

```
pebbles:\
        :bf=zImage.pplus\
        :ht=ether\
        :ha=0001AF09CF30\
        :ip=193.146.252.58\
        :dn=oan.es\
        :ds=193.146.252.16\
        :sm=255.255.255.128\
        :gw=193.146.252.1\
        :rp=/var/lib/nfs/pebbles-po\
        :hn:bs=auto\
        :hd=/:td=/var/lib/tftpboot
```

- Cada uno de los parámetros se separa del anterior con dos puntos ":", y se utiliza una barra invertida como continuación de línea.
- La opción `bf` indica el archivo de arranque.
- La opción `ht` indica que el arranque se hace por red
- La opción `ha` identifica el ordenador utilizando la dirección hardware (MAC) de la tarjeta.

- La opción `ip` establece una dirección IP para el equipo con la dirección MAC anterior.
- La opción `dn` indica el nombre del dominio.
- La opción `ds` indica el servidor de nombres de la red local.
- La opción `sm` indica la máscara de la subred.
- La opción `gw` indica la puerta de enlace (gateway).
- La opción `rp` indica la ruta del directorio raíz servido por NFS.
- La opción `bs` establece el tamaño en bloques de 512 bytes del archivo de arranque.
- La opción `hd` indica el directorio en el servidor donde se encuentra el archivo de arranque.
- Finalmente la opción `td` indica el directorio TFTP.

3.2. TFTP

Tanto si el VME realiza una petición BOOTP, como si se establece estáticamente la dirección IP de la tarjeta, la máscara de la red, la dirección de la puerta de acceso, y la dirección IP del servidor, es necesario disponer de un servidor TFTP que entregue archivos cuando se produce una petición. En Debian hay tres paquetes que permiten montar un servidor TFTP:

- *atftpd* - Advanced TFTP server
- *tftpd* - Internet trivial file transfer protocol server.
- *tftpd-hpa* - HPA's tftp server

Como primera opción se instaló el paquete `atftpd` por tratarse aparentemente del más avanzado de todos. Sin embargo este paquete causó numerosos errores que exigieron una monitorización detallada de la red que se describe en la siguente sección. El servidor no respondía adecuadamente a las peticiones que le solicitaba el cliente del MVME y este repetía periódicamente su petición.

El problema se solucionó instalando el paquete tftpd-hpa. El demonio se puede iniciar desde *inetd* o independientemente. En nuestro caso se eligió la opción de arranque independiente. La configuración se fija en el archivo `/etc/default/tftpd-hpa`:

```
#Defaults for tftpd-hpa
RUN_DAEMON="yes"
OPTIONS="-l -s /var/lib/tftpboot
```

Es imprescindible establecer explicitamente la ruta del directorio donde se encuentran los archivos que se van a entregar a los clientes. El directorio y los archivos deben tener permiso de lectura para todos los usuarios y el directorio además debe tener permiso de ejecución, ya que permite entrar a ese directorio. El demonio se inicia, detiene y reinicia empleando instrucciones del tipo:

```
/etc/init.d/tftpd-hpa start|stop|restart
```

donde las opciones que se han indicado son excluyentes entre sí. Es decir se debe elegir o `start`, o `restart` o `stop`. Los archivos que el servidor puede entregar se situaron en el directorio `/var/lib/tftpboot`.

4. Espionaje de paquetes en la red

Para poder descubrir los problemas que se producían con el paquete *atftpd* fue necesario espiar los paquetes TCP que circulaban por la red entre el servidor TFTPD y el cliente MVME. Esto se consigue empleando la aplicación *tcpdump*, proporcionada por el paquete del mismo nombre. Dado que la red soporta un tráfico considerable de paquetes es necesario filtrarlo para aceptar sólo la comunicación entre dos ordenadores. En la práctica esto se consigue empleando la siguiente instrucción en el servidor TFTPD:

```
tcdump host pebbles.oan.es -w ficheroDeSalida
```

donde `pebbles.oan.es` es el ordenador con el que se comunica el servidor y la opción `-w ficheroDeSalida` permite guardar los paquetes en bruto, en un archivo que después se puede analizar empleando una utilidad de análisis de paquetes. En nuestro caso la utilidad se denomina *netdude*.

netdude permite abrir un fichero con los paquetes registrados. Al abrir dicho fichero en el recuadro superior de la ventana de la aplicación se muestra un resumen de la comunicación. Para obtener información más detallada es necesario situarse sobre una de las líneas. Al situarse sobre dicha línea, esta queda resaltada y en el recuadro inferior aparecen 4 solapas que muestran la cabecera Ethernet, IP, la cabecera UDP y los datos brutos del paquete. Seleccionando sobre cada una de ellas se muestra dicha información.

