ARQUITECTURA DE SOFTWARE EMBEBIDA

Por Jordi Montero Ferrer (**Director Técnico**) y Andreu Sabé Cruixent (**Arquitecto de Software**)
SALICRU

Introducción

Durante mucho tiempo se ha considerado que la clave para reducir el tiempo de diseño en sistemas embebidos es empezar el proceso de diseño del software con anticipación.

Otra sería, en nuestra opinión, el uso de Arquitecturas de Software para Sistemas Embebidos: según diversos informes, un proyecto de software embebido tarda una media de 13,1 meses en terminarse si se empieza de la manera clásica, pero este tiempo se puede reducir drásticamente si se usa una Arquitectura Software para Sistemas Embebidos.

Aumente su competitividad mediante un menor tiempo de desarrollo y mayor fiabilidad

Actualmente, la aplicación de Arquitecturas de Software para Sistemas Embebidos se ha identificado como un área que no está siendo plenamente explorada en los desarrollos de productos con Sistemas Embebidos.

Esta solución además de satisfacer las necesidades del producto permite adecuarse a las capacidades de los procesadores normalmente usados por una empresa. Éstos son diversos aspectos que se pueden mejorar con la ayuda de las Arquitecturas de Software:

- Bibliotecas de aplicaciones de fácil uso.
- Bibliotecas de aplicaciones de fácil reutilización.
- Partes comunes muy probadas y libres de errores.
- Facilidad de adaptación a diferentes sistemas de Hardware.
- Facilidad de test ya que poseen Abstracción de Hardware.
- Posibilidad de funcionamiento en plataformas Hardware con múltiples núcleos.
- Mayor facilidad de incorporación de recursos humanos a los equipos de desarrollo.
- Gran transversabilidad entre diferentes productos que usan la misma ESSA.
- Herramientas software/métodos comunes para los procesos de Producción y Servicio Postventa.
- Mayor coherencia entre los diferentes productos de la empresa.
- Facilidad de documentación.

Introducción a las ESSA

Típicamente, lo primero