Cuaderno de Bitácora: Anotaciones y Documentación.

Apunte 1.

- **Fase de documentación (1-15 de Octubre de 2002):**
  - Se ha estudiado proyectos similares para proporcionar punto de partida y visión de conjunto, como son: Autopilot, Cal-Poly Flight Simulator y Marvin.
  - Se ha elegido como fabricante del microcontrolador a ATMEL. Se ha hecho acopio de documentación del mismo optándose por el kit de desarrollo AT91EB40A. Documentación asociada: características eléctricas del micro de la placa, SW de programación cruzada (Angel), diseño del micro y emsamblador.
  - Estudio de diferentes opciones de S.O. para la placa objetivo: uClinux, eCos, uCos II, Angel (proporcionado por ATMEL).
  - Estudio de entornos de programación sobre plataformas de Intel:
    - Bajo Windows: Matlab: Simulink, RTW (Real Time Workshop), Aerospace Blockset, Fuzzy Control y compiladores.
    - Bajo Windows y Linux: WxWorks.
  - La mayor parte de la documentación se encuentra en el CD1 y en el ordenador del departamento AERO1.

    (a) Fase de prueba de distribución uClinux (8-12 de Octubre de 2002):
  - El paquete de la distribución de uClinux para el chip de ATMEL se encuentra disponible en internet (véase "Targeting uClinux on the ARM7TDMI with GNU Tools"). Se bajó al portátil de Ismael Alcalá (profesor tutor) y se me proporcionó en el CD1.
  - Para probar cómo funcionaba dicho paquete se necesitaba de un ordenador configurado, pero en el departamento no se disponía de uno. Ante este problema se optó por acudir a Miguel Ángel Lupiáñez Rueda; aportó su equipo portátil con una distribución de Debian.
  - El paquete era de RedHat (.rpm) luego tuvimos que transformar sus relaciones de dependencias a las de Debian (.deb). El SW necesario se bajó de Internet desde la conexión de mi casa al portátil de Miguel Ángel.
  - Una vez transformada se descomprimió el paquete observándose que disponía con un fichero "script" que contenía todas las órdenes necesarias para compilar el kernel de uClinux y el compilador de código libre para el micro de ATMEL.
  - En un equipo PIII-500 se necesitó cerca de 40' en la compilación.
  - Hace falta configurar el kernel.

    (b) Fase de configuración del sistema (15-21 de Octubre de 2002):
  - Para el estudio y análisis posterior del kit de desarrollo se necesita usar un PC. Para ello se utiliza el P133 del departamento.
  - Para probar los distintos SW de desarrollo se ha optado por instalar W95 y Linux RedHat 6.7, cada uno en una unidad de HD del ordenador.
  - La instalación de ambos sistemas operativos se efectuó con éxito tras comprobar qué HD's estaban disponibles.
  - Se hace necesario comprobar el correcto funcionamiento de los mismos.

Incidentes a destacar: debido a que el kit de desarrollo ve retrasada su entrega frente a la fecha que indicó el fabricante, el proyecto se encuentra a la espera.

Apunte 2

- **Fase de instalación de SO:**
  - **31-10-2002**: Ya está instalado un nuevo CD-ROM en el P133 para poder instalar W95 y Linux RedHat 6.7, cada uno en una unidad de HD. Habrá que esperar al Lunes 4-11 por la mañana para acercarme por el laboratorio porque el tutor no estará por el puente del 1-Nov.
  - **31-10-2002**: Ismael me ha comentado por e-mail que el kit de desarrollo todavía no ha llegado, aunque el fabricante dijo que estaría hace dos semanas. Él llamará al distribuidor para saber más información.

Apunte 3

- **Fase de aclaración de dudas:**
  - **02-11-2002**: Le pregunté a Miguel Ángel cómo podría realizar la instalación de los dos SO. He aquí las notas de lo que me contestó:
    - Es posible que debido a que se trata de un P133, no arranque del CD-ROM si no encuentra un SO en la disquetera. En este caso, defino en la BIOS que lea del CD forzosamente al iniciar el sistema.
    - La instalación de ambos sistemas operativos posee interfaz gráfica.
    - Se realizan las particiones con la distribución RedHat 6.7:
  - Una para swapping de Linux: 256 Mb.
  - Otra para el propio Linux: 2 Gb; con el nombre "/root"
  - Otra para W95: 1,8 Gb; formateada en FAT16 o DOS.
    - Se sale de la instalación de esta distribución con Ctrl-Alt-Supr.
    - Se procede a instalar W95; éste sólo reconocerá al partición de DOS/FAT16, donde se instalará. Grabará en el sector de arranque del HD el procedimiento por el cual arrancará como único SO.
    - Una vez instalado se pasa a instalar el Linux. Éste reconocerá que existe un W95 sobre la partición de DOS. Nos saltamos el proceso de las particiones del HD, procediendo a la instalación del SO. Éste actualizará el sector de arranque del HD para posibilitar el doble arranque mediante LILO.
- **Fase de instalación:**
  - **03-11-2002**: Mañana procederé a instalar los sistemas en el P133. Hablaré con Ismael para que me proporcione ambos SO's.
  - 04-11-2002: No lo encuentro en su despacho. Le envío desde mi casa un correo preguntándole cuándo lo puedo encontrar.
  - 05-11-2002: él me contesta diciéndome que los CD's de instalación se encuentran al lado del PC, en unos cajones. Efectivamente.

Apunte 4:

- **Fase de Instalación (05-11-2002):**
  - SW disponible:

a) CD de instalación de W95.
b) Disquete de arranque W98.
c) Distribución Linux RedHat 7.2.

  - **Objetivo:** lograr un arranque dual Linux – W95 en P133 de ISA.
  - Planteamiento:

a) Realizar las particiones con el instalador de Linux.
b) Instalar W95 en partición DOS.
c) Retomar la instalación de Linux sin modificar las particiones de DOS.
d) Colocar sistema de arranque LILO en sector de arranque del disco duro primario para lograr el arranque dual.

  - Principales Problemas e Incidencias:

a) La secuencia de arranque no empieza con el CD-ROM.
b) Existe protección desde BIOS contra escritura en el sector de arranque del HD primario para evitar la entrada de virus.
c) No haber instalado nunca un sistema dual, ni una distribución Linux.

  - Soluciones planteadas:

a) Programar en la BIOS la prioridad en el arranque: CD-ROM, a:, c:.
b) Inhabilitar la protección contra escritura en la BIOS.
c) La interfaz gráfica que ofrece la distribución Linux es intuitiva.

  - Desarrollo del Proceso. Anotaciones:

Apunte 5:

1) **5/11/2002**: Colocamos el CD1 de Linux en el CD-ROM. Forzamos la secuencia de arranque de la Bios: BootUpSequence CDROM,a:,c:. El CD1 es autoinstalable.
2) Instalación de Linux en entorno gráfico: definimos las particiones de la siguiente forma:

Partición	Tamaño (Mb)	Nombre	Sistema de Archivos	Punto de Montaje	Formateo
/dev/hda1	1914	Sistema	Vfat	-	Sí
/dev/hdb1	3095	Usuario	Vfat	-	Sí
/dev/hdc1	3702	Linux	Ext3	/root	Sí
/dev/hdc2	400	Swap	Swap	-	Sí

Se usan los dos primeros HD's para Windows y el último, dividido en dos particiones, para Linux. Se aceptan las particiones así definidas y salimos del proceso de instalación de Linux: Ctrl-Alt-Supr; Linux ejecuta un proceso de cerrado ordenado y expulsa de CD. Iniciamos rearranque.

3) Colocamos el CD de W95 y el disquete de W98 (sin él no se detecta el CD de W95 puesto que no es autoinstable). W98 no detecta como unidades válidas los HD's c:\ y d:\; formateamos ambos. También detecta como CD-ROM a f:\; ejecutamos f:\instalar.exe. El proceso de instalación se bloque porque salta la alarma: "Boot Record"; bloqueándose el sistema.

   Debido a que W98 no reconoció sus particiones cómo válidas, retomaremos el proceso de definición de particiones bajo Linux.

4) Definimos las particiones siguiendo el mismo esquema que el en punto (2).
5) Iniciamos el sistema bajo W98-DOS:

   Se reconocen:

C:\	Sistema
D:\	Usuario
E:\	MS-Ramdrive
F:\	CD-ROM

   Se ha solventado el problema por la falta de reconocimiento por W98 de sus propias particiones.

   No se reconocen las particiones de Linux.

   Ejecutamos f:\instalar.exe, pero surge el mismo problema: Boot Record: bloqueo del sistema.

6) Iniciamos el sistema con el CD1 de Linux para instalar completamente el sistema operativo:
   - Mantenemos el mismo esquema de particiones, pero sin formatear las que están dedicadas a W98.
   - Grabamos el gestor de arranque del sistema LILO para que se dé el arranque dual, definiendo como S.O. por defecto a DOS. Aquí no existen problemas de "Boot Record" aunque se escriba en el sector de arranque del disco.
   - Se efectúa la instalación completa del sistema junto con X-Window.
   - Reinicio.

7) Dual Boot: Dos – Linux:
   - Si DOS:
     - Con disquete W98 y W95-CD: f:\instalar.exe → problema "Boot Record", bloqueo del sistema y no se puede instalar.
     - Sin disquete W98 no se detecta S.O. válido.
   - Si Linux: se inicia el sistema satisfactoriamente; debido a que no me manejo por él no sé cómo instalar W95.
8) Iniciamos el sistema, pero entramos forzosamente en la configuración de la BIOS. Para inhabilitar la protección contra escrituras en el sector de arranque de la partición primaria seguimos los siguientes pasos:
   - Standard Bios Setup: ok
   - Boot Sector Virus Detection: Disabled: ok
   - Save Settings and Exit: ok
9) Iniciamos el sistema con el CD1 de Linux: definimos las particiones manteniendo el esquema del punto (2). Ctrl-Alt-Supr.
10) Iniciamos el sistema eligiendo DOS: W98-DOS reconoce las particiones del punto (5). Con el CD de W95 ejecutamos f:\instalar.exe realizándose toda la instalación de W95 sobre c:\windows habiéndose solventado el problema "Boot Record".
11) Arranque de W95 por 1ª vez:
   - Existen problemas porque el sistema no encuentra el "s3mm.dll"; buscamos este archivo en el CD de W95 de forma manual desde W98-DOS, pero no existe explícitamente puesto que son todos los archivos tipo Cabinet (formato comprimido de Windows).
   - Tampoco reconoce el sistema la unidad de CD-ROM aunque haya sido instalado desde uno.
   - Tampoco reconoce la existencia de red, aunque esté una tarjeta Ethernet correctamente configurada y reconocida: posible falta de definición de DNS (dominio del nombre del servidor).
12) Iniciamos el sistema: ahora no pide ningún archivo, pero no reconoce el CD-ROM ni la red. Mensaje "Bienvenido a Windows".
13) 7/11/2002: Decido descartar el W95 e instalar W98SE:
   - Uso la distribución de W98SE que tengo en mi casa.
   - Con W98-DOS formateamos c:\ y instalamos W98SE desde CD-ROM.
14) Inicio por 1ª vez:
   - Definición de usuario:
     - Nombre: Sergio Latorre.
     - Organización: Proyecto AERO1 FCS.
   - Iniciando la base de datos de drivers: HW y Plug'n'Play...OK
   - Reinicio de W98SE. Finalizando la configuración...OK
   - Reinicio de W98SE.
     - Búsqueda de nuevo HW: SMC EtherEZ (8416).
     - Detecta: Puerto de comunicaciones, puerto de impresora, PCI VGA, S3 Tri64V y PCI (765).
   - Reinicio de W98SE.
     - Falta archivo msmp32.dll: no facilita conexiones de red.
     - Detectando HW nuevo: no hay HW nuevo.
     - Mensaje de "Bienvenido a Windows 98".
     - Forzamos la búsqueda de nuevo HW desde el Panel de Control.
   - Se detectan problemas con:
- SMC EtherEZ (8416).

- PCI Multimedia Audio Device.
- Compatibilidad para Administración avanzada de Energía.
  - Windows dice que están correctamente instalados, pero tienen problemas de funcionamiento.
  - Reinicio de W98SE: solicita ciertos archivos que aparecen en los registros del sistema, pero que no existen en el sistema de ficheros. Estos son:
    - Vnetsup.vxd; vnetbios.vxd; vredir.vxd; dfs.vxd.
    - Observando en qué fichero Cabinet de la distribución de W98SE están se usó el ordenador PII350 para extraer los ficheros requeridos del CD de W98SE mediante WinRar a un disquete.
    - Observamos en qué subdirectorios aparecen estos ficheros dentro del ordenador PII350 y los grabamos respectivamente en el P133.
  - Reinicio de W98SE en P133: ya no solicita estos ficheros: problema solventado; pero sigue sin reconocer la red (¿msnp32.dll o DNS?).

15) Salimos de W98 y reiniciamos con CD1 de Linux para proceder a su total instalación (borrando la instalación ya hecha del mismo).
    - Instalación gráfica de Linux:
      - Definición de particiones (mismo esquema que en (2)); al formatear hdc1 y hdc2 se borra el sistema Linux preinstalado.
      - LILO en el sector de arranque: DOS – Linux; DOS por defecto.
      - Inicio de instalación: copia de archivos.
    - Formateo del sistema.
    - Transferencia de la imagen del programa de instalación al hdc1.
    - Instalando paquetes.
      - Instalación completada: OK.
16) Iniciamos el sistema eligiendo Linux: se instaló en modo gráfico: posee barra de tareas llamada “Panel”.
17) **8 y 11/11/2002**:Definimos DNS en P133 observando la dirección IP del servidor que muestra el PII350 mediante “ipconfig” bajo terminal de comandos (DOS). El ordenador nos devuelve:
    - adaptador Ethernet conexión área local:
      - dirección IP: 193.147.161.70
      - máscara de subred: 255.255.255.0
      - puerta de enlace predeterminada: 193.147.161.1
      - servidor DNS primario: 193.147.161.202
      - servidor DNS alternativo: 150.214.130.15

    Si ejecutamos “ipconfig” en el P133 nos indica que no está definida: ésta puede ser una de las causas de la falta de reconocimiento de la red: no está definido cuál es el servidor. Los valores devueltos por PII350 son copiados al otro ordenador, salvo la dirección IP de la máquina, que se tomó: 193.147.161.71. Tras esto el P133 pide que se reinstalen los controladores (drivers) para la tarjeta de red SMC EtherEZ (8416). Los archivos que se van solicitando están el en CD de W98SE, pero hay que extraerlos de los ficheros Cabinet (.cab) mediante WirRar del PII350; éstos son:

Nombre fichero	Cabinet origen	Nombre fichero	Cabinet origen
Protman.dos	Net7.cab	Secur32.dll	Net7.cab
Svrapi.dll	Net7.cab	Ndishlp.sys	Net8.cab
Ndiswmi.sys	Net8.cab	Protman.exe	Net8.cab
Ndis.vxd	Net9.cab	Ndis2sup.vxd	Net9.cab
Nvetbios.vxd	Net9.cab	Smc8020m.sys	Base5.cab

Se reinició el sistema bajo W98, pero continuaba sin reconocerse el servidor.