Utilizando esta aplicación descubrimos que el servidor TFTPD implementado por el paquete Debian *atftpd* respondía con un paquete de 2 octetos (2 bytes) a la petición de entrega del archivo con el núcleo de Linux. En realidad la respuesta debería contener aproximadamente 512 octetos. Al no recibir el cliente de TFTP la respuesta adecuada, este volvía a realizar otra petición solicitando de nuevo el archivo. El servidor respondía en un puerto diferente (un número superior al anterior) con otro paquete que sólo contenía 2 octetos, y esta situación se repetía ad infinitum.

5. Cliente de BOOTP y TFTP. El punto de vista del MVME2434

Si el VME está configurado con los parámetros estáticos de la red (archivo a descargar, número IP, máscara de red, puerta de enlace, servidor de nombres), una vez que el cliente ha descargado el archivo de arranque este se recibe en el MVME y se situa en una zona de memoria desde donde se ejecuta la primera instrucción. A continuación mostramos el tipo de mensaje que muestra el MVMVE al recibir un archivo e intentar ejecutar la primera instrucción que contiene:

```
Device Name: /pci@80000000/pci1011,19@e,0:0,0
```

```
Loading: zImage.prep

Client IP Address      = 193.146.252.58
Server IP Address      = 193.146.252.30
Gateway IP Address     = 193.146.252.1
Subnet IP Address Mask = 255.255.255.128
Boot File Name         = zImage.prep
Argument File Name     =

Network Boot File load in progress... To abort hit <BREAK>

Bytes Received =&870906, Bytes Loaded =&870906
Bytes/Second   =&435453, Elapsed Time =2 Second(s)

Residual-Data Located at: $0FF88000
loaded at:     00005400 000E1FD0
relocated to:  00800000 008DCBD0
board data at: 0FF88000 0FF8EA0C
relocated to:  008D3134 008D9B40
zimage at:     00806B90 008D2ABD
avail ram:     00400000 00800000

Linux/PPC load
```

Como se puede observar se muestra el número de bytes recibidos, el tiempo transcurrido, y después la zona de memoria donde se almacena temporalmente, la zona donde se mueve definitivamente y la dirección de memoria de la primera instrucción, así como la memoria RAM disponible.

Si el archivo descargado contiene una imagen ejecutable, el microprocesador intentará ejecutar la primera instrucción que encuentra. Si la imagen es la adecuada debe aparecer un mensaje del tipo:

```
Linux/PPC load:
```

Si por el contrario se han desactivado las opciones estáticas de red, el VME realizará una petición BOOTP a la red local y recibirá respuesta del servidor BOOTP. La primera ocasión en la que ocurre esto, el cliente tarda en ser respondido. En las siguientes ocasiones, aunque se haya configurado la opción en PPC4 de que realice una nueva petición con cada arranque, el MVME conserva en memoria la información del servidor BOOTP, es decir su dirección IP y el archivo que este sirve. Los mensajes se muestran a continuación:

```
Copyright Motorola Inc. 1988 - 1999, All Rights Reserved

PPC4 Debugger/Diagnostics Release Version 1.2 - 02/15/99 RM01
COLD Start

Local Memory Found =10000000 (&268435456)
```

```
MPU Clock Speed =350Mhz

BUS Clock Speed =100Mhz

Reset Vector Location  : ROM Bank B
Mezzanine Configuration: Single-MPU
Current 60X-Bus Master : MPU0
Idle MPU(s)            : NONE

Initializing System Memory (DRAM)...                                    S
led (ECC-Memory Detected)
L2Cache:       1024KB, 140Mhz


SelfTest/Boots about to Begin... Press <BREAK> at anytime to Abort ALL

NetBoot about to Begin... Press <ESC> to Bypass, <SPC> to Continue

Network Booting from: DEC21143, Controller 0, Device 0
Device Name: /pci@80000000/pci1011,19@e,0:0,0
Loading: //zImage.pplus

Client IP Address      = 193.146.252.58
Server IP Address      = 193.146.252.30
Gateway IP Address     = 0.0.0.0
Subnet IP Address Mask = 0.0.0.0
Boot File Name         = //zImage.pplus
Argument File Name     =

Network Boot File load in progress... To abort hit <BREAK>

Bytes Received =&1002041, Bytes Loaded =&1002041
Bytes/Second   =&501020, Elapsed Time =2 Second(s)

Residual-Data Located at: $0FF88000
loaded at:     00005400 000FB5DC
relocated to:  00800000 008F61DC
zimage at:     0080594B 008F24A3
avail ram:     00400000 00800000

Linux/PPC load:
```

6. Núcleo Linux para PPC PReP

Como acabamos de ver, en el servidor TFTPD necesitamos situar una imagen de Linux para PowerPC, en particular para arquitecturas PReP. Debian proporciona estas imágenes pero ninguna de ellas condujo a una instalación satisfactoria. Se probaron las imágenes de Potato, Woody y Sarge (Octubre de 2004) y los resultados fueron en todos los casos decepcionantes:

- La imagen de la versión Potato no arranca.
- La imagen de la versión Woody si arranca pero cuando el control pasa a la consola a través del puerto serie la velocidad se ralentiza hasta extremos que hace inusable la instalación.
- La imagen 2.6 de Sarge muestra un mensaje del tipo “Now booting the kernel” pero no arranca.

Finalmente Jan Goersmann (comunicación privada) nos proporcionó una imagen generada por él mismo utilizando compilación cruzada que empleamos para iniciar todo el proceso de instalación. La compilación cruzada se describe en [**?**].

7. Servidor NFS de /root.

7.1. Parámetros de inicio del cliente

Durante el proceso de inicio del núcleo de Linux, este se detiene durante dos segundos permitiendo que el usuario introduzca parámetros de inicio. Una vez que se comienza a escribir el proceso se detiene completamente hasta que se pulsa Intro. Una lista de los parámetros de inicio se puede encontrar en [5]. Si deseamos que Linux monte el sistema de archivos de root empleando NFS es necesario introducir la siguiente combinación de parámetros:

```
Linux/PPC load: root=/dev/nfs ip=193.146.252.58:193.146.252.30 nfsroot=/var/lib/nfs/pebb
```

El primer parámetro indica al núcleo de Linux que el dispositivo donde se montará root es `/dev/nfs`, es decir se empleará NFS. La siguiente opción indica el número IP del cliente, es decir de la propia tarjeta MVME, seguida, y separada por “:”, de la dirección IP del servidor NFS. La última opción indica la ruta absoluta al directorio NFS que se exporta desde el servidor.

Todo el contenido del parámetro que se pasa al núcleo de Linux durante su arranque se puede almacenar en una variable, por ejemplo `bootargs` empleando el comando **GEVDIT** de PPCBug:

```
PPC4-Bug> GEVEDIT bootargs
bootargs=root=/dev/nfs ip=193.146.252.58:193.146.252.30 nfsroot=/var/lib/nfs/pebbles-sa

Update Global Environment Area of NVRAM (Y/N)? Y
```

Una vez configurado el parámetro `bootargs` no es necesario introducir ningún parámetro de arranque al iniciar Linux.