Apunte 6:

1) **14/11/2002**: la semana pasada llegó el kit de desarrollo de ATMEL.
   - La fuente de alimentación no viene con el kit.
   - En el laboratorio existe una fuente de alimentación para DC 12V, con un conector que no se sabe cuál es su polaridad.
   - 1ª cuestión: averiguar las características adecuadas de la fuente de alimentación.
2) Debido al problema con la dirección IP, Ismael me ha proporcionado una, ésta es: **193.147.161.82**; no tendrá problemas de incompatibilidad en la red.
   - Tras el cambio de dirección IP en P133 bajo W98, manteniendo el resto de la configuración (DNS, puerta de enlace, máscara) igual que en PII350, se reinstala la tarjeta SMC EtherEZ (8416).
   - Los ficheros de instalación de la tarjeta los almacené en el P133, dentro del subdirectorio "c:\mis documentos\archivos de tarjeta Ethernet", para facilitar la instalación.
   - Reiniciamos el sistema.
3) Al arrancar W98 sigue faltando el archivo "msnp32.dll"; W98 indique que, debido a ello, varias de las facilidades de Microsoft Network no estarán habilitadas.
   - Buscamos este archivo en el PII350, pero no está (sin embargo, sí funciona la red desde él).
   - Descargamos el archivo desde www.dll-files.com. El archivo msnp32.zip contiene también un readme.txt en el que se indique que se descomprima la librería dinámica en el subdirectorio c:\windows\system.
   - Eso hice, pero, tras reiniciar nuevamente, continúa sin detectar la red. No sé cuál puede ser el fallo.
4) Con el kit de ATMEL se proporciona SW de desarrollo y un manual en CD del uso del mismo y de la propia placa.
   - Solución al problema de la fuente de alimentación:
     - 7-9 V DC, 1A, polaridad no crítica.
5) Se resolvió el problema introduciendo el CD ATMEL ARM THUMB. Posee tutoriales autolanzables bajo el formato de hipertexto.
   - Muchos de los manuales están en .pdf: se instalará Adobe Acrobat Reader 5.0, el cual está disponible en el propio CD.
6) Ismael me ha encargado que los programas en ensamblador que (del estilo "Hello World") aparecen en uno de los libros de ARM sean probados en la placa del kit bajo los diferentes entornos de desarrollo.
   - Este punto necesita:
     - Disposición de una fuente de alimentación adecuada.
     - Estudio preliminar de los diferentes entornos de desarrollo en W98 y Linux.
   - Luego: ambos sistemas deben estar perfectamente instalados.
     - Instalación del SW.
     - Estudio de los programas ejemplo y su implementación.
7) SW disponible:
   - Atmel Arm Thumb: entorno de evaluación de prestaciones, funcionalidad y consumo de potencia de la placa.
   - Arm Developer Suite Evaluation Version 1.1 (45 days):
     - Compiladores para C, C++.
     - Ensamblador y enlazador.
     - Entorno de desarrollo integrado.
     - Requisitos: Pentium y 32 Mb de RAM.

- MetaWare High C/C++ ™ ARM: Herramienta de Desarrollo para Sistemas Embedidos.
- Multi2000: entorno de desarrollo integrado para Atmel Arm AT91 producido por Green Hills. Único que funciona bajo W98SE y Linux.

8) Debido a que Linux con X-Window le exige demasiado al P133: reinstalamos Linux, pero sólo en modo texto.
   - Se mantiene el mismo esquema de particiones, formateando las particiones hdc1 (ext3 /root) y hdc2 (swap).
   - Se descarta GNOME y la configuración X-Window.
   - Proceso de instalación de paquetes...instalación completa.
9) Reiniciamos bajo Linux: modo terminal.
   - Pretendemos configurar acceso a Internet: comando “ifconfig eth0”.
   - Pero no reconoce a “eth0” como dispositivo válido. Quizás la tarjeta Ethernet no esté correctamente configurada bajo Linux.

Apunte 7:

1) Para configurar TCP/IP bajo Linux nos informamos de cómo está definida en W98, es decir:

Dirección IP	193.147.161.82
Puerta de Enlace	193.147.161.1
DNS Primario	193.147.161.202
DNS Secundario	150.214.130.15

Nos informamos de cómo se usa el comando "ifconfig" bajo Linux. Linux, mediante "man ifconfig" nos informa lo siguiente:

- "ifconfig": configura interfaces de red residentes en el Kernel. Su efecto se manifiesta tras la reiniciación del sistema.
- Opciones: "-a": muestra el estado de todos los interfaces, incluso de aquéllos que se hayan caído.
- Familias de direcciones: "ifconfig interface_name family"; posibles familias: inet (TCP/IP), inetv6 (Ipv6).
- Interfaces: "eth0": primer interfaz de Ethernet.

Al escribir en el terminal de comandos de Linux: "ifconfig eth0" nos contesta que ése dispositivo no existe.

2) Bajo W98SE instalamos el SW que proporciona el Kit de desarrollo de ATMEL:
   - Adobe Acrobat Reader 4.0:
     - C:\archivos de programa\Adobe\Acrobat 4.0
   - ARM Developer Suite Evaluation Version v1.1:
     - D:\SW_de_Desarrollo\ARM\ADSv1_1
     - Actualiza autoexec.bat guardando la versión anterior como C:\autoexec.001
     - Reinicio del sistema.
   - MetaWare Toolset for ARM 4.5ª:
     - D:\SW_de_Desarrollo\hcarm_4.5ª
     - Copia del antiguo autoexec.bat en autoexec.old.
   - CodeWright:
     - D:\SW_de_Desarrollo\StarBase\CodeWright
     - Uso de clave del producto: CW650-7X7B-68845
     - Durante el proceso de instalación se definen los siguientes parámetros:
   - Se usa como compilador: Otro, no Microsoft, no Borland.
   - Entornos de desarrollo integrados en CodeWright:

Sí	Microsoft Visual Basic
Sí	Microsoft Visual C++
Sí	Borland C++
Sí	Borland C++ Builder Development Enviroment
No	Borland Delphi 3.2
No	Texas Instruments Code Composer Studio

Librerías CodeSense instaladas:

Borland C++ Runtime Library	Borland OWL Library	CodeWright Library
Microsoft ATL Library	Microsoft MFC Library	Windows API + MSC Library
Java 1.2.2 Library	Java 1.1.8 Library	Java 1.3.0 Library

El programa de instalación solicita dónde están situados los ejecutables asociados a cada librería; como no están instalados, pulsamos "Skip".

- Multi2000 “Green Hills”:
  - D:\SW_de_Desarrollo\GHS

3) Actualizamos Linux instalando los paquetes de X-Windows junto con GNOME (gestor de ventanas) y los productos de desarrollo.
   - Objetivo: lograr configurar TCP/IP.
   - Se transfieren 59 nuevos paquetes manteniendo las particiones y LILO.
4) Iniciamos Linux: pero se abre el modo terminal ➔ reinstalamos todos los paquetes de Linux: 580 paquetes o 1316 Mb.
5) Mediante el PII350 buscamos el www.linuxnewbies.org un foro en el que me pueda aclarar cómo se usa “ifconfig” para instalar una nueva interfaz de acceso a la red; esto encuentro:
   - Para definir un nuevo interfaz:
     - ifconfig eth0 IPaddress broadcast IPaddress netmask 255.255.255.0
     - route add-net default gw IPaddress
     - ifconfig –a
   - Para activarlo:
     - ifconfig eth0 up

   Con la primera línea de comandos me responde el sistema que “eth0” no está definido y que no es un tipo de dispositivo; con lo que no puedo seguir con el resto de líneas de comando.
   Sigo buscando en el mismo foro y encuentro el comando “netconfig” que supuestamente abre una ventana de diálogo en la que se configura la conexión TCP/IP. Tecleo “netconfig” apareciendo dicha ventana: hay cuatro campos: dirección IP, máscara de subred, DNS primario y DNS secundario. Los completo y acepto. Seguidamente tecleo en el terminal “ifconfig eht0” para que muestre la configuración del acceso: pero no reconoce el dispositivo.
6) Bajo W98SE: en el mismo foro encontré una posible solución al problema de que no encuentre “msnp32.dll” aunque sí esté en el sistema: desinstalar los componentes de red y volverlos a instalar completamente. Quizás también solucione el acceso a Internet desde W98.
   - Desinstalación de la tarjeta SMC EtherEZ (8416).
   - W98 inicia automáticamente su reinstalación, para ello solicita una serie de ficheros que pertenecen al CD de W98SE. Como están comprimidos en cabinet’s, debo extraerlos manualmente mediante WinRar. Los archivos que solicita son:

(i) net7.cab		
(c) arp.exe	dhcpsvc.exe	ftp.exe
icmp.dll	inetmib1.dll	ipcfg.dll
iphlpapi.dll	msafd.dll	networks
protocol	Qosname.dll	rapilib.dll
rnr20.dll	Routetab.dll	rpcltc3.dll
prclts3.dll	Rsvpsp.dll	services
smc8000.dos	snmpapi.dll	vdhcp.386
vip.386	vnbt.386	vtcp.386
vtdi.386	vudp.386	
net8.cab		
hosts.sam	ipconfig.exe	Imhosts.sam
locproxy.exe	locproxy2.exe	nbtstat.exe
netstat.exe	ping.exe	minicpfg.exe
route.exe	Rsvp.exe	telnet.exe

telnet.hlp	tracert.exe	
net9.cab		
smc8000n.sys	wshtcp.vxd	

Descomprimimos todos estos archivos del CD de W98SE a disquete en el PII350 para grabarlos en C:\mis Documentos\archivos de instalación del P133.

- Desinstalamos la configuración parcial que hizo W98 de la tarjeta cuando solicitó todos estos archivos (los que no solicitó los instaló).
- Reiniciamos la instalación de la tarjeta indicando el origen de los archivos: instalación completada satisfactoriamente con todos los archivos y reinicio del sistema.
- En la ventana de diálogo "Conexiones de Red":
  - Agregamos una conexión nueva del tipo TCP/IP.
  - Se asocia ésta a EtherEZ (8416) automáticamente....OK
  - Definimos los campos de la dirección IP, DNS, puerta de enlace y máscara de subred.
  - Se reinstalan los controladores de la tarjeta (todos los archivos anteriores).

➔ pero sigue sin detectar la red.

7) Reinicio: sigue apareciendo el mensaje:
"No se puede cargar la biblioteca de vínculos dinámicos: msnp32.dll. El sistema no puede hallar el archivo especificado. Microsoft Network no disponible.

8) Mediante Soulseek he encontrado a tres usuarios con msnp32.dll compartido, pero aparecen dos tipos de archivo: uno de 80 kb y otro de 84. El que está en el P133 es de 80 (una versión anterior antendiendo a las fechas de creación de ambos archivos). Lo descargo al PII350 y lo copio en el P133, dentro de C:\windows\system. Al reiniciar el P133 sigue apareciendo el mismo mensaje.
9) En el foro de www.linuxnewbies.org aparece un mensaje que dice que una posible solución al problema del mensaje de msnp32.dll es que faltan ciertos componentes de red en la instalación de W98SE. Luego, aunque se copie el archivo en el lugar especificado, no se cambia el registro de archivos de configuración mediante regedit.exe.
10) En "Agregar o quitar programas" entramos en la opción de instalación de componentes de Windows. Seleccionamos los siguiente a añadir:
    - Del apartado "Comunicaciones" seleccionamos todos los paquetes.
    - Del de "Herramientas de Red" se añade:
      - Conexión compartida a Internet.
      - Web-Based Entreprise Management.

➔ a la hora de instalar estos paquetes W98 me remite al CD de instalación. Como en él todos los archivos están comprimidos, no son accesibles directamente. Son demasiados los archivos solicitados como para instalarlos manualmente uno a uno → e:\w98\instalar.exe.

11) Reinstalación de W98:
    - Comprobación del sistema:
      - ScanDisk: W98 no se cerró correctamente.
      - Buscar componentes instalados.
      - Comprobando si existe suficiente espacio en disco.
    - Guardamos por seguridad los archivos asociados a la configuración actual de W98SE en D:\; en total alcanza los 110 Mb.
    - Generación de disco de inicio W98.
    - Copia de los archivos.
12) W98 se reinicia por 1ª vez:

- error de protocolo 0.
- error 6715: el nombre del grupo de trabajo ha de ser distinto que el del PC. Este problema se abordará más tarde.

13) Probamos si W98 puede conectarse a la red: SÍ.

Apunte 8:

21-11-2002

1) Instalamos Linux nuevamente, pero sin X-Window: en modo texto.
   - Pretendemos actualizar el sistema para evitar la instalación de todos los paquetes de la distribución; tan sólo se pretende que la opción, tras el inicio del sistema, sea el modo intérprete de comandos.
   - Iniciamos la instalación con el CD1 de la distribución de Linux:
     - Mantenemos la estructura de particiones de los HD's y el gestor de arranque.
     - Linux comienza la búsqueda de paquetes a actualizar → no encuentra ninguno.
     - Durante el proceso de actualización no aparece la opción que defina el modo texto como el modo por defecto del sistema.
   - Problema: el modo por defecto, tras este punto, es el de X-Window.
   - Solución: instalación completa sin los paquetes asociados a X-Window.
2) Arranque de W98:
   - Al arrancar no muestra que necesita "msnp32.dll".
   - Pretendemos configurar la impresora, pero no conozco la ruta de red de la misma:
     - W98 solicita los drivers que están en el CD de W98SE, pero al estar comprimidos en .cab no son accesibles directamente.
     - No me he traído el CD de W98SE.
     - Solución: actualizar el sistema desde v4.windowsupdate.microsoft.com.
   - Instalamos la versión de evaluación de Norton Antivirus 2003, lo descargamos de www.symantec.com, quedando registrado hasta el 22-11-03. Generamos los discos de rescate en D:\Rescue.
3) Adquisición de fuente de alimentación para la placa de ATMEL.
   - Especificaciones:
     - Tensión de salida programable: 4.5V, 7.5V , 9V, 12V.
     - Corriente de salida: hasta 1300mA.
   - Falta confirmar la garantía.
4) Adquisición de dos destornilladores: uno de estrella y otro plano.
5) Búsqueda de información sobre FCS para mejorar la estabilidad de vuelo y disminuir la carga del piloto.
6) Problema: el kit viene con un cable serie para conectar la placa al PC, pero el P133 no posee puerto serie libre. Solución: el PII350 sí posee puerto serie libre, en él tendremos que trabajar. Trabajar con Linux va a quedar subyugado a la condición de que se disponga de otra placa madre para PC que tenga un puerto serie libre.

22-11-2002.

1) Instalamos en el PII350 el software de desarrollo que traía el kit de ATMEL:
   - ARM Developer Suite V1.1 Evaluation Version.
     - Directorio destino: D:\SW_FCS\ARM\ADSv1_1.
     - Problema: el gestor del fichero de licencia del software me indica que éste está corrupto. Solicita que se descargue desde el servidor del proveedor del software.
     - Solución: se descarga de Internet un gestor de licencia nuevo, pero sigue estando corrupto. No se soluciona el problema.

25-11-2002.

   - Multi2000 de "Green Hills.
     - Directorio destino: D:\SW_FCS\GHS.

        - Indica que para solicitar al fabricante la licencia se ejecute "License Generator".
    - MetaWare Toolset for ARM 4.5ª:
        - Directorio destino: D:\SW_FCS\hcarm_4.5ª.
        - Nombre de Usuario: Sergio Latorre.
        - Nombre de Compañía: EsCuela de InGenieros – Proyecto AERO1 FCS.
    - CodeWright IDE:
        - Directorios de destino:
            - D:\SW_FCS\hcarm_4.5ª\CodeWrightInstall
            - D:\SW_FCS\Starbase\CodeWright
        - Instalamos las librerías "Code Sense".
        - Mismo nombre de Usuario y de Compañía que en el caso anterior.
2) Instalamos completamente el Linux en P133, saltándonos la configuración de X-Window. Se mantiene la estructura de particiones, formateando las asociadas a Linux, y el gestor de arranque. El número de paquetes a instalar es menor que si se incluye la configuración de X-Windows. Se definen dos usuarios:
    - /root: administrador del sistema, cuya clave es: mustek11.
    - serlat: usuario ordinario, cuya clave es: mustek11.
    - El modo de arranque es ahora: texto.

27-11-2002.

1) Seguimos la recomendación mediante "Getting Started" del CD de ARM:
    a) Arrancar y probar la placa de evaluación.
        i) Comprobar la posición del jumper JP1 en STD (standard)....OK
        ii) Alimentar la placa:
            (1) Conectamos la fuente (programada a 9V[1]) a la placa....OK
            (2) Conectamos la fuente a la red eléctrica....OK

    Interruptor en ON → Error: la placa no responde.

            (3) OFF y desenchufamos la fuente de la red eléctrica.
                (a) El transformador huele a sobrecalentamiento. Lo abrimos y observamos que la clavija de salida, interiormente, está en corto consigo misma....la chapita estaba doblada.
                (b) La desdoblamos con ayuda del destornillador plano, sobre el cual salta un arco voltaico que descarga al transformador de la fuente.
            (4) Enchufamos el transformador a la red eléctrica y ON.
                (a) LED's de la placa de encienden simultáneamente: de D1 a D11.
                (b) Tras unos segundos se apagan de D2 a D9 → boot completado: OK.
            (5) OFF.
    b) Iniciar la prueba de las funciones de la placa ("on-Board Functional Test SW").
        i) Conectar cable serie entre los puertos serie A y B.

        → problema: los conectores de los puertos en la placa están demasiado juntos como para que se pueda enchufar los dos extremos del cable simultáneamente.

        → solución: con ayuda del destornillador plano cortamos una capa del borde de plástico de los extremos del cable serie. Se consigue enchufar ambos extremos.

        ii) Mantenemos pulsado SW1 y ON → no responde la placa.
            (1) El problema no está en el transformador, ya hemos arreglado los falsos contactos.

[1] según ARM la placa puede ser alimentada entre 7 y 9 V, con 1000 mA.

(2) Movemos ligeramente el conector J1 que alimenta la placa. Hacía mal contacto.
(3) Se inicia el arranque satisfactoriamente:
(a) Desde D1 a D11 parpadean una vez; se suelta SW1 tal y como indica la recomendación de ARM, iniciando "Functional Test".
(b) De D1 a D5 parpadean 1 vez: test pasado OK.

iii) OFF.

c) Conectamos el PC (COM1) a la placa (SERIAL A) mediante el cable serie.
d) Instalamos la librería AT91 en el PC para ADS v1.1:
i) Extraer el archivo zip con la librería v2.10 desde del CD de ARM.
(1) Se fuerza por programa a que se extraiga sobre C:\.
ii) Iniciamos ADS v1.1:
(1) Se intenta abrir CodeWarrior for ADS v1.1:
(a) ➔ Error: archivo de licencia corrupto.
(b) ➔ Solución: bajarlo de Internet como aconseja el gestor de la licencia.
(i) http://www.c-dilla.com/support/lms.html
(ii) ftp://ftp.macrovisioneruope.com/outgoing/cdalms.exe
(c) ➔ Error: al ejecutar cdalms.exe muestra el mensaje de error: "(1615:6): usuarios de Windows NT deben comprobar los derechos de acceso con el Administrador. Si el problema persiste, reinicie su PC".
(d) ➔ Solución: reiniciamos el PII350 y reinstalamos el producto ADS v1.1 (Evaluation Version).
(e) ➔ Error: mismo error que antes.

e) Instalamos la librería AT91 en el PC para MULTI2000:
i) Extraemos la librería AT91 sobre C:\ desde el CD de ARM.
ii) Iniciamos Multi2000:
(1) ➔ Error: mensaje: "Unable to get license for multi: Failure 100: Network error: Unable to send messages to license software: No such a file o directory."

28-11-2002.

1) Buscamos en el manual de la placa información relativa al consumo de la placa.
a) Motivo: mi profesor tutor lo necesita para hacer un cálculo rápido sobre las baterías de alimentación que se montarán en el Helicóptero.
b) Datos de la placa según "User Guide":
i) DC 7V – 9V y 1A.
ii) Medida del consumo de corriente en tiempo real:
(1) Mediante jumpers JP5A/B (V_{DDIO}) y JP7A/B (V_{DDCORE}). Permite conectar un amperímetro directamente. Referencia en página 5-3 y 32/39 de "User Guide" para la placa EB40a.

2) Solución al problema de Licencia de ADS v1.1:
a) Desintalar la versión v1.1 actual.
b) Del CD1 7-10-2002 se descomprime la versión v1.2 (que es completamente funcional).
i) Directorio destino: D:\SW_FCS\temporal.
ii) Directorio origen:
E:\AT91\Entorno de Desarrollo Windows\ARM_developer_suite_v1_2.rar.

c) Tras la descompresión se genera dentro de "..\temporal" el subdirectorio "..\ARM.developer.suite v1.2".

i) Ejecutamos “setup.exe” de ese subdirectorio.
(1) Directorio destino de instalación: D:\SW_FCS\ARM\ADSv1_2.
(2) Opción de Instalación: completa.
ii) Instalando componentes...OK.

d) Estudio de puesta en funcionamiento del crack:
i) Subdirectorio: ..\temporal.
(1) Según “install.txt”: tras la instalación, al empezar “ARM License Wizard”, se selecciona “instalar licencia” abriendo el archivo “..\crack\license.dat”.
ii) Tras la instalación de componentes se abre “ARM License Wizard”:
(1) Browse: se abre por defecto en D:\SW_FCS\ARM\ADSv1_2\Bin para buscar “license.dat”. Copiamos “..\crack\license.dat” en esa carpeta.
(2) Le indicamos al gestor de la licencia que en el fichero “D:\SW_FCS\ARM\ADSv1_2\Bin\license.dat” está la licencia válida ignorando el campo del código de acceso (lo dejamos en blanco).
(3) Pulsamos “Next” aceptándose la licencia proporcionada.
(4) Se pregunta si la licencia dada el temporal o permanente: indicamos que es permanente. Pulsamos “Finish”.
(5) Mensaje del instalador: “ADS v1.2 está instalado completamente.

e) Comprobamos que ADS 1.2 se abre correctamente...OK.

3) Instalación de librerías AT91 en el PC para ADS1.2:
a) Extraemos desde el CD de ARM las librerías en C:\.
b) Mediante Metrowerks CodeWarrior for ADSv1.2 abrimos la librería de nombre C:\AT91\Software\parts\r40008\r40008_lib16.mcp.
i) Debido a que la librería proporcionada por ARM está diseñada para ADS1.1, ADS1.2 la actualiza correctamente.
c) Seleccionar todos los paquetes de la librería y MAKE desde CodeWarrior.
i) Se inicia el compilado de todos los paquetes dando 0 errores y 1 aviso.
(1) Según recomendaciones de ARM, el aviso lo ignoramos.
ii) Se cierra el proyecto y la ventana de errores/avisos.

4) Reconstruir la biblioteca de controladores de ARM ® para ADS1.2:
a) Abrimos con CodeWarrior: c:\at91\software\drivers\lib_drv\lib_drv_16.mcp.
b) Actualización automática a la última versión.
c) Seleccionamos MAKE: compilándose todos los archivos sin errores ni avisos.

5) Reconstruir un proyecto ejemplo: led swing...OK
6) Reconstruir la librería Tools...OK
7) Descargar un programa estándar en la placa y ejecutarlo:
a) Al depurar aparecen 2 errores: mensaje: no se encuentra la librería r40008_lib16.a.
b) ➔ No se puede descargar hasta que no se depure correctamente.

Apunte 9:

15-2-2003:

1) Del P133 han retirado los HD's: no se reconoce ningún HD en el proceso de arranque del sistema ni S.O. válido. Utilizo el disquete de arranque de W98 e inicio el sistema, el cual no detecta ningún HD en la máquina. Menos mal que todo el SW que en este PC estaba instalado y que me fuera útil para el proyecto estaba instalado en el PII350.
2) Del PII350 no han modificado nada, por lo menos a lo referente a mi cuenta.
   a) Renuevo la contraseña de la cuenta: Mustek11.
   b) La cuenta sigue todavía activa.

Apunte 10:

20-2-2003:

1) Objetivos:
   a) Chequeo del funcionamiento de la placa de ATMEL.
   b) Llevar a cabo todo el "Getting Started" que proporciona el fabricante.
2) Planteamiento: Getting Started:
   a) Insertamos el CD de ATMEL: ARM Thumb
   b) Plantea dos paquetes de SW:
      i) Green Hills: que nos da problemas de no validación de la licencia; con lo que no podemos trabajar con él.
      ii) ARM ADS v1.2: el cual ya está instalado en el PII350 y funciona correctamente.
   c) Seleccionamos la opción: ARM ADS – ANGEL: esta opción incluye el SW que trae consigo la placa dado por el fabricante ATMEL.
   d) Inicio y Prueba de la placa de Evaluación.
      i) Configuración y Alimentación correctas: JP1 en STD y VDD = 9V.
      ii) Arranque completado:
         (1) D1,D2,...,D11 se encienden iniciándose el chequeo interno de la placa.
         (2) D1,D10 y D11 permanecen encendidos: chequeo OK.
      iii) Inicio del SW de prueba de sistemas en la placa. Para ello se llevan a cabo las siguientes operaciones:
         (1) Conexión del cable serie entre Serial A y Serial B: OK
         (2) Apretamos el botón RESET y SW1.
         (3) Soltamos SW1 iniciándose la prueba de sistemas.
            (a) D1,D2,...,D5 parpadean sólo 1 vez consecutivamente.
         (4) Chequeo de sistemas: OK
   e) Instalación de ARM Developer Suite (ADS) v1.2: ya estaba instalada.
   f) Conexión del PC a la placa:
      i) Apagar la placa.
      ii) Conexión desde Com1 del PC a Serial A de la placa.
   g) Instalación de la librería AT91 proporcionada por el fabricante:
      i) Extracción de la librería:
         (1) Auto – Extracción en C:\
         (2) librerías en C:\at91\software\....:OK
      ii) Reconstrucción de una librería de dispositivo con ARM v1.2:
         (1) Abrimos "Metrowerks CodeWarrior for ADS v1.2":
         (2) Abrimos un nuevo proyecto:
            (a) c:\at91\software\parts\r40008\r40008_lib16.mcp
            (b) Incidencia: el programa solicita la actualización del proyecto y de todos los archivos asociados a la última versión de ADS. Aceptamos.
         (3) En "Targets" se seleccionan todas las variantes (Edit → Select All)...OK
         (4) Project → Make...no aparece ninguna ventana de error: OK
      iii) Reconstrucción de un proyecto de la librería AT91 con ADS: "Led Swing":
         (1) File → Open...
            (a) c:\at91\software\projects\led_swing_eb40a\led_swing.mcp
            (b) Se actualiza a la última versión todos los componentes del proyecto.
         (2) En "Targets" seleccionamos todas las variantes para compilarlas.
         (3) Project → Make...ERROR: no encuentra la librería r40008_lib16.a
            (a) Solución: buscamos el archivo con Windows, el cual está en:

(i) c:\at91\software\parts\r40008\r40008_llib16_Data\optm_full\r40008_lib.a
(ii) En la pestaña “Files” se incluye esta librería.
(b) Project → Make...OK: devuelve informe del resultado
(4) File → Close...Cerramos el proyecto.
iv) Reconstrucción de la librería “Tools” con ADS:
(1) File → Open: abrimos un proyecto de:
(a) c:\at91\software\tools\flash_downloader\flash_16x4\flash_16x4.mcp
(b) Actualización a la última versión...OK
(2) En la pestaña “Targets” se selecciona “int_sram”.
(3) Project → Make...OK: informe del proceso.

h) Descarga de un programa standard a la placa y ejecutarlo en ella:
i) File→ Open: c:\at91\software\projects\led_swing_eb40a\led_swing.mcp
ii) En “Targets” se aplica Edit→ Select All y Project→Make
(1) 17 Errores: símbolos indefinidos. De las 4 variantes los 17 errores aparecen al compilar la variante “flash”; esto lo hemos averiguado al compilar cada variante independientemente.
(2) Decisión: sigamos adelante con los errores.
iii) Alimentamos la placa e inicio de chequeo automático...OK
iv) Project→ Debug:
(1) Inicio del depurador AXD (en ventana propia). Configuración por defecto:“ARMulator ADS v1.2 (Build 805)” en el registro del depurador.
v) Configuramos el depurador:
(1) Options→Configure Target...(dentro de AXD).
(2) Selecionamos: configure APD...OK
(3) Seleccionamos “serial driver” para su configuración. Es el puerto de comunicaciones entre el PC y la placa.
(4) Configuramos la conexión: COM1, 38400 bps.
vi) En el registro del depurador (RDI Log) aparece: “Angel Debug Monitor (serial) for at91eb40a”.
vii) Execute → Go: dos veces:➔ Los Led’s parpadean formando una onda....OK
(1) El puerto serie Serial A registra actividad como lo atestiguan los leds TX_D y RX_D.
viii) Execute → Stop: detenemos el programa...OK

i) Grabar “Led Swing” en la memoria Flash de la placa y ejecutarlo en ella.
i) **AVISO: JP1 en USER**: evitamos sobreescribir sobre los primeros sectores de la memoria Flash.
(1) Conmutamos el Jumper 1 (JP1) de STD a USER...OK
ii) En AXD: File→Load Image...
(1) c:\at91\software\tools\flash_downloader\flash_16x4\flash_16x4_Data\int_sram\flash_16x4.axf...OK
(2) Aparece el mensaje: ¿reemplazar la imagen cargada?...Damos OK
iii) Execute → GO (2 veces)
En la “Console” aparece: “*** Load Address ***”
(1) Escribimos: “0x1000000”.
iv) En la “Console” aparece: “***File Address ***”
(1) Escribimos:”c:\at91\software\projects\led_swing_eb40a\led_swing_data\flash\led_swing.bin” y return...OK
(a) ERROR: “Cannot open file image” como mensaje en “Console”.
(b) Solución: buscar el archivo .bin con el Windows.

Apunte 11:

21-2-2003:

(2) Retomamos, desde el arranque de la placa hasta el punto anterior, en proceso de descarga en flash de un código y su ejecución en la placa. La ARM7TDMI-Console sirve para interactuar con el micro de la placa mediante programa al usar "printf" y "sscanf".

(3) buscamos el archivo led_swing.bin con Windows: se encuentra en la ruta:c:\at91\software\projects\led_swing_eb40a\led_swing_Data\flash\led_swing.bin. Puesto que en el manual "Getting Started" aparece "led_swing_data" con minúscula; probamos a indicarle con mayúscula la ruta del archivo a encontrar por si distingue entre mayúsculas y minúsculas.

(4) Aparece en la consola un mensaje de error porque no existe tal archivo. Estudiamos el código fuente que gestiona la consola; en él aparece: "imagen=fopen(name,'rb')" devolviendo NULL.

(5) Estudiamos la sintaxis de "fopen"; para ello nos documentamos en http://c.conclase.net/ficheros/ficheros002.html, en donde aparece: "FILE *fopen(char *nombre, char *modo)" siendo:
   (a) 'r': sólo lectura: exige que exista el fichero.
   (b) 'b': tipo binario.

v) File → Exit.

vi) Apagamos la placa y la arrancamos en modo usuario (JP1=USER) ➔ los led's están oscilando...OK el programa había sido descargado a la memoria flash de la placa satisfactoriamente.