Si la tarjeta MVME se ha iniciado haciendo una petición BOOTP y este se encuentra correctamente configurado los siguientes parámetros deberían bastar para que el sistema arranque correctamente:

```
Linux/PPC load: ip=on root=/dev/nfs console=/ttyS0
```

7.2. Configuración del servidor NFS root

Debian 3.1 proporciona dos paquetes para implementar un servidor NFS:

- nfs-user-server - User space NFS server
- nfs-kernel-server - Kernel NFS server support

El paquete *nfs-user-server* proporciona un servidor NFS que genera bastantes problemas durante la instalación de los paquetes utilizando NFS root. Se obtienen mensajes de este tipo:

```
apt-get install passwd
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be upgraded:
  passwd
  1 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
  Need to get 0B/542kB of archives.
  After unpacking 844kB of additional disk space will be used.
  debconf: delaying package configuration, since apt-utils is not installed
  (Reading database ... 11691 files and directories currently installed.)
  Preparing to replace passwd 20000902-12
  (using .../passwd_1%3a4.0.3-30.1_powerpc.deb) ...
  Unpacking replacement passwd ...
  dpkg: error
  processing /var/cache/apt/archives/passwd_1%3a4.0.3-30.1_powerpc.deb
  (--unpack):
   unable to create ‘./usr/share/man/man8/grpunconv.8.gz’: Too many levels of
   symbolic links
   dpkg-deb: subprocess paste killed by signal (Broken pipe)
   Errors were encountered while processing:
    /var/cache/apt/archives/passwd_1%3a4.0.3-30.1_powerpc.deb
```

Este tipo de errores impide la actualización del sistema o la instalación de paquetes nuevos. Para evitarlo es necesario instalar el paquete *nfs-kernel-server*:

```
apt-get install nfs-kernel-server
```

La configuración se hace creando la zona donde residirá el contenido del disco:

```
mkdir /var/lib/nfs
mkdir /var/lib/nfs/pebbles
```

El subdirectorio `/var/lib/nfs/pebbles` se ha preparado exclusivamente para alojar el disco de *pebbles*. Si existiera otro VME sin disco se podría crear otro subdirectorio en `/var/lib/nfs` para él.

Después es necesario modificar el archivo `/etc/exports` para que el servidor NFS autorice a montar externamente el directorio a un ordenador en particular:

```
/var/lib/nfs            193.146.252.58(rw,no_root_squash,no_all_squash)
```

El número IP indica al servidor NFS que sólo se admite el montaje desde un ordenador que tenga asignada dicha dirección IP. Las opciones entre paréntesis indican que el sistema es de escritura/lectura (`rw`), permite que los usuarios root en ordenadores clientes tengan acceso a root en el servidor (`no_root_squash`), opción necesaria para ordenadores cliente sin disco como es nuestro caso y también se permite que los usuarios diferentes de root se consideren como tales (`no_all_squash`) y no se asocien a nobody.

Una vez creada el área descargamos en ella el contenido de `base2_2.tgz` de la versión *potato* de Debian. La secuencia de comandos sería:

```
cd /var/lib/nfs/pebbles
wget ftp.de.debian.org/debian-archive/dists/potato/main/disks-powerpc/current/base2_2.tg
tar xzvf base2_2.tgz
```

Una vez descomprimido el contenido de *potato* podemos actualizar el sistema hasta usar la versión de Debian como se describe en la siguiente sección.

8. Instalación de Debian Sarge en el MVME2434

Cuando el sistema se inicia totalmente la primera operación a realizar es configurar adecuadamente la red, añadiendo la ruta de salida. Esto se consigue empleando la siguiente instrucción:

```
route add default gw 193.146.252.1 netmask 0.0.0.0
```

Esta operación no será necesario repetirla después porque la configuración queda guardada para el futuro.

Una vez que el acceso por la red está operativo es necesario emprender la actualización a otra versión. Para ello es necesario modificar `/etc/apt/sources.list` e incluir las rutas para utilizar Woody (Debian 3.0). La actualización se hace siguiendo el procedimiento estándar. Una vez que Woody está instalado se puede volver a modificar el archivo `/etc/apt/sources.list`.

Una vez completada la actualización se recompiló el núcleo empleando las utilidades de Debian:

```
apt-get install kernel-package
apt-get install initrd-tools
apt-get install libc6-dev
cd /usr/src/linux
make config
make-kpkg clean
make-kpkg --revision=prep1 kernel_image
cd ..
dpkg --install kernel-image-2.4.27_prep1_powerpc.deb
```

La configuración del núcleo se hizo con los valores predeterminados.

Este procedimiento no sería necesario si la instalación de Sarge se hiciese de modo correcto desde el principio. Si este fuera el caso deberíamos situar el núcleo de Sarge en el directorio desde donde TFTPD sirve los archivos.

9. Configuración final

9.1. Montaje del disco del sistema

El disco raíz del sistema se monta a través de la red, por tanto es necesario especificarlo en el archivo `/etc/fstab` del siguiente modo:

```
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
193.146.252.30:/var/lib/nfs/pebbles0     /       nfs     defaults        0 0
proc            /proc           proc    defaults        0       0
```

9.2. Configuración para operar con el puerto serie de la tarjeta de Motorola

En el rack del MVME se alojan varios equipos, entre los que se encuentra un sintetizador que permite desplazar la frecuencia central del autocorrelador. La frecuencia se sintoniza empleando un conector serie a través del cual se monitoriza y lee la frecuencia. El objetivo es utilizar el conector serie de depuración de la tarjeta VME para comandar y leer el sintetizador.

Dado que el MVME es un ordenador sin disco, sin teclado ni pantalla la consola es el puerto serie. En el puerto serie se vuelca toda la información del sistema durante el inicio y el apagado del sistema. Por tanto es necesario evitar que se inicie el programa `getty` que implementa consolas virtuales. En circunstancias normales la consola debería ser el dispositivo `/dev/ttyS0` pero `getty` bloquea el acceso al puerto serie para el resto de las aplicaciones por lo que debemos evitar lanzar ningún `getty`. Para ello es necesario comentar las entradas siguientes en el archivo `/etc/inittab`

```
#  <id>:<runlevels>:<action>:<process>
#1:2345:respawn:/sbin/getty 38400 tty1
#2:23:respawn:/sbin/getty 38400 tty2
#3:23:respawn:/sbin/getty 38400 tty3
#4:23:respawn:/sbin/getty 38400 tty4
#5:23:respawn:/sbin/getty 38400 tty5
#6:23:respawn:/sbin/getty 38400 tty6
```