3) Objetivo:

- Diseñar un programa de prueba tipo "Hola Mundo".
- Descargarlo a la flash de la placa.
- Ejecutarlo en ella.
- Nota:
  - Al iniciar la placa siempre en modo JP1=STD.
  - Al cargar una imagen en flash: siempre en modo JP1=USER.
  - Al cargar una imagen en sram: en modo JP1=STD.
  - Definir el objetivo en AXD si aparece el modo ARMulator (ventana de LOG).

a) Se arranca la placa en modo JP1=STD.

b) Abrir CodeWarrior for ADS v1.2.

c) File→New:
   i) Pestaña "Project": optamos por ARM Executable Image de entre las opciones de proyectos nuevos.
   ii) Ruta: c:\Documents and Settigs\serlat\Mis documentos\Códigos fuente\.
   iii) Nombre el proyecto: Hello World.mcp...OK

d) Definir "Input Files to link": pulsamos el botón "new text file".
   i) Se abre una ventana en la que se puede escribir el código C++.
   ii) Escribimos el código y lo salvamos como: "...\Códigos fuente\Hello World.c".

e) Project→Add Hello World to project...OK
   i) El archivo .c ya forma parte del proyecto en curso.

f) Make...Se abre una ventana de Errors and Warnings: los errores son corregidos satisfactoriamente...OK

g) Make...OK: se presenta el informe del resultado.

h) JP1=USER.
i) Debug:
   i) Se abre AXD; el cual descarga la imagen: led's de puerto serial A activada.
   ii) Execute→Go: en la ventana de consola aparece el texto "Hello World"...OK

4) Objetivo:

- Diseñar un programa de prueba que genere, iterando, los 10 primeros términos de la serie de Fibonacci.
- Descargar el código en flash y ejecutarlo.

a) Se enciende la placa en modo JP1=STD.
b) Escribir el código abriendo un nuevo proyecto y la ventana de texto.
c) Guardar el código en: ...\Códigos fuente\fibonacci.c.
d) JP1=USER.
e) Debug...OK
f) Execute → Go. Se presenta por pantalla los 10 primeros términos de Fibonacci...OK

22-2-2003.

5) Objetivo:

- Diseño de Banco de Pruebas para uC de placa.
  - Tratamiento con enteros.
  - Tratamiento con flotantes.
  - Medidas de tiempo de proceso (uso del tiempo de sistema).
- →Determinación del peso de cómputo límite según el tiempo de muestreo.

a) Análisis de Problemática:

Desarrollo de rutinas que involucren el tratamiento con enteros o con flotantes. Planteamiento:

- Para Flotantes:
  - Generación aleatoria de secuencia de N valores.
    - Operaciones simples: sumas y multiplicaciones sobre la secuencia.
    - Cálculos Estadísticos: Media, varianza y superiores. Límite en el orden.
  - Inversión de Matrices: límite en el orden.
- Para enteros: límite en el número de operaciones y en el orden del filtro.
  - Concretamos los algoritmos de flotantes para enteros, en los casos que esto sea posible.
  - Implementación de ecuaciones en diferencias de sistemas estables con estados posibles bajo aritmética entera.
  - Número de operaciones: sumas, multiplicaciones.

b) Medida del peso del cómputo: número de operaciones / tiempo de operación.

- Número de sumas, multiplicaciones para enteros y flotantes.
- Número de lecturas/escrituras: estudio del código ensamblador (ARM Thumb) asociado a las operaciones sobre enteros y flotantes.

Para ello se desarrollan rutinas simples que midan el tiempo consumido por una serie de cálculos repetitivos (mejor medida que un único cálculo), sean sumas, multiplicaciones, lectura de registro y escritura de registro. Exige el estudio de las funciones de gestión del tiempo del sistema que proporciona ADS v1.2.

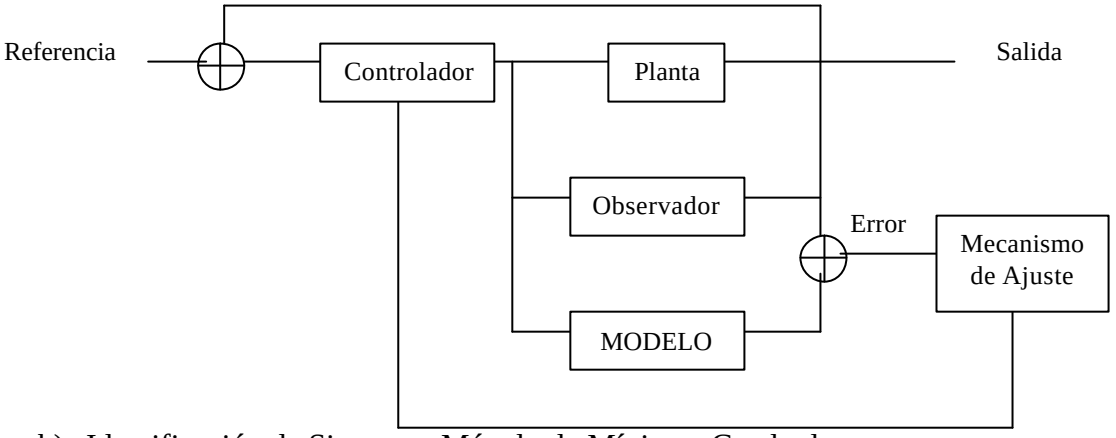
c) Presentación de informe de resultados.

6) Objetivo:

- Estudio del Sistema: diferentes planteamientos de observación, estimación y control del sistema dinámico.
- Aplicación de las medidas de tiempo para la cuantificación del consumo de cada planteamiento para su posterior discriminación.

- ➔ Informe comparativo de diferentes métodos según consumo de cómputo.

a) Análisis del problema: diferentes planteamientos (no excluyentes).
   i) Problema de Identificación de sistemas.
      (1) identificación en línea ➔ control adaptativo.
      (2) Método de mínimos cuadráticos:
         (a) recursivo (on-line).
         (b) no recursivo (off-line).
      (3) Método de mínimos cuadrados ponderados: factor de olvido.
   ii) Problema del Regulador de Mínima Varianza: autosintonía (control adaptativo).
   iii) Problema del Regulador LQR. (off-line): obtención del horizonte infinito.
   iv) Problema de Observadores: filtro de Kalman (LQG).
   v) Problema MPC: Model Predictive Control.
   vi) Control Adaptativo: autoajustable.
      (1) Bucle abierto.
      (2) Bucle cerrado.
   vii) Esquema de bloques:



b) Identificación de Sistemas: Método de Mínimos Cuadrados.
   i) Estudio de sistema monovariable:
      (1) Convergencia y factor de olvido: estabilidad.
      (2) Tiempo de muestreo.
      (3) Medidas de Consumo de cómputo: conteo del número de operaciones y relación con el consumo/operación.
   ii) Extensión a sistema multivariable:
      (1) Definición de sistemas multivariables mediante ecuaciones en diferencia.
      (2) Factor de olvido multivariable.
      (3) Definición de Supermatriz de covarianzas (triple índice: filas, columnas, profundidad).
      (4) Definición de Supermatriz de parámetros estimados.
      (5) Estudio de convergencia y factor de olvido.
      (6) Tiempo de muestreo.
      (7) Medidas de Consumo de cómputo.

c) Regulador de Mínima Varianza.

Apunte 12:

12-3-2002:

1) **Objetivo:** realización práctica del punto 5.a) del día 22-2-2003 (Apunte 11) para el tratamiento de flotantes. Todos los códigos se han probado en un PII350 de sobremesa.
   a) Dentro del proyecto "bancoDePruebas" se elaboran las siguientes rutinas:
      i) Inicialización de una tabla con muestras de la variable aleatoria U(0,1). Sobre ella se aplicarán los bancos de pruebas de suma y producto.
      ii) Uso de bucles "for" para recorrer la tabla y operar.
      iii) Estudio de las funciones de gestión de tiempo de Windows, habiéndose encontrado:
         (1) void gettime(struct time t); Esta función inicializa la estructura time con la hora actual del sistema. Siendo:

```
struct time{
unsigned char ti_min;   //minutes
unsigned char ti_hour;  //hours
unsigned char ti_hund; //hundredth of second
unsigned char ti_sec;          //seconds
        };
```

Esta función se encuentra en la librería "dos.h".

      iv) Se ha elaborado la función "float tiempoSistema(float *tiempo); devuelve el tiempo del sistema en segundos, sobre la variable "tiempo", con una precisión de centésima de segundo.

   b) Tabla de medidas de tiempo:

(i) Nº de muestras	Tiempo de Suma (s)	Tiempo de Producto (s)
10000	0.06	0.05
30000	0.17	0.16
50000	0.28	0.27
100000	0.55	0.55
500000	2.69	2.69
1000000	5.71	5.49

Notas sobre resultados:

   i) Se invierte gran parte del tiempo de procesado en la reserva dinámica de la tabla de muestras aleatorias. La reserva se hace de una sola vez, siendo no practicable en el sistema uC; por lo tanto se realizará la generación de las sucesivas muestras aleatorias en línea, es decir, a la par que se opera con ellas.
   ii) Gran parte del tiempo computado se debe a operaciones de lectura/escritura, tal y como se demuestra observando el código ensamblador asociado. Una solución es no tratar con muestras situadas en una tabla, sino ir generándolas conforme se necesiten.

   c) Las modificaciones se contemplan en el proyecto "bancoDePruebas2.exe". Cuya tabla de medidas de tiempo:

(ii) Nº de muestras	Tiempo de Suma (s)	Tiempo de Producto (s)
10^6	11.21	10.76

Problemática: La doble llamada a la función VA() para que genere valores aleatorios ralentiza marcadamente el tiempo de cómputo; el cual no está asociado, en mayor medida, al tiempo de proceso.

d) Solución: hacer 1 llamada a VA() con el número máximo de cifras significativas que permita la codificación del formato “int” y operar con esa muestra repetidas veces.
   i) Calibramos el número máximo de cifras significativas a 9, con más cifras significativas el valor de “rango” no es coherente. Véase función “diezElevadoA” de “bancoDePruebas3.cpp”.

e) Las modificaciones se observan en el proyecto “bancoDePruebas3.exe” cuya tabla de medidas es la siguiente:

Nº de operaciones	Tiempo de Suma (s)	Tiempo de Producto (s)
10^5	0	0
5*10^5	0.49	0
10^6	0.27	0
5*10^6	0.71	0.17
10^7	0.77	0.33
5*10^7	2.17	1.48
10^8	3.52	3.02
10^9	30.98	30.32

Se observa que:

   i) para <10^6 la influencia multitarea por división en tiempo de Windows es manifiesta.
   ii) >5*10^7: esta influencia va desapareciendo por tiempos promedios.
   iii) En el código ensamblador asociado aparecen las instrucciones que implementan el bucle “for”, además de las propias para sumar flotantes “fadd” y para multiplicar flotantes “fmul”. Midamos cuánto tiempo consumen estas instrucciones: para ello se dejan como comentario las instrucciones relativas al procesado de datos.

f) Las modificaciones se observan en el proyecto “bancoDePruebas3_1.exe” cuya tabla de medidas es la siguiente:

Nº de operaciones	Tiempo de Suma (s)	Tiempo de Producto (s)
10^7	0.77	0.16
10^8	2.36	1.81
5*10^8	9.88	9.28
10^9	19.33	18.56

Para más operaciones el efecto multitarea es minimiza, sobre todo cuando se procede a cerrar todas las aplicaciones que corren bajo Windows salvo las del sistema.

   i) Atendiendo a los datos:

Nº de operaciones	Tiempo de Suma (s)	Tiempo de Producto (s)
10^9	11.65	11.96

   ii) Para la suma se invierte el 37.6 % en las operaciones.

   iii) Para el producto se invierte el 39.45 % en las operaciones.

g) Estos algoritmos han de probarse en el uC de ATMEL; para ello se han de buscar las rutinas de acceso al tiempo del sistema propias del micro.

Apunte 13:

13.3.2003:

1) **Objetivo:** Probar "bancoDePruebas3" y "bancoDePruebas3_1" en uC de ATMEL. Para ello se requiere:
   a) Adaptar las rutinas de detección y gestión de tiempo del sistema a la arquitectura del micro ARM7TDMI de ATMEL.
2) **Problemática:**
   a) He conectado el transformador a la placa y a la red.
   b) He arrancado la placa y ha ejecutado el chequeo interno correctamente.
      i) ¡El transformador se ha calentado en exceso!
      ii) Lo desenchufo y lo abro con el destornillador: El transistor de potencia que tiene a la salida parece abollado. Vuelvo a cerrarlo.
   c) Le comento estas incidencias a Ismael por correo.
   d) Para comprobar el funcionamiento del transformador medimos con un voltímetro la salida del mismo cuando está conectado a la red y sin carga. (Espero a que se enfríe previamente). ➔ el transformador da 0.0V a la salida.
3) **Solución:** hablo con Ismael y decidimos comprar otro transformador (mañana iré):
   a) Tienda: "M. León"; c/Castilla.
   b) A cuenta del "Departamento de Automática" y a nombre de "Ismael Alcalá".
   c) Entregar el albarán a Ismael.

1) **Objetivo:** Búsqueda de librerías y rutinas de gestión de tiempo en C++.
   a) El uC de ATMEL es compatible con las rutinas de ANSI C; siendo éstas mejoradas para el propio micro, tal y como lo aseguran los manuales de ADS.
2) **Bibliografía:**
   a) Título: "ANSI C a su alcance."
      i) Autor: Herbert Schildt.
      ii) Editorial: Osborne / McGraw-Hills.
      iii) ISBN: 84-7615-603-0.
      iv) Funciones asociadas:

- clock_t clock(void);
  - Compatible con ANSI C e incluido en "time.h".
  - Devuelve el número de ciclos de reloj del sistema desde el comienzo de ejecución de la aplicación que la llama. Para transformar el segundos se divide el resultado por la macro CLK_TCK.
  - El tipo clock_t = long unsigned int.
  - Página 409.
- time_t time(time_t *hora);
  - Compatible con ANSI C e incluido en "time.h".
  - Devuelve el tiempo del sistema en el formato "hora del calendario".
  - time_t = long unsigned int.
  - Si hora==NULL: devuelve el tiempo del sistema.
  - Si hora!=NULL: además, inicializa el puntero dado con el tiempo del sistema.
  - Página 413.
- double difftime(time_t hora2, time_t hora1);
  - Compatible con ANSI C e incluido en "time.h".
  - Devuelve hora2 – hora1 en segundos.
  - Página 411.

   b) Título: "Borland C++: Manual de Referencia."
      i) Autor: Herbert Schildt.
      ii) Editorial: Osborne / McGraw-Hills.

iii) ISBN: 84-481-1141-9.
iv) Funciones asociadas.

- long biostime(int orden, long nuevaHora);
  - Pertenece a “bios.h”, no compatible con ANSI C.
  - Lee o escribe el reloj del sistema; siendo la hora dentro de cada día a partir de las 0:00 am.
  - Si orden==0: devuelve el valor de la hora del sistema en ticks.
  - Si orden==1: devuelve el valor de la hora dada por nuevaHora y la escribe en el reloj del sistema.
  - Página 393.
- void delay(unsigned time);
  - Pertenece a “dos.h”, no compatible con ANSI C.
  - Detiene la ejecución de un programa durante time (ms).
  - Función relacionada: sleep().
  - Página 399.
- void ftime(struct timeb *hora);
  - Pertenece a “sys\time.h”; no compatible con ANSI C.
  - Página 417.
- int kbhit(void);
  - Pertenece a “conio.h”; no compatible con ANSI C.
  - Página 431.
- void sleep(unsigned time);
  - Pertenece a “dos.h”; no compatible con ANSI C.
  - Suspende la ejecución del programa durante time (segundos).
  - Página 441.
- void exit(int estado);
  - Pertenece a “stdlib.h”; siendo compatible con ANSI C.
  - Devuelve la ejecución al sistema operativo.
  - Si estado == 0: terminación normal. (EXIT_SUCCESS).
  - Si estado == 1: EXIT_FAILURE.
  - Página 73.
- Uso de variables globales en archivos compilados por separado:
  - Archivo1:

```
int x,y;
char c;
main(void)
{
   ...
}
void func11(void)
{
x=23;
}
```

  - Archivo 2:

```
extern int x,y;
extern char c;
void func21(void)
{
 x=y/10;
}
 void func22(void)
{
 y=10;
}
```

Apunte 14:

14.3.2003:

1) **Objetivo:** Comprobación del correcto funcionamiento del transformador nuevo. Para ello se llevan a cabo las siguientes medidas:

Valor Programado de Tensión (V)	Valor medido (V)
4.5	6.40
6	7.81
7.5	9.52
9	11.30
12	14.85

Las medidas se hicieron en abierto, con el transformador descargado.

   a) Las medidas se hicieron en orden creciente de voltaje para evitar la carga parcial del bobinado secundario alterándose el valor de la medida.
   b) Entre medidas se apagó el transformador (punto rojo del interruptor oculto).