Una vez que el VME arrancó se probó el tráfico a través del conector serie

10. Instalación de un nucleo 2.6.8

Para poder controlar el autocorrelador es necesario disponer de los módulos *universe* para el bus VME y *pebbles* para la tarjeta autocorreladora. Estos módulos han sido escritos por Gabriel Paubert de IRAM, que actualizó los módulos para la serie 2.6. Por tanto se instaló un núcleo 2.6.8 en el procesador VME. A continuación se resume el archivo de configuración empleado para generar el núcleo. Se han eliminado del listado aquellas opciones que no están configuradas:

```
#
# Automatically generated make config: don't edit
```

```
#
CONFIG_MMU=y
CONFIG_RWSEM_XCHGADD_ALGORITHM=y
CONFIG_HAVE_DEC_LOCK=y
CONFIG_PPC=y
CONFIG_PPC32=y
CONFIG_GENERIC_NVRAM=y

#
# Code maturity level options
#
CONFIG_EXPERIMENTAL=y
CONFIG_CLEAN_COMPILE=y
CONFIG_BROKEN_ON_SMP=y

#
# General setup
#
CONFIG_SWAP=y
CONFIG_SYSVIPC=y
CONFIG_POSIX_MQUEUE=y
CONFIG_SYSCTL=y
CONFIG_LOG_BUF_SHIFT=14
CONFIG_IKCONFIG=y
CONFIG_IKCONFIG_PROC=y
CONFIG_KALLSYMS=y
CONFIG_FUTEX=y
CONFIG_EPOLL=y
CONFIG_IOSCHED_NOOP=y
CONFIG_IOSCHED_AS=y
CONFIG_IOSCHED_DEADLINE=y
CONFIG_IOSCHED_CFQ=y

#
# Loadable module support
#
CONFIG_MODULES=y
CONFIG_MODULE_UNLOAD=y
CONFIG_OBSOLETE_MODPARM=y

#
# Processor
#
CONFIG_6xx=y
CONFIG_PPC_GEN550=y
CONFIG_PPC_STD_MMU=y
```

```
#
# Platform options
#
CONFIG_PPLUS=y
CONFIG_KERNEL_ELF=y
CONFIG_BINFMT_ELF=y
CONFIG_CMDLINE_BOOL=y
CONFIG_CMDLINE="ip=on root=/dev/nfs console=ttyS0"

#
# Bus options
#
CONFIG_GENERIC_ISA_DMA=y
CONFIG_PCI=y
CONFIG_PCI_DOMAINS=y
CONFIG_PCI_NAMES=y

#
# Advanced setup
#
# CONFIG_ADVANCED_OPTIONS is not set

#
# Default settings for advanced configuration options are used
#
CONFIG_HIGHMEM_START=0xfe000000
CONFIG_LOWMEM_SIZE=0x30000000
CONFIG_KERNEL_START=0xc0000000
CONFIG_TASK_SIZE=0x80000000
CONFIG_BOOT_LOAD=0x00800000

#
# Device Drivers
#

#
# Generic Driver Options
#
CONFIG_STANDALONE=y
CONFIG_PREVENT_FIRMWARE_BUILD=y

#
# Networking support
#
CONFIG_NET=y

#
```

```
# Networking options
#
CONFIG_PACKET=y
CONFIG_UNIX=y
CONFIG_INET=y
CONFIG_IP_MULTICAST=y
CONFIG_IP_PNP=y
CONFIG_IP_PNP_BOOTP=y
CONFIG_SYN_COOKIES=y

#
# SCTP Configuration (EXPERIMENTAL)
#
CONFIG_IP_SCTP=m
CONFIG_SCTP_HMAC_MD5=y
#
# Network testing
#
CONFIG_NETDEVICES=y

#
# Ethernet (10 or 100Mbit)
#
CONFIG_NET_ETHERNET=y
CONFIG_MII=y

#
# Tulip family network device support
#
CONFIG_NET_TULIP=y
CONFIG_TULIP=y

#
# Input device support
#
CONFIG_INPUT=y

#
# Userland interfaces
#
CONFIG_INPUT_MOUSEDEV=y
CONFIG_INPUT_MOUSEDEV_SCREEN_X=1024
CONFIG_INPUT_MOUSEDEV_SCREEN_Y=768

#
# Input I/O drivers
#
```

```
# CONFIG_GAMEPORT is not set
CONFIG_SOUND_GAMEPORT=y

#
# Character devices
#
CONFIG_VT=y
CONFIG_VT_CONSOLE=y
CONFIG_HW_CONSOLE=y

#
# Serial drivers
#
CONFIG_SERIAL_8250=y
CONFIG_SERIAL_8250_CONSOLE=y
CONFIG_SERIAL_8250_NR_UARTS=4
# CONFIG_SERIAL_8250_EXTENDED is not set

#
# Non-8250 serial port support
#
CONFIG_SERIAL_CORE=y
CONFIG_SERIAL_CORE_CONSOLE=y
CONFIG_UNIX98_PTYS=y

# Console display driver support
#
CONFIG_DUMMY_CONSOLE=y

#
# Pseudo filesystems
#
CONFIG_PROC_FS=y
CONFIG_PROC_KCORE=y
CONFIG_SYSFS=y
CONFIG_TMPFS=y
CONFIG_RAMFS=y

# Network File Systems
#
CONFIG_NFS_FS=y
CONFIG_NFS_V3=y
CONFIG_ROOT_NFS=y
CONFIG_LOCKD=y
CONFIG_LOCKD_V4=y
CONFIG_SUNRPC=y
```

```
#
# Partition Types
#
CONFIG_PARTITION_ADVANCED=y

#
# Library routines
#
CONFIG_CRC32=y

#
# Kernel hacking
#
CONFIG_SERIAL_TEXT_DEBUG=y

#
# Cryptographic options
#
CONFIG_CRYPTO=y
CONFIG_CRYPTO_HMAC=y
CONFIG_CRYPTO_MD5=y
```

El parámetro `CONFIG_CMDLINE="ip=on root=/dev/nfs console=ttyS"` permite obviar la entrada de opciones y el empleo de `GEVDIT` durante el arranque del núcleo. Como se puede observar no es necesario especificar ni la ruta del NFS root ni la dirección del servidor y del cliente, ya que el servidor BOOTP ya lo indica.
El parámetro `CONFIG_PPLUS=y` indica que la plataforma es una PowerPC Plus.
La ejecución de make genera varios núcleos en el directorio `/usr/src/linux/arch/ppc/boot/images`. De ellos la imágen que se utiliza es la `zImage.pplus`.

Durante el inicio se obtienen los siguientes mensajes:

```
Copyright Motorola Inc. 1988 - 1999, All Rights Reserved

PPC4 Debugger/Diagnostics Release Version 1.2 - 02/15/99 RM01
COLD Start

Local Memory Found =10000000 (&268435456)

MPU Clock Speed =350Mhz

BUS Clock Speed =100Mhz

Reset Vector Location  : ROM Bank B
Mezzanine Configuration: Single-MPU
Current 60X-Bus Master : MPU0
Idle MPU(s)            : NONE
```

```
Initializing System Memory (DRAM)...                                  System M
led (ECC-Memory Detected)
L2Cache:       1024KB, 140Mhz


SelfTest/Boots about to Begin... Press <BREAK> at anytime to Abort ALL

NetBoot about to Begin... Press <ESC> to Bypass, <SPC> to Continue

Network Booting from: DEC21143, Controller 0, Device 0
Device Name: /pci@80000000/pci1011,19@e,0:0,0
Loading: //zImage.pplus

Client IP Address      = 193.146.252.58
Server IP Address      = 193.146.252.30
Gateway IP Address     = 0.0.0.0
Subnet IP Address Mask = 0.0.0.0
Boot File Name         = //zImage.pplus
Argument File Name     =

Network Boot File load in progress... To abort hit <BREAK>

Bytes Received =&1002041, Bytes Loaded =&1002041
Bytes/Second   =&501020, Elapsed Time =2 Second(s)

Residual-Data Located at: $0FF88000
loaded at:     00005400 000FB5DC
relocated to:  00800000 008F61DC
zimage at:     0080594B 008F24A3
avail ram:     00400000 00800000

Linux/PPC load: ip=on root=/dev/nfs console=ttyS0
Uncompressing Linux...done.
Now booting the kernel
id mach(): done
MMU:enter
pplus_find_end_of_memory
MMU:hw init
hash:enter
hash:find piece
hash:patch
hash:done
MMU:mapin
MMU:setio
MMU:exit
setup_arch: enter
setup_arch: bootmem
```

```
pplus_setup_arch: enter
pplus_setup_arch: find_bridges
pplus_setup_arch: set_board_type
pplus_setup_arch: exit
arch: exit
init_irq: enter
openpic: enter
openpic: timer
openpic: external
openpic: spurious
openpic: exit
init_irq: exit
Total memory = 256MB; using 512kB for hash table (at c0280000)
Linux version 2.6.8 (root@pebbles) (gcc version 3.3.5 (Debian 1:3.3.5-8)) #11 Th
Motorola PowerPlus Platform
Port by MontaVista Software, Inc. (source@mvista.com)
Built 1 zonelists
Kernel command line: ip=on root=/dev/nfs console=ttyS0
OpenPIC Version 1.3 (2 CPUs and 16 IRQ sources) at fdfc0000
OpenPIC timer frequency is 12.500698 MHz
PID hash table entries: 2048 (order 11: 16384 bytes)
time_init: decrementer frequency = 25.000413 MHz
Console: colour dummy device 80x25
Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
Memory: 256576k available (1388k kernel code, 508k data, 284k init, 0k highmem)
Calibrating delay loop... 698.36 BogoMIPS
Mount-cache hash table entries: 512 (order: 0, 4096 bytes)
NET: Registered protocol family 16

PCI: Probing PCI hardware
Setting PCI interrupts for a "MVME 2300"
Initializing Cryptographic API
Serial: 8250/16550 driver $Revision: 1.90 $ 8 ports, IRQ sharing disabled
ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
Linux Tulip driver version 1.1.13 (May 11, 2002)
tulip0:  EEPROM default media type Autosense.
tulip0:  Index #0 - Media MII (#11) described by a 21142 MII PHY (3) block.
tulip0:  MII transceiver #8 config 1000 status 7809 advertising 01e1.
eth0: Digital DS21143 Tulip rev 65 at 0xfffdf80, 00:01:AF:09:CF:30, IRQ 18.
mice: PS/2 mouse device common for all mice
NET: Registered protocol family 2
IP: routing cache hash table of 2048 buckets, 16Kbytes