2) **Objetivo:** Repetición de las medidas para 6, 7.5 y 9 V programados conectando la placa del uC de ATMEL, se obtiene la siguiente tabla:

Valor programado de Tensión (V)	Valor medido (V)	Notas
6	7.05	Chequeo OK
7.5	8.61	Chequeo OK
9	10.44	Chequeo OK

Objetivo: Repetición de las medidas para 6, 7.5 y 9 V programador con la placa conectada y ejecutando “led_swing”.

Valor programado de Tensión (V)	Valor medido (V)	Notas
6	7.05	Chequeo OK: LEDs bailando
7.5	8.75	Chequeo OK: LEDs bailando
9	10.20	Chequeo OK: LEDs bailando

DECISIÓN: **El transformador se programará a 6 V (dándose 7.05 V).**

3) **Objetivo:** adaptar al standard ANSI C el código del “bancoDePruebas3_1.cpp”.
4) **Resultado:** código “bancoDePruebas3_5.cpp”. Hay que probarlo sobre el uC de ATMEL (funciona en PII233).

Apunte 15:

20.3.2003:

1) **Objetivo:** Depuración del proyecto "bancoDePruebas3_5.cpp":
2) Para ello hacemos lo siguiente:

- Excluimos del proyecto "bancoDePruebas.mcp" el fichero bancoDePruebas3_2.cpp e incluimos el bancoDePruebas3_5.cpp.
- Encendemos el uC habiendo programado el transformador a 6 V previamente.
- Conmutamos el JP1 de STD a USER.
- Depuramos el código: carga de la imagen correcta.
  - Processor Views → Source...: se selecciona el fichero .cpp.
    - Para depurar aparece el código ensamblador y el C++.
    - Nota: el tiempo devuelto el siempre un entero (múltiplo del segundo), como si no tratara con partes del segundo.
- Se define un nuevo proyecto (imagen ejecutable): bancoDePruebas3_5.mcp.
- Descargamos la imagen en modo usuario.
  - No se logra saber el valor de "time" al depurar. No me permite el debugger definir un watch con dicha variable.
  - Probamos con diferentes números de operaciones para asegurar que se tarde más de un segundo en su ejecución. Los ticks que se devuelven son siempre múltiplo de 100.
  - Para que eventos puntuales no afecten en demasía al resultado de la medida de tiempo de proceso elevamos lo suficiente el número de operaciones. Mediante diversas pruebas, se constata que se requiere de un mínimo de 100 millones de operaciones.
- Incluimos en el código printf("...",CLK_TCK); para que, mediante la macro CLK_TCK, se presente por pantalla el número de ticks por segundo. Obtenemos 100 ticks/seg.
- Tabla de resultados de "bancoDePruebas3_7.cpp":

Nº Iteraciones	**tiempo Suma (s)**	**tiempo Producto (s)**
$100 * 10^6$	9	9
$200 * 10^6$	19	18
$500 * 10^6$	46	46
10^9	91	91
$1.5 * 10^9$	136	137

- Repetimos las medidas, pero con los cuerpos de los bucles "for" comentados:

Nº Iteraciones	**tiempo Suma (s)**	**tiempo Producto (s)**
$100 * 10^6$	9	9
$200 * 10^6$	18	18
$500 * 10^6$	46	46
10^9	91	91
$1.5 * 10^9$	136	137

  - Comentarios: los resultados son excesivamente similares.
- Modificamos el código de forma que sólo se soliciten las operaciones de suma y depuramos.
- Intentamos colocar un "breakpoint" sobre la operación "va+va", pero no se permite.

- De acuerdo con el código ensamblador asociado al bucle "for" (for(k=0;k<noper;k++) {va+va;}) tenemos:

```
mov r0,#0          //k=0;
cmp r0,r4          //k<noper;
bge 0x80e8;        //salto si mayor el igual
add r0,r0,#1       //k++;
```

  - En donde no aparece ninguna instrucción asociada a "va+va".
- Al intentar definir un watch asociado a "va" el depurador nos muestra un mensaje que dice: Variable removed by Compiler (optimization).
- En ADS v1.2: Edit → DebugRel Settings... → ARM C++ Compiler → Debug/Opt/
  - Definimos Optimization Level en Minimum (best debug wiev).
- En el depurador intentamos definir un breakpoint en "va+va": ahora sí lo permite.
- Observamos el código ensamblador asociado a "va+va": NOP (no operation).
- Definimos aux=va+va y compilamos. Ahora no tiene asociada NOP.
- Tabla de resultados:

Nº Iteraciones	tiempo Suma (s)	tiempo Producto (s)
$10 * 10^6$	7	8
$20 * 10^6$	15	15
$50 * 10^6$	38	38
$100 * 10^6$	76	77
$200 * 10^6$	151	155

- Tabla de resultado (con los cuerpos de "for" comentados):

Nº Iteraciones	tiempo Suma (s)	tiempo Producto (s)
$10 * 10^6$	1	1
$20 * 10^6$	2	2
$50 * 10^6$	4	4
$100 * 10^6$	9	9
$200 * 10^6$	18	19

- Comentarios: ahora sí se observan grandes diferencias entre realizar los cálculos o sólo la gestión de los mismos.
  - Para la suma se usa el 88.08 % del tiempo en los cálculos; realizándose 1,5035 MFLOPS[2] para sumas.
  - Para el producto, el 87.74 %; realizándose 1,4705 MFLOPS para productos.

3) **Objetivo:** Repetir las medidas, pero con enteros.
4) Para ello:
   a) Cambiamos el tipo de dato de las variables a operar de flotante a entero.
   b) Guardamos los cambios como "bancoDePruebasInt.cpp".
   c) Definimos el nivel de optimización del compilador al mínimo.
   d) Tabla de Medidas:

Nº operaciones	Tiempo Suma (s)		Tiempo Producto (s)	
$10 * 10^6$	1	2	1	2
$20 * 10^6$	2	4	2	4
$50 * 10^6$	4	10	5	10
$100 * 10^6$	9	20	9	21
$200 * 10^6$	18	40	19	42

   i) Las columnas 2ª y 4ª se refieren al cómputo sin el cuerpo de "for".
   ii) Las columnas 3ª y 5ª se refieren al cómputo con el cuerpo de "for".

[2] MFLOPS: million of floating point instructions per second

iii) Para la suma se usa el 55% del tiempo para los cálculos; realizándose 9.090 MINTS[3] para sumas.
iv) Para el producto, el 54.7%; realizándose 8.704 MINTS para productos.

5) **Objetivo:** desarrollo de filtro de orden 1 y medida de resultados.
6) Códigos fuente asociados: "filtro_orden1.cpp" para flotantes y, para enteros, "filtro_orden1Int.cpp".
7) Tabla de resultados:

Flotantes		**Enteros**	
Nº operaciones	**Tiempo (s)**	**Nº operaciones**	**Tiempo (s)**
1000	21	1000	19

a) Luego se invierten unos 21 ms/ciclo con flotantes y 19 ms/ciclo con enteros.
b) En este cómputo se ha contabilizado sólo el tiempo invertido en la resolución de la ecuación en diferencias. A la hora de considerar el límite temporal en los lazos de control hay que contar todo el tiempo invertido, tanto en las operaciones como en el control de flujo.

21.3.2003:

1) **Objetivo:** operaciones estadísticas:
   a) Características del proceso:
      i) En línea.
      ii) Cálculo de medias, varianzas y de estadísticos superiores:
         (1) $m = E[x]$; $chi_n = E[(x-m)^n]$;
      iii) Estudio de tiempos según el orden (n) y el número de muestras (N).
      iv) Extensión a aritmética entera y flotante.
   b) Definimos el proyecto Imagen Ejecutable ARM con el nombre: "CalculoEstadistico.mcp".
2) Tabla de Resultados (flotantes) de "CalculoEstadistico2.cpp":

		Orden del estadístico (n)											
		2			**3**			**5**			**10**		
(N)	**100**	4	1.001	0.329	3	1.000	0.245	2	0.999	0.162	2	1.000	0.086
	1000	21	1.000	0.333	16	1.000	0.250	12	1.000	0.166	20	1.000	0.090
	2000	37	1.000	0.333	40	1.000	0.250	42	1.000	0.166	41	1.000	0.091
	5000	92	1.000	0.333	102	1.000	0.250	101	1.000	0.167	86	1.000	0.091
	10000	204	1.000	0.333	199	1.000	0.250	207	1.000	0.167	214	1.000	0.091

   a) Siendo, tras N iteraciones:
      i) 1ª, 4ª, 7ª y 10ª columnas los segundos invertidos.
      ii) 2ª, 5ª, 8ª y 11ª columnas la media estimada de las muestras.
      iii) 3ª, 6ª, 9ª y 12ª columnas el estadístico de orden n estimado.
   b) Siendo la media y variación de la variable aleatoria de entrada: 1.0 y 0.01, respectivamente.
3) Tabla de Resultados (enteros) de "CalculoEstadistico3.cpp":

		Orden del estadístico (n)											
		2			**3**			**5**			**10**		
(N)	**100**	1	10000	0	3	10000	0	2	0.999	0.162	1	10000	0
	1000	18	10000	0	22	10000	0	20	1.000	0.166	20	10000	0
	2000	42	10000	0	47	10000	0	35	1.000	0.166	45	10000	0
	5000	113	10000	0	103	10000	0	104	1.000	0.167	108	10000	0
	10000	235	10000	0	225	10000	0		1.000	0.167		10000	0

[3] MINTS: million of integer instructions per second.

a) No existe mucha diferencia entre ambos casos puesto que para el cálculo de chi_n se precisa el uso de la función "pow" que trabaja con "double".
b) La media y variación de la variable aleatoria de entrada: 10000 y 100.

4) **Objetivo:** Adaptación del Código "InvertirMatriz2.cpp" al uC.
   a) Generación de nueva imagen ejecutable para ARM: "inversionMatriz.mcp".
   b) Adaptación de Rutinas y Librerías.
   c) Generalización de "invertirMatriz3.cpp" para trabajar con enteros o flotantes, para ello:
      i) typedef float entrada; //para trabajar con flotantes.
      ii) typedef integer entrada; //para trabajar con enteros.
      iii) Adaptación de las funciones pertinentes.
   d) Compilado correcto: eliminación de errores: OK
5) Probar casos sencillos de matrices para el depurado.
6) Incluir cuenta del tiempo: "invertirMatriz4.cpp".
7) Tabla de Resultados:

	Tiempo (s)	
Dimensión (n x n)	**Flotantes**	**Enteros**
4	0	No procede
5	0	No procede
6	0	No procede

8) Definir archivo "matrices.cpp" que contenga las funciones de gestión y de tratamiento de matrices.
   a) En "invertirMatriz5.cpp" está el código necesario para presentar los datos de la tabla anterior y las llamadas a las funciones de gestión de matrices.
   b) La generación de entradas aleatorias de la matriz se ha redefinido para que sea correcta: y=(max-min)*U(0,1)+min➔ y = U(min,max);
9) Para órdenes mayores o iguales a 7 el algoritmo provoca una interrupción al uC que aborta la ejecución del algoritmo.
10) **Objetivo:** Estudio de la cantidad de entradas de matriz que han de estar reservadas en memoria al mismo tiempo para invertir una matriz de orden n x n.
    - Propia matriz: n^2 entradas.
    - 1 matriz adjunta de orden n-1: $(n-1)^2$ entradas.
    - 1 llamada a "Det" con orden n-2: 1 adjunta de orden n-2: $(n-2)^2$ entradas.
    - Y así sucesivamente hasta: 1 llamada a "Det" con orden 2: 2^2 entradas.

    a) Luego, aunando todas: N <= SUM{k^2,k=2,k=n};
    b) La placa de ATMEL ARM7eb40a dispone de 256kb para usuario de RAM; luego: Si N = 256 * 2^{10} / sizeof(float) = 65536 ➔ n<60.
    c) Así pues, el orden queda limitado a 60 según la memoria del sistema.
    d) El límite por tiempo es tremendamente más restrictivo, así que no hará falta modificar el código para que contemple el caso de que la reserva dinámica de memoria falle (devuelva NULL).
11) **Objetivo:** a partir del "filtro_orden1.cpp" extensión para un filtro de orden n,m:
    - $G(z^{-1}) = (b_1z^{-1} + ... + b_mz^{-m}) / (1 + a_1z^{-1} + ... + a_nz^{-n})$; m<n: causalidad;

    a) Generalización para que se use en tanto para flotantes como para enteros.

Apunte 16:

26.3.2003:

1) **Objetivo:** Depurar el código de filtroNM.cpp en uC y obtener tabla de resultados.
2) **Problemática:** la generación de los coeficientes del filtro no está correctamente definida, se redefine correctamente mediante: y = (max - min) x + min.
3) Se reconstruye el código para que trate diferente número de iteraciones, obteniéndose la siguiente tabla de tiempos de cómputo.

		Nº Operaciones		
Tiempo (s)	**Orden (n, m)**	**100**	**1000**	**10000**
	(1,1)	1	14	210
	(2,2)	2	26	-
	(3,3)	2	25	-
	(4,4)	2	20	-
	(5,5)	2	23	-
	(10,10)	4	25	-
	(20,20)	2	29	-
	(50,50)	3	31	227

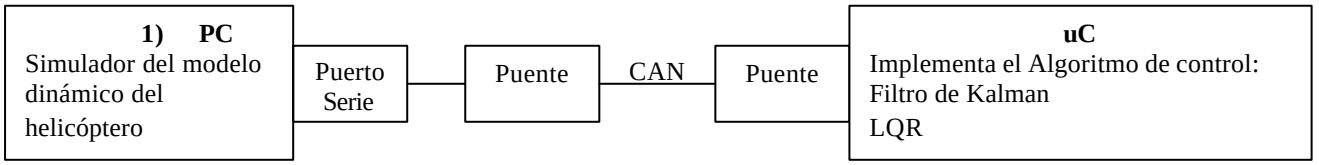
4) Resultado: la mayor influencia en el tiempo de cómputo la tiene el control de flujo más que el propio cálculo de la respuesta del filtro, esto se manifiesta en la escasa influencia del orden.

Apunte 17:

2.4.2003:

1) **Objetivo:** Probar "Matrices" en uC para obtener la tabla de tiempos asociada (bajo el compilador "Angel").
   a) Archivos fuente: "InvertirMatriz5.cpp", "matrices.cpp".
   b) Proyecto: "inversionMatriz.mcp".
2) Editamos "matrices.cpp":
   a) ERROR: en la función asignaValoresMatrices la generación aleatoria no esta correctamente definida.
      i) Se modifica por ésta: y=(max-min)x+min = U(min,max); x = U(0,1);
      ii) Comprobando la generación aleatoria: ahora sí es correcta.
3) Realizamos las medidas para flotantes: max=10.0; min = -10.0.
   a) ERROR: para orden >=7x7 la rutina cuelga el uC; para órdenes menores se ejecuta correctamente invirtiendo <100ms en la inversión.
   b) Depuramos para orden >=7x7, pero no se arregla: posiblemente le falte memoria en la placa para realizar los cálculos.
4) **Objetivo:** Notas de conversación con Ismael Alcalá:

- Sistema del helicóptero: $x'=Ax+Bu$; $x \in \Re^{6x1}$; $u \in \Re^{4x1}$; $A \in \Re^{6x6}$; $B \in \Re^{6x4}$;
  - No intervienen matrices de orden mayor que 6 ➔ usemos el algoritmo de "invertirMatriz5.cpp".
- Vector de ganancias de realimentación del vector de estados: $u=-Kx$; $K \in \Re^{4x6}$;
  - Las constantes se calculan fuera de línea bajo Matlab-Simulink-RTW. RTW genera un esqueleto en C++ del diagrama de bloques de Simulink, el cual se puede compilar para el uC.
- Diseño: Tiempo de muestreo de **1ms**.
- Diseño de Aplicaciones:
  - Rutina que simule el sistema (en PC) y su control (en uC) para estimación de tiempos.
  - Desarrollo de aplicación de comunicaciones vía puerto serie en Visual Basic o Visual C++ para plataforma PC.
  - Desarrollo de interfaz para la introducción de referencias (orientación, velocidad) y la indicación de los resultados de la simulación para plataforma PC.
- Diagrama de bloques del sistema:



5) **Objetivo:** Obtener una medida de tiempo más precisa de la inversión de matrices: para ello se pretende el desarrollo de una rutina que invierta N veces la misma matriz.
   a) Archivos fuente: "invertirMatriz6.cpp" y "matrices.cpp".
   b) ERROR: problemas de "excepción" en uC conforme se aumenta el número de iteraciones: Menor umbral en iteraciones a mayor orden de la inversión.

3.04.2003:

1) **Objetivo:** implementar u=+Kx; efectuar medidas de tiempo:
   a) medir tiempo de iteración.
   b) Sea:
      i) $K=[K_{ij}]$; i = 1,2,3,4; j = 1,2,...,6;
      ii) u1,...,u6,x1,...,x6: variables; aux1,...,aux6: variables auxiliares.
   c) Esquema:

```
t0=clock();
REPETIR N
leer x;          //leer de 6 posiciones de memoria (emulación bus CAN).
calcula u
escribir u;      //escribir en 6 posiciones de memoria (emulación bus CAN).
FIN REPETIR
t1=clock()-t0;
```

   d) Guardado en .../C++/ControlProp.cpp.
   e) ➔ efectuar medidas y calibrar Tm mínimo (el lunes 7.04.2003).

2) **Objetivo:** Desarrollo de aplicación de comunicaciones vía puerto serie en Visual Basic.
   a) Ficheros fuente: prueba1.vbp, frmPrueba;
   b) Para la versión mejorada: Prueba2.vbp.
   c) Ver => Objeto: ver el formulario.

04.04.2003:

1) **Objetivo:** Estudio del Filtro de Kalman y del algoritmo LQR
   a) Desarrollo de su algoritmo en C++ para el uC (on-line).
   b) Obtención de medidas de tiempo.
2) **Notas:**

- Existen dos tareas:
  - Cada N muestras (iteración) se actualiza LQR: cambios en las codiciones externas o internas han podido modificar A, B o C: cambia, por tanto, K_{INF}.
  - En cada muestra se actualiza el estimador de KALMAN.
- Requiere de kernel mutitarea en tiempo real.
  - Buscar librerías de tiempo real para los compiladores del uC entre la documentación que ha proporcionado el fabricante.

3) **Objetivo:** Plateamiento de Algoritmo para la gestión de tiempos:
   a) Archivo fuente: “TiempoPrueba.cpp”.
   b) Probar en uC el Lunes 7.04.03.
   c) Intefaz multitarea con gestión de tiempos:

```
int retraso(clock_t, clock_t);
static clock_t t; //marca de tiempo de la tarea (se ha de definir una por tarea)
//global: vista en “main” y en “nombre_tarea”
//static: guarda el valor entre diferentes llamadas a nombre_tarea
void nombre_tarea(clock_t);
int main(void)          //kernel en tiempo real emulado
{
        clock_t tm;     //tiempo de muestreo de la tarea (uno por tarea)
        tm = XXX;       //valor numérico del tm (ms)
        t=clock();      //iniciamos la marca de tiempo

for(;;)
        {
                nombre_tarea™;      //1 línea por tarea
                //código de fondo (background)
        }
        return 0;
}
void nombre_tarea(clock_t tm)               //se constituye una función por tarea
{
```

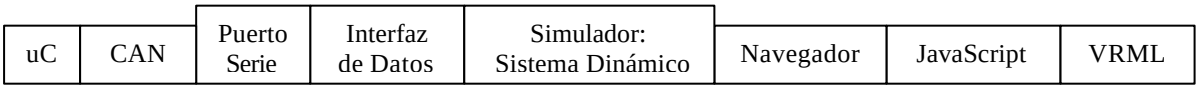
```
        if(retraso(tm,t)) return;          // si 1, no se ejecuta la tarea
        //cuerpo de la tarea               // si 0: se ejecuta la tarea
        t=clock();      //actualizamos la marca de la tarea (sólo si ésta se ejecuta)
        return;
}
int retraso(clock_t tm, clock_t t0)      //t0 ha de ser local y dinámica a "retraso"
{
   if(tm>clock()-t0) return 1;           // devuelve 1 se  no se ha cumplido el tiempo tm
   else return 0;                        // devuelve 0 si ya se ha cumplido el tiempo tm
}
```

   d) Comunicación entre tareas: uso de variables globales (todas las variables son atómicas).

4) **Objetivo:** Medidas del peso del kernel (main) frente al resto de tareas:
   a) medidas del peso del cómputo de cada tarea.
   b) medidas del número de veces que se ejecuta frente al que debiera ejecutarse cada tarea.
   c) medidas de tiempo de ejecución de fondo (background).
   d) ➔ Calibración respecto al tiempo de muestreo de cada tarea.
5) Resultado de medidas: si tm<10ms => la diferencia frente al comportamiento deseado > 15%, aunque el peso de fondo supere el 99.95 %.
   a) medidas tomadas en PII233 sobre W98.
   b) repetir las medidas sobre uC de Atmel el Lunes 07.04.03.

05.04.2003:

1) **Notas:** Esquema de Aplicaciones - HW:



2) **Documentación:**
   a) Java: www.sun.com/ downloads.html
   b) VRML:
      i) Especificaciones:
         (1) www.web3d.org/Specifications;
         (2) www.web3d.org/fs-specifications.html;
         (3) www.sdk.web3d.org;
      ii) Editores y Visores:
         (1) VRMLPad: ftp://ftp.parallelgraphics.com/parallelgraphics/bin1/vrmlpad.exe
         (2) Cortona (iexplore): ftp://ftp.parallelgraphics.com/parallelgraphics/bin1/cortvrml.exe

Apunte 18:

9.4.2003:

1) **Objetivo:** Prueba de algoritmos en uC de ATMEL:
   a) Obtención de medidas de tiempos.
   b) Códigos fuente: ControlProp.cpp (u=-Kx) y tiempoPrueba.cpp.
2) Depuramos el primer código para que sea óptimo: cambios guardados en ControlProp2.cpp.
   a) Opciones del compilador C++ para ARM en su grado mínimo de optimización: se pretende que toda instrucción en código fuente tenga su correspondiente imagen en código máquina.
   b) Se depura instrucción a instrucción para observar que pasa por todas ellas.

Nº repeticiones	**Tiempo (cs)**	**Tiempo (ms) por iteración**
10^5	300	30
$5 \cdot 10^5$	1600	32
10^6	3300	33

   c) Luego, se toma como 33 μs por iteración. Cada tick del gestor de tiempos del uC se genera cada centésima de segundo.
   d) Gracias a los tiempos de respuesta tan pequeños, se permite introducir un estimador de estado (observador o filtro de Kalman).
3) Depuramos "tiempoPrueba.cpp".
   a) Eliminamos los avisos (warnings) al compilar.
   b) Para 2500 ticks (25 segundos) se obtiene:

Peso por tarea (%)		**Nº veces que se ejecutó**	**Nº de veces que se debió ejecutar**	**Tiempo por chequeo (cs)**
Tarea 0 :	8.027	24	2500	8.36
Tarea 1 :	8.027	25	833	8.36
Tarea 2 :	8.361	25	500	8.36
Fondo :	75.585			8.36

   c) El modelo planteado de kernel NO es válido: buscamos las librerías para el compilador sobre gestión de tiempos.
   d) Observando los subdirectorios de ARM hemos encontrado la librería time.h. La cual incluye el prototipo de la función "clock()". No se ha encontrado, en el resto de librerías funciones del tipo "wait" o "delay".

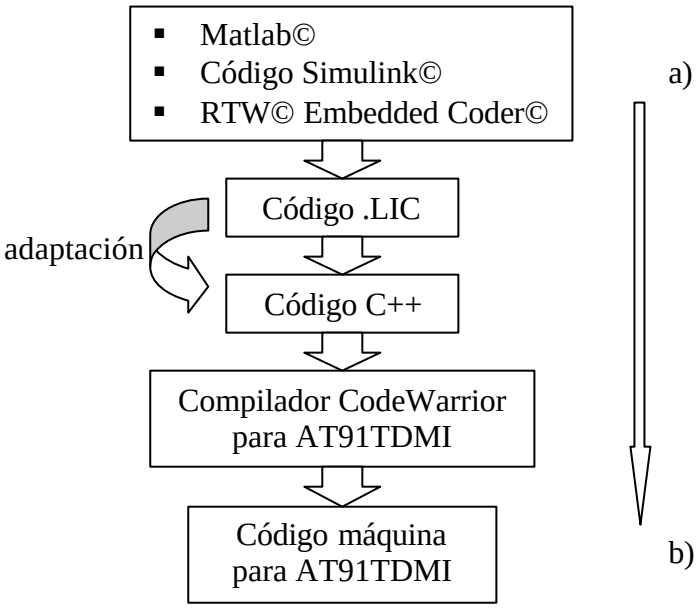
Apunte 19:

10.04.2003:

1) **Objetivo:** Estudio del funcionamiento y de las facilidades del paquete "Real Time Workshop© Embedded Coder©" de Matlab©.
2) Esquema jerárquico de dependencia:

<table>
<tr><td colspan="2">Atmel AT91TDMI</td><td>Motorola MPC555</td><td>Texas Instruments TIC6000</td></tr>
<tr><td colspan="4">Real Time Workshop© Embedded Coder©</td></tr>
<tr><td colspan="4">Real Time Workshop©</td></tr>
<tr><td colspan="2">Simulink©</td><td colspan="2">compilador Matlab© C</td></tr>
<tr><td colspan="4">Matlab©</td></tr>
</table>

3) Diagrama de Bloques sobre el funcionamiento:



4) Planteamiento:
   a) Estudio de los ejemplos y facilidades de "Motorola© MPC555" y de (Texas Instruments© TIC6000".
      i) Compilado automático.
      ii) Adaptación de código .lic a código .cpp automatizada.
      iii) Monitorización de Señales (uso de observadores – watchers).
   b) Generación y prueba en simulink de los ejemplos y estudiar el código .lic generado con la intención de obtener relaciones de adaptación para el código .cpp adecuado a AT91TMDI de Atmel©.
      i) Ejemplos tipo: multiplicación de matriz con vector.
      ii) señal escalón con un "scope" (osciloscopio): monitorización.

13.04.2003:

5) **Objetivo:** se ha discriminado la documentación del CD "Ayuda Matlab 6" relativa al objetivo anterior pasándose a formato papel para su mejor estudio. En concreto se ha resaltado:
   a) RTW: Arquitectura del Programa.
      i) Introducción.
      ii) Modelo de Ejecución.

(1) Temporizado del Modelo (Model Timing).
(2) Ejecución de Programa.

iii) Prototipado Rápido dentro del Marco de Programa.
(1) Arquitectura.
(2) Componentes dependientes del Sistema.
(3) Componentes independientes del Sistema.
(4) Componentes para Aplicación.

iv) Marco de trabajo para Programa Imbuido (Embedded Program Framework).

b) Codificador Imbuido 3.0 para RTW: Notas de Realización.
i) Introducción.

c) Codificador Imbuido 2.0 para RTW: Notas de Realización.

d) Codificador Imbuido 1.0 para RTW: Notas de Realización.

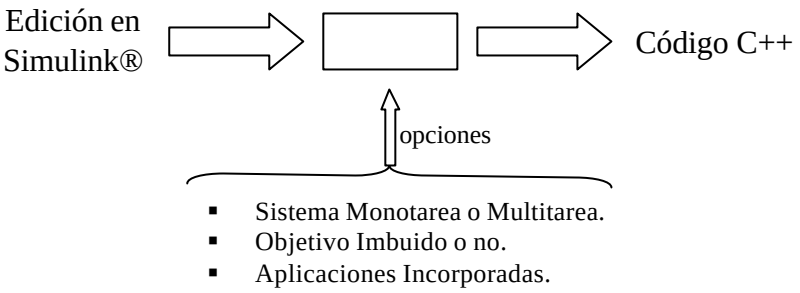
e) TIC6000 DSP 2.0: Notas de Realización.

f) MPC555: Notas de Realización.

Apunte 20:

23.4.2003:

1) **Objetivo:** Elaborar un resumen – esquema de la funcionalidad de RTW® y RTW Embedded Coder®.
   a) Arquitectura de Programa para RTW:



      i) Estilos de Código:
         (1) Imbuido: RTW Embedded Coder ®.
         (2) Prototipado Rápido (simulación a través del código):
            (a) Tiempo real genérico y reubicable (malloc) sin SO en tiempo real.
            (b) Funciones-S: se incluye una función-S dentro de un modelo de Simulink® mediante un archivo C-Mex.
            (c) VxWorks®: pruebas sobre un SO en tiempo real.
            (d) Real Time Windows/Dos ®: tiempo real por interrupción.
   b) Modelos de Ejecución.
      i) Monotarea y sin tiempo real.
      ii) Multitarea (uso de pid) y sin tiempo real.
      iii) Monotarea y con tiempo real (temporizador para interrupción periódica).
      iv) Multitarea (uso de pid) y tiempo real por interrupción.
   c) Entorno de Programa para Prototipado Rápido.
      i) Generación de Código:
         (1) Ecuaciones del Modelo del Sistema.
         (2) Bloques de Parámetros e Inicialización.
      ii) Código adicional: API (estructuras para encapsulado de datos):
         (1) Modelos de Aplicación.
         (2) Entorno de Programación.
         (3) Generación de Programa Autónomo mediante RTW Builder®.
   d) Arquitectura para Prototipado Rápido:
      i) Interfaz en Tiempo de Ejecución:
         (1) Componentes Dependientes del Sistema:
            (a) Programa Principal.
            (b) Temporización e Interrupciones.
            (c) E/S, historial de datos.
            (d) Comunicación con Módulo Externo.
         (2) Componentes Independientes del Sistema:
            (a) Resolvedores de la Integración.
            (b) Plan de Ejecución del Modelo.
      ii) Componentes de Aplicación:
         (1) Código del Modelo.
         (2) Estructura de Datos del Modelo.
         (3) Funciones-S.
         (4) Parámetros del Modelo.
   e) Main en RTW:

i) Inicialización:
   (1) Constructor del Modelo.
   (2) Tamaño de la Información del Modelo.
   (3) Vectores de Muestreo y Offsets, Estados y manejo de Interrupciones.
   (4) Define la tarea de fondo (background).
ii) Ejecución del Modelo:
   (1) En background: tareas:
      (a) Comunicación con el Host.
      (b) Espera hasta el próximo Instante de Muestreo.
   (2) Rutina de Gestión de Interrupción del Temporizador:
      (a) Ejecutar el Modelo.
      (b) Guardar datos y resultados del Modelo en búffer.
iii) Terminar el Programa: Destructor del Modelo:
   (1) destrucción de la estructura de datos del modelo.
   (2) desaloja de la memoria en modelo (si dinámico).
   (3) escribe resultados en un archivo para su estudio posterior.

f) Entorno de Programa Imbuido: Análogo al caso de Prototipado Rápido, pero optimizando en proceso en tamaño, velocidad y recursos.

g) RTW Embedded Coder®.
i) Referencias: Real Time Workshop ® Embedded Coder ® User's Guide.
ii) Esquemas de Aplicación:
   (1) Una placa con control en tiempo real sin un SO en tiempo real.
   (2) Una WS con WxWorks® para depurar el código.
iii) Capacidades:
   (1) programa principal autogenerado: ert_main.c.
   (2) Generación de Plan de Trabajo (Scheduler) determinista: multitarea y multi - tiempo de muestro.
   (3) Interrupción Asíncrona.
   (4) Validación del Código desde Simulink® (depurador).
   (5) Modificación manual del código C.
   (6) Generación de resumen y documentación como de las trazas en HTML.
iv) Restricciones:
   (1) No permite estados continuos: sólo discretos.
   (2) Evitar los bloques que dependan del tiempo absoluto cuando se diseñe algoritmos de tiempo infinito.
   (3) Toda función-S debe tener su fichero .tlc.
   (4) Bloques no soportados[4]:
      (a) Continuos.
      (b) Mantenedor de orden 1 (FOH).
      (c) Funciones y Tablas de Matlab®:
         (i) Matlab Fcn.
         (ii) M-file, Fortran S-functions, C-Mex S-functions.
      (d) Constantes algebraicas.
      (e) Fuentes: clock, chirp signal, ramp, repeating sequence, signal generator.

h) Objetivo Imbuido para Motorola® MPC555.
i) Producción de Código desde Simulink®:
   (1) Para compiladores cruzados: genera archivos de proyecto y códigos objeto (soporta Metrowerks CodeWarrior).