TCP: Hash tables configured (established 16384 bind 32768)
NET: Registered protocol family 1
NET: Registered protocol family 17
Sending BOOTP requests .<6>eth0: Setting full-duplex based on MII#8 link partner
```

```
. OK
IP-Config: Got BOOTP answer from 193.146.252.30, my address is 193.146.252.58
IP-Config: Complete:
      device=eth0, addr=193.146.252.58, mask=255.255.255.128, gw=193.146.252.1,
     host=pebbles, domain=oan.es, nis-domain=(none),
     bootserver=193.146.252.30, rootserver=193.146.252.30, rootpath=/var/lib/nfs
Looking up port of RPC 100003/2 on 193.146.252.30
Looking up port of RPC 100005/1 on 193.146.252.30
VFS: Mounted root (nfs filesystem) readonly.
Freeing unused kernel memory: 284k init
INIT: version 2.86 booting
Activating swap.
Cannot access the Hardware Clock via any known method.
Use the --debug option to see the details of our search for an access method.
System time was Mon Apr 18 10:07:33 UTC 2005.
Setting the System Clock using the Hardware Clock as reference...
Cannot access the Hardware Clock via any known method.
Use the --debug option to see the details of our search for an access method.
System Clock set. System local time is now Mon Apr 18 10:07:33 UTC 2005.
Cleaning up ifupdown...done.
Calculating module dependencies... done.
Loading modules...
All modules loaded.
Checking all file systems...
fsck 1.35 (28-Feb-2004)
Setting kernel variables ...
... done.
Mounting local filesystems...
Cleaning /tmp /var/run /var/lock.
Running 0dns-down to make sure resolv.conf is ok...done.
Setting up networking.../dev/shm/network/...done.
Setting up IP spoofing protection: rp_filter.
Configuring network interfaces...SIOCADDRT: File exists
Failed to bring up eth0.
done.
Mounting remote filesystems...
Loading the saved-state of the serial devices...
/dev/ttyS0 at 0x03f8 (irq = 4) is a 16550A

Setting the System Clock using the Hardware Clock as reference...
Cannot access the Hardware Clock via any known method.
Use the --debug option to see the details of our search for an access method.
System Clock set. Local time: Mon Apr 18 10:07:35 UTC 2005

Initializing random number generator...done.
Setting up X server socket directory /tmp/.X11-unix...done.
Setting up ICE socket directory /tmp/.ICE-unix...done.
```

```
INIT: Entering runlevel: 2
Starting system log daemon: syslogd.
Starting kernel log daemon: klogd.
Starting PCMCIA services: module directory /lib/modules/2.6.8/pcmcia not found.
Starting internet superserver: inetd.
Starting OpenBSD Secure Shell server: sshd.
Starting NTP server: ntpd.
INIT: no more processes left in this runlevel
```

En otro informe se describen el módulo “universe”, el módulo “pebbles” y el demonio “pebblesd” que se comunica con el módulo anterior. En dicho informe se describen los cambios realizados en el demonio para poder controlar los distintos modos de operación del autocorrelador. También se describe el controlador escrito en espacio de usuario para utilizar la tarjeta PCI Mezzazine con 16 puertos serie.

Referencias

[1] http://mcg.motorola.com/us/ds/pdf/ds0058.pdf

[2] https://mcg.motorola.com/wp/index.cfm?pagetypeID=59&Source=21&dtype=3&PageSrc=v2400ai

[3] http://www.zytrax.com/tech/layer_1/cables/tech_rs232.htm

[4] https://mcg.motorola.com/cfm/templates/documentation.cfm?PageID=959&ProductID=9&PageTypeID=

[5] http://www.ibiblio.org/mdw/HOWTO/BootPrompt-HOWTO.html