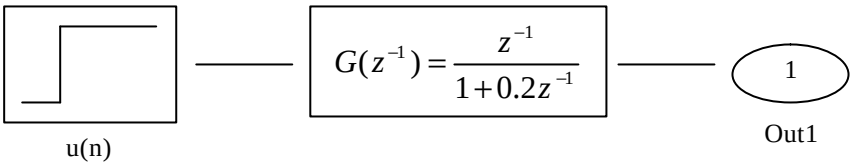
[4] página 5-2: RTW Embedded Coder User's Guide: ecoder_ug.pdf.

(2) Soporta depurador: Metrowerks CodeWarrior Debugger.

ii) CAN Calibration Protocol Block: compatible con Visual C/C++ Compiler.

24.4.2003:

2) **Objetivo:** Estudio de Funcionamiento de Matlab® y RTW®:
   a) Ejemplo Prueba: "untitled". Diagrama Simulink®:



   b) Al compilar para RTW® se genera una serie de archivos en la carpeta del espacio de trabajo de Matlab®: d:\matlab6p5\work\untitled_grt_rtw.
      i) Siendo:
         (1) grt: generic real time.
         (2) rtw: real time workshop.
      ii) Archivos .c y .h:
         (1) untitled_data.c.
         (2) untitled.c: proporciona el interfaz en tiempo de ejecución.
         (3) untitled_types.h.
         (4) untitled_private.h.
         (5) rtmodel.h: contiene el modelo de Simulink ®.
         (6) untitled.h.
   c) Dentro de la carpeta D:\matlab6p5\work se generaron los archivos:
      i) untitled.exe: al ejecutarlo genera "untitled.mat" con los resultados de la simulación en el formato .mat adecuado para Matlab®. Al no definir un objetivo en el compilador RTW® se tomó por defecto: pcwin.
      ii) untitled.mdl: describe el modelo del sistema bajo Simulink ®.

3) **Objetivo:** Probamos el compilado de los ficheros generados por RTW® en Metrowerks CodeWarrior ®.
   a) Generamos un nuevo proyecto en CodeWarrior ®:
      i) Ruta de Directorio: ...serlat\Mis Documentos\Códigos Fuente\RTW\untitled.
      ii) Archivo: untitled.mcp.
   b) Añadimos todos los archivos .c y .h que generó Matlab® al proyecto nuevo.
   c) Preprocesamos uno a uno:
      i) Se producen errores en los archivos .c al no poderse encontrar 3 librerías por no saber su ruta de directorio. Se buscaron con Windows ®, siendo:
         (1) d:\matlab6p5\include\bin\tmwtypes.h
         (2) d:\matlab6p5\simulink\c\simstruct_types.h
         (3) d:\matlab6p5\rtw\include\c\rt_logging.h.
      ii) Añadimos estas tres librerías al proyecto.
   d) Compilamos todos los archivos del proyecto: sin errores; aparece un aviso.
      i) El fichero "untitled.c" posee 1244 líneas de código y 568, de datos.

4) **Objetivo:** estudio de untitled.mat en Matlab ®.
   a) Se carga en el espacio de trabajo el fichero untitled.mat.
   b) Se renombran las variables por comodidad:
      i) t = rt_tout; % rt_tout = real time _ t out
      ii) y = rt_yout; % rt_yout = real time _ y out
   c) mediante plot(t,y) aparece la gráfica típica de la respuesta al escalón de un sistema de primer orden en discreto, tal y como había predicho Simulink ®.

Apunte 21:

25.4.2003:

1) **Objetivo:** descargar el código a la placa y depurarlo con CodeWarrior.
   a) Problema: tras la descarga e iniciación de la ejecución, el depurador muestra en mensaje: "Processor ARM_1 raised an exception. Cause: the processor was reset".
   b) Solución planteada: indicar al RTW® que se trata de un sistema imbuido (embedded) para que no tome por defecto la configuración PCWIN.

- Abriendo Matlab → Simulink → Open → untitled.mdl y dentro de la ventana del modelo: Tools → RTW → Options → Browse (System Target File) se cambia:
  - grt.tlc (generic Real Time) por ert.tlc (RTW Embedded Coder).
  - Ruta de ert.tlc: ...Matlab6p5\rtw\c\ert\ert.tlc.
- En "Category" escogemos:
  - ERT code generation options (1,2 y 3): se definen las opciones.
  - Se toma ANSI_C como estándar para flotante en lugar de ISO_C (por compatibilidad con el micro de ATMEL).
  - Se solicita la generación del reportaje en HTML.
  - Objetivo (target) : "bareboard" (placa simple).

   c) Para el diagrama de bloques untitled.mdl se genera el código C++ obteniéndose, según el reportaje HTML, para la ruta ...matlab6p5\work\untitled_ert_rtw los ficheros:
- ert_main.c; untitled.c; untitled_data.c;
- untitled.h; untitled_private.h; untitled_types.h;

   d) Estableciéndose la dependencia estática con las siguientes librerías de Matlab®:
- ...matlab6p5\extern\include\tmwtypes.h;
- ...matlab6p5\simulink\include\simstruc_types.h;
- ...matlab6p5\rtw\c\libsrc\rt_logging.h;

   e) Definimos un nuevo proyecto en CodeWarrior: ...\rtw\ertPrueba1; en él incluimos todos los archivos generados por RTW® Embedded Coder® y las librerías con las que existe dependencia estática.
   f) Make: mensaje con tres **errores** porque no existe la definición de los símbolos:
- rt_UpdateTXYLogVars.
- rt_StartDataLogging.
- rt_StopDataLogging.

   g) **Solución:** un sistema imbuido por el de ARM no posee dispositivos de almacenamiento en los que guardar los resultados del búffer tras la simulación.
      i) **Nota:** RTW® generó un ejecutable (...\work\entitled.exe) asociado a la compilación y construcción con RTW® Embedded Coder ® (por defecto usa el compilador para PcWin).
      ii) Lo ejecutamos y genera "untitled.mat" como resultado.
   h) Guardamos el modelo como ...\work\ertPrueba1.mdl.
   i) Construimos el código C++ con RTW® que se guarda en:...\work\ertPrueba1_ert_rtw.
      i) En las opciones de RTW® se inhabilita la generación de un historial con el formato de un archivo para Matlab® (Mat – File Logging).
      ii) Al ejecutar el ejecutable (...\work\ertPrueba1_ert_rtw\ertPrueba1.exe) se indica por la salida estándar que se programa simula en tiempo infinito.
   j) Guardamos el modelo Simulink® como "ertPrueba2.mdl" inhabilitada la opción del fichero de historial para compilar con CodeWarrior.

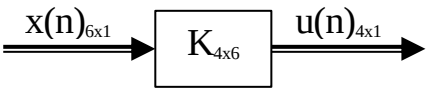
k) En un nuevo proyecto para CodeWarrior (ertPrueba2.mcp) se añaden los ficheros .c y .h, así como las dependencias estáticas, se compila sin errores y se depura.
l) En AXD Debugger se descarga (modo USER) y se ejecuta: aparece el mismo mensaje de ejecución indefinida, aunque no se resetea el micro como antes.
m) En Matlab®: pretendemos usar "External Interface Control Panel" para comunicarnos con la placa desde Matlab®, pero la placa no responde o no soporta este funcionamiento (quizás el archivo de interfaz C-Mex no esté definido) ➔ trabajaremos con CodeWarrior como interfaz con la placa (por ahora).

2) **Objetivo:** depuramos el código de "ertPrueba2" con AXD Debugger para incluir el código "printf("Valor de salida: ....);".
   a) **Error:** en la función "rtmGetErrorStatus(ertPrueba2.M)" no se puede entrar ni siquiera usado "Step in".
   b) **Solución:** probamos con "ertPrueba1", pero ni siquiera compila porque está activada la opción que genera el historial para Matlab® -símbolos indefinidos-:
      i) Buscamos esos símbolos en el contenido de todos los ficheros que cuelgan de ...matlab6p5\..., pero no se encuentran en ninguno.
      ii) Buscamos en la ayuda de Matlab®:
         (1) "rt_UpdateTXYLogVars" contenido en "Signal Monitoring via Block Outputs".
         (2) "rt_StartDataLogging" y "rt_StopDataLogging" no se encuentran.
      iii) Se trata de variables que están definidos dentro de la estructura del modelo, pero que sólo se usan cuando se genera un fichero de historial. No queda más remedio que usar "watchers" dentro del depurador AXD e inhabilitar la opción de generación del fichero de historial.
   c) **Problema:** no se puede encontrar fácilmente cuáles son los nombres de las variables que da Matlab® en el desarrollo del código C++.

3) **Modelo de Trabajo:** RTW® Embedded Coder ®:
   a) con .mat logging: cuando la simulación del diagrama Simulink ® genere ejecutable en PC.
   b) sin .mat logging: cuando la aplicación resida en la placa de ATMEL.
      i) Modificación del código para observar la simulación:
         (1) Comunicación con la Consola.
         (2) Comunicación vía bus CAN.

28.4.2003:

1) **Objetivo:** desarrollo del bucle de ganancias de realimentación con Simulink ®:
   a) Diagrama de Bloques: ...\work\gainMatrix.mdl.



2) **Objetivo:** observar y analizar los archivos .tlc para definir la configuración del target para ARM7TDMI de ATMEL con el compilador Metrowerks CodeWarrior.
   a) Especificar como compilador CodeWarrior.
3) **Objetivo:** Depurar ejemplo sencillo de Simulink® bajo CodeWarrior para observar con "watchers" el avance correcto de la simulación.
4) **Nota:** Objetivos 2) y 3) definidos tras charlar con Ismael (Tutor).

- Archivos .tlc**:** ...matlab6p5\toolbox\rtw\targets\mpc555dk\mpc555dk:
  - **mpc555_settings.tlc:** Define la configuración para el objetivo MPC555.

- Muchos de los campos que dicen dónde se sitúa el compilador no están definidos (definidos como: “”; cadena vacía de caractéres) → permite su definición como ruta de directorio.
- Campos de manejo del puerto Comm para la comunicación con la placa desde el depurador que aporta Simulink®.

- **mcp555rt.tlc:**
  - Incluye las librerías “mpc555_settings.tlc” y “mpc555rt_genfiles.tlc”
  - Define TargetCompiler = “diab”; es un compilador cruzado del mercado.
  - Define las opciones del compilador entre las sentencias “Begin_RTW_Options” y “End_RTW_Options”.
    - Estas opciones aparecen en el panel de opciones para RTW® de Simulink®.
    - Algunas están definidas, otras no.
- **mpc555pil.tlc:**
  - Incluye: “mpc555_settings.tlc” y “ert_pil_lib.tlc”, para importar la configuración global, “mpc555pil_genfiles.tlc, para construir objetivos mpc555pil (processor-in-the-loop).

a) **Nota:** los archivos .tlc de mpc555dk están todos en disquette.

30.4.2003:

1) **Objetivo:** Definir CodeWarrior® como compilador para ARM7TDMI dentro de los archivos que usa Matlab®. Existen archivos o carpetas que usan “diab”→ debemos estudiarlas para trabajar con CodeWarrior®.
   a) Copiamos la estructura de directorio de ...\targets\mpc555dk a ...\targets\ARM7TDMIdk para modificar todas las cadenas de texto “mpc555dk” como “ARM7TDMI” tanto en los nombres de los archivos como en su contenido.
   b) Toda entrada “diab” se cambia por “CodeWarrior” tanto en los nombres de los archivos como en su contenido.
   c) Ruta del compilador: “D:\SW_FCS\ARM\ADSv1_2\Bin\IDE.exe”.
   d) Ruta del depurador: “ D:\SW_FCS\ARM\ADSv1_2\Bin\axd.exe”.

Nota: toda la carpeta ARM7TDMIdk se ha comprimido .rar y pasado al PC233.

Apunte 22:

07.05.2003:

Objetivo: Analizar la documentación de RTW para definir un target[5] propio.

Información de Plantillas y Targets para RTW ® Embedded Coder ®.

- ert.tlc: archivo del sistema objetivo para RTW-EC[6].
- Plantillas existentes para diferentes entornos de desarrollo:
  - ert_bc.tmf: compilador Borland C.
  - ert_lcc.tmf: compilador LCC (propio del Matlab ®).
  - ert_vc.tmf: compilador Visual C.
- Rutas de directorio: ...\matlab6p5\rtw\c\ert

Estas plantillas contienen las macros para que las interprete el proceso make_rtw, es decir, el proceso que elabora los archivos objeto. Siendo las macros más representativas:

- MAKECMD: invoca la utilidad "make".
- HOST: indica la plataforma en la que reside "make".
- BUILD: invoca el proceso de construcción bajo RTW (yes/no).
- SYS_TARGET_FILE: nombre del archivo del sistema objetivo (ert.tlc).

Las preferencias del target son configurables mediante el comando "settargetprefs" que proporciona Matlab ®. Se puede especificar los siguientes campos:

- TargetCompiler: nombre del compilador cruzado.
- TargetCompilerPath: ruta de directorio de la carpeta que contiene el compilador.
- TargetDebuggerExe: ruta de directorio del depurador.
- TargetDebugger: modo de depuración.
- CommHostPort: puerto de comunicaciones del equipo anfitrión.
- CommTargetPort: puerto de comunicaciones del equipo objetivo.
- CommBaudRate: velocidad binaria de la comunicación serie entre los dos equipos.

Definimos las preferencias para el sistema "mpc555dk" que sí soporta Matlab® de la forma:

- TargetCompiler = 'CodeWarrior';
- TargetCompilerPath = 'd:\sw_fcs\arm\adsv1_2\bin';
- CommTimeout = '4';
- TargetDebugger = 'SingleStep';
- TargetDebuggerExe = 'd:\sw_fcs\arm\adsv1_2\bin\axd.exe';
- CommHostPort = 'COM2';
- CommTargetPort = 'COM1';
- CommBaudRate = '38400';

➔ ¡¡Se comprueba que no surge efecto sobre los ficheros del target MPC555DK!!

8.05.2003:

Objetivo: probar código C generado para mpc555 de un diagrama simple en Simulink®.

- mpc555dk queda configurado para Metrowerks® CodeWarrior® y AXD Debugger®.
- Placa conectada en modo USER.
- Generación del código:
  - nombre del fichero Simulink®: gananciaUnidad.mdl.
  - se genera el subdirectorio: ...\work\gananciaUnidad.
- Habilitamos las opciones "build" y escogemos mpc555PIL (Processor-in-the-Loop) como plantilla .tlc.
- PIL produce error.

---

[5] sistema HW objetivo en donde se ejecutará el código fuente

[6] Real Time Workshop ® Embedded Coder ®

- Probemos con mpc555(algorithm – exports) como plantilla .tlc.
  - Generación del Código.
  - Se inicia CodeWarrior, pero con el compilador apropiado para mpc555 que es "555Phytec Debug Version", el cual no nos sirve para el microprocesador de ATMEL.
- No funciona: no podemos engañar a Matlab®.
- SOLUCIÓN: generación de la plantilla .tlc adecuada para la máquina de ATMEL.

Búsqueda en la ayuda de Matlab®:

1) Tutorial: creating a Custom Target Configuration.

a) Crear un directorio en el que se guarden los archivos fuente C, .tlc (definición del objetivo) y .tmf (plantilla). Sea:d:\matlab6p5\work\arm7tdmi.

b) Añadirlo como ruta de directorio de Matlab®: en la consola de comandos de Matlab® se teclea addpath('d:\matlab6p5\work\arm7tdmi') .

c) Que este directorio sea el de trabajo de Matlab® puesto que éste se usa como el directorio destino del código C que genera RTW.

d) Copiar el ejemplo 'timestwo.c' desde:

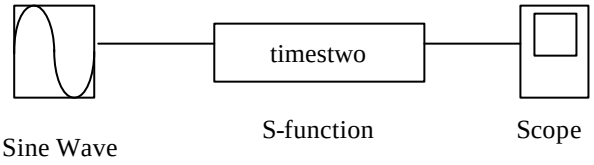
d:\matlab6p5\toolbox\rtw\rtwdemos\tlctutorial\timestwo\

En:

d:\matlab6p5\work\arm7tdmi\

e) mex timestwo.c Construir el archivo .mex de timestwo.c en el directorio destino, para ello se usa el comando: mex timestwo.c.

f) Crear un modelo de Simulink®



i) Definir una función S.
ii) Definir las opciones de Solver para RTW: fixed_type y ode(Runge Kutta).
iii) Nombre del modelo: ...\arm7tdmi\timestwo.mdl.

g) Definimos el nombre del bloque de la función S como "timestwo" de forma que quede ligado al código "timestwo.c".

h) Iniciar la simulación bajo Simulink®:
i) ERROR: el modelo y la función S poseen el mismo nombre.
ii) SOLUCIÓN: nuevo nombre del modelo: senoPor2.mdl.

i) Simulamos: se comprueba que la salida es 2 veces la entrada.

j) Se necesita el archivo .tlc correspondiente al .c para generar código interno (inlined); si no, se genera una llamada a una función S externa. Luego se copia timestwo.tlc en d:\matlab6p5\work\arm7tdmi desde la ruta de directorio d:\matlab6pd\toolbox\rtw\rtwdemos\tlctutorial\timestwo\.

k) Se copia d:\matlab6p5\rtw\c\grt\grt_main.c y d:\matlab6p5\rtw\c\grt\grt.tlc a d:\matlab6p5\work\arm7tdmi redenominándolos arm7tdmi_grt_main.c y arm7tdmi_ert_main.c.

l) Los comentarios de arm7tdmi_erc.tlc y arm7tdmi_grt.tlc sirven para añadir "arm7tdmi" al mostrados de los archivos de los objetivos (System Target File Browser).

m) El código generado por RTW se guarda en un subdirectorio cuyo nombre está formado por el del modelo junto con un sufijo. Para especificar un sufijo propio de nuevo target se usa "rtwgensettings.BuildDirSuffix = 'arm7tdmi_rtw'" dentro del archivo .tlc correspondiente.

n) D:\matlab6p5\rtw\c\grt contiene plantillas (.tmf) específicas para diferentes compiladores tanto para un objetivo en tiempo real genérico como para el caso de un sistema embeddido.
   i) Copiamos al ...\matlab6p5\work\arm7tdmi los .tmf: grt_lcc.tmf y ert_lcc.tmf; asociados al compilador LCC de Matlab®.
   ii) Los renombramos como arm7tdmi_grt.tmf y arm7tdmi_ert.tmf.
   iii) NOTA: los cambios siguientes son sólo aplicables al compilador LCC. Si se usa otro compilador, las reglas para la fuente (REQ_SRCS) y los archivos objeto (%.obj :) pueden diferir ligeramente.
o) Cambios en arm7tdmi_grt.tmf:
   i) SYS_TARGET_FILE = arm7tdmi_grt.tlc.
   ii) REQ_SRCS = $(MODEL).C $(MODULES) arm7tmdi_grt_main.c ...
   iii) %.obj : $(MATLAB_ROOT)/rtw/c/grt/%.c por
   %.obj : $(MATLAB_ROOT)/work/arm7tdmi/%.c
p) Cambios en arm7tdmi_ert.tmf:
   i) SYS_TARGET_FILE = arm7tdmi_ert.tlc.
   ii) REQ_SRCS = $(MODEL).C $(MODULES) arm7tmdi_ert_main.c ...
   iii) %.obj : $(MATLAB_ROOT)/rtw/c/ert/%.c por
   iv) %.obj : $(MATLAB_ROOT)/work/arm7tdmi/%.c

Apunte 23:

q) Abrir el cuadro de diálogo “Parámetros de Simulación” de RTW.
   i) En la opción “Browse...” aparecen 4 nuevas opciones:
      (1) arm7tdmi_ert.tlc: RTW Embedded Coder.
      (2) arm7tdmi_ert.tlc: Visual C/C++ Project Makefile only for the RTW Embedded Coder.
      (3) arm7tdmi_grt.tlc: Generic Real-Time Target.
      (4) arm7tdmi_grt.tlc: Visual C/C++ Project Makefile only for the “grt” target.
   ii) Definiciones:
      (1) System Target File: arm7tdmi_grt.tlc o arm7tdmi_ert.tlc.
      (2) Template Makefile: arm7tdmi_grt.tmf o arm7tdmi_ert.tmf.
      (3) Make Command: make_rtw.

r) Construcción: iniciamos el proceso “Build”.
   i) ERROR: se genera el código C colocándose en la carpeta ...\arm7tdmi\senoPor2_grt_rtw; por lo tanto el campo BuildDirSuffix de rtwgensettings no se ha aplicado correctamente.
   ii) SOLUCIÓN: especificamos este campo en el propio archivo .tlc, no desde la consola de comandos de Matlab®:
      (1) rtwgensettings.BuildDirSuffix = ‘_arm7tdmi_grt_rtw’ en el archivo ...\arm7tdmi\arm7tdmi_grt.tlc.
      (2) rtwgensettings.BuildDirSuffix = ‘_arm7tdmi_ert_rtw’ en el archivo ...\arm7tdmi\arm7tdmi_ert.tlc.
   iii) Iniciamos el proceso “Build”: se genera el código C en la carpeta definida ...\arm7tdmi\senoPor2_arm7tdmi_grt_rtw al seleccionar el en cuadro de diálogo ‘Parámetros de Simulación’ el caso de un sistema genérico de tiempo real. Para el caso embeddido se actúa análogamente, pero surge un error puesto que RTW Embedded Coder no soporta bloques continuos con son los del ejemplo “timestwo”.

s) Editamos el archivo: ...\arm7tdmi\senoPor2_arm7tdmi_grt_rtw\senoPor2.c para localizar la función “MdlOutputs”; en ella se observa el código añadido por la función S:

```
/* S-function block: <Root>/S-function*/
/* Multiply input by two */
rtB.S_function = rtB.Sine_Wave*2.0;
```

Es decir: al bloque ‘Sine_Wave’ ve multiplicada su salida por 2.0 y se asigna como salida del bloque ‘S_function’.

9.05.2003:

2) Adding a Custom Target to the System File Target Browser.

a) Modificar o añadir comentarios del examinador (Browser) en la cabecera del archivo del sistema objetivo:
   i) En arm7tdmi_ert.tlc:
      (1) %SYSTLC: RTW Embedded Coder for ATMEL ARM7TDMI.
   ii) En arm7tdmi_grt.tlc:
      (1) %SYSTLC: Generic Real-Time Target for ATMEL ARM7TDMI.

b) Generar un subdirectorio que contenga:
   i) archivo del sistema objetivo.
   ii) archivo ‘make’ de plantilla (.tmf).

iii) archivos del interfaz en tiempo de ejecución (programa principal y las funciones S).

Nota: debe estar este subdirectorio dentro del directorio raíz de Matlab®.

c) Añadir el camino: addpath <nombreDelDirectorio>.

➔ para que el camino se añada cada vez que Matlab® arranque se ha de incluir esta línea de comando en el fichero 'startup.m'.

3) **Objetivo:** Extender la configuración del compilador cruzado por defecto de Mpc555dk al compilador Metrowerks CodeWarrior® para arm7tdmi®.
   a) Los archivos .tmf contienen la información del compilador concreto.
   b) Estudiemos cómo se configura "Diab C Cross-Compiler" para Mpc555dk.
   c) Extendámoslo a los ficheros arm7tdmi_ert.tmf y arm7tdmi_grt.tmf.

- **Archivos originales:**

1) d:\matlab6p5\toolbox\rtw\targets\mpc555dk\mpc555dk\mpc555_ext.tmf.

Se usa para el modo "algorithm export": plantilla diseñada para construir una aplicación autónoma sobre PC mediante el generador de código C y el compilador "Diab Data Power PC 4.3g".

2) d:\matlab6p5\toolbox\rtw\targets\mpc555dk\mpc555dk\mpc555pil.tmf.

Plantilla para construir código embeddido con Diab C PC.

3) d:\matlab6p5\toolbox\rtw\targets\mpc555dk\mpc555dk\mpc555rt.tmf.

Plantilla para construir una aplicación autónoma para PC mediante Diab C PC.

➔ Estudiaremos el caso B para compararlo con el fichero arm7tdmi_ert.tmf.

- **Macros de interés:**
  - HOST: indica la configuración del sistema anfitrión. HOST = PC.
  - SYS_TARGET_FILE: indica el nombre del archivo .tlc.

Para mpc555pil.tmf esta macro se define como mpc555.tlc; así pues nosotros definimos, en el archivo arm7tdmi_ert.tmf, esta macro como arm7tdmi_ert.tlc.

- Estudiamos los archivos arm7tdmi_ert.tmf y arm7tdmi_grt.tmf:
  - En el apartado "tools specifications" se indica el compilador y su uso.
- Copia de seguridad de arm7tdmi_ert.tmf en arm7tdmi_ert.bak.
- Modificamos sobre arm7tdmi_ert.tmf la configuración del compilador para que, en lugar de usar LCC, se use CodeWarrior®.
  - PROBLEMA: no ofrece una solución factible por incomprensión de la nomenclatura.
- Observando mpc555rt.tlc:

%% import global settings
% include "mpc555_settings.tlc"
%%must be diab for RT Target at this time
% assign CompiledModel.Settings.TargetCompiler = "diab"

- En mpc555_settings.tlc:
- Se define la estructura CompiledModel:
  - CompiledModel:
    - Settings:
      - TargetCompiler.
      - TargetCompilerVersion.
      - TargetCompilerPath.
      - COM_BAUD.
      - CommTimeout.
      - COM_PORT.
      - CommTargetPort.
      - ...

    - PathInfo:
        - MODEL_FILE.
        - MODEL_REL_PATH.
        - ...
    - PlugIns:
        - MPC_555DK_PLUGIN_INCLUDES.
        - MPC_PLUGIN_LIBS.
- Asignación de valores a cada uno de los campos mediante "% assign ".
    - De la estructura Pathinfo
    - De la estructura Settings a partir de los campos predefinidos de mpc555_group.
    - De la estructura Plugins a partir de los campos de la estructura mpc555_group.
- Guardamos una copia de arm7tdmi_ert.tlc como arm7tdmi.bak y construimos una estructura análoga a la de mpc555rt.tlc y mpc555_settings.tlc.
- Modificaciones:

% assign TargetOS = "BareBoardExample"
...
%%import global settings:
% include "arm7tdmi_settings.tlc"
%% must be CodeWarrior for ARM Target at this time
% assign CompiledModel.Settings.TargetCompiler = "CodeWarrior".

- Copiamos mpc555_settings.tlc en ...\arm7tdmi_settings.tlc.
- Definimos los campos de las estructuras para CodeWarrior – Angel:

% assign CompiledModel.Settings.CWDEFINE_PROC[7] = "7tdmi"
% assign CompiledModel.Settings.TargetCompiler = "CodeWarrior"
% assign CompiledModel.Settings.TargetCompilerPath = "d:\sw_fcs\armm\adsv1_2\bin\ide.exe"
% assign CompiledModel.Settings.BSP_Target = "DebugRel"

→ Estaba definido como "phyCore-555", que es el compilador cruzado apropiado para mpc555 que implementa CodeWarrior. "DebugRel" es el correspondiente a arm7tdmi.

% assign CompiledModel.Settings.COM_BAUD = 38400
% assign CompiledModel.Settings.CommTimeout = 4
% assign CompiledModel.Settings.COM_PORT = COM2
% assign CompiledModel.Settings.CommTargetPort = "COM2"

Modificamos el diálogo de aviso (warndlg) para que sea coherente con arm7tdmi.
En los campos BSP_DIR y BSP_TLC_DIR de PathInfo se indica la ruta de directorio de los compiladores adecuados para ARM CodeWarrior y Tlc Matlab®, respectivamente.
→ No sé cómo se inicializan por el complejo mapa de objetos y el desconocimiento de la notación de la sintaxis: las dejamos como si fueran comentarios.
Guardamos el archivo como .old.

- Al compilar con RTW: error en la línea 72: el comando "%" no existe.

Línea 72: % the prefsGroup...
Línea 73: % assign ARM7TDMI_GROUP = FEVAL("gettargetprefs","mpc555dk","structure")
➔ Modificamos la línea 72 como sigue: %% the prefsGroup...
➔ Usamos "mpc555dk" para que ARM7TDMI_GROUP se inicialice con la configuración de mpc555dk, la cual se modificó para CodeWarrior.

- Al compilar con RTW: error en la línea 120: PIL_TLC_DIR no definido.

[7] codewarrior define processor

- Existe una referencia a PIL_TLC_DIR antes de que sea usada, siendo ésta:

% assign CompiledModel.PathInfo.PIL_TLC_DIR = “%<TARGET_ROOT>%<PATH_SEP>pil”

- Existe una referencia anterior a TARGET_ROOT:

% assign CompiledModel.PathInfo.TARGET_ROOT = FEVAL(“filepart”,”%<TLC_DIR>”,”path”)

Con la evaluación de la función “filepart” se consigue obtener el camino (“path”) del archivo especificado con “%<TLC_DIR>”.

- Existe una referencia anterior a PATH_SEP:

% assign CompiledModel.PathInfo.TLC_DIR = FEVAL(“filepart”,”%<TLC_FILE>”,”path”)

- Existe una referencia anterior a TLC_FILE:

% assign CompiledModel.PathInfo.TLC_FILE = FEVAL(“strtokn”,”%<TLCFILES>”,”[,]”,2)

➔ Pero TLCFILES no se define en arm7tdmi_settings.tlc. Veamos en la ayuda de Matlab®:

Se encuentra la entrada “TLC Error Messages” contenida en “TLC Error Handling”. En ella se define cómo se usa la función FEVAL, entre otras cuestiones, pero no ofrece una solución.

➔ Posible Solución: definir TLCFILES como es archivo .tlc.

% assign CompiledModel.PathInfo.TLCFILES = “arm7tdmi_ert”

- **Error** de Compilado en la línea 122 porque PIL_TLC_DIR no está definido.

% assign CompiledModel.PathInfo.PIL_TLC_DIR = “%<TARGET_ROOT>%<PATH_SEP>pil”

- Posible causa del error: no existe subdirectorio ..\arm7tdmi\pil.
- **Solución:** copiar el subdirectorio ..\mpc555dk\pil en ...\arm7tdmi manteniendo el archivo arm7tdmi_settings.tlc.
- **Error** en la línea 122: PATH_SEP y PIL_TLC_DIR no están definidos.
- **Solución:** el modo PIL no se implementará en ARM7TDMI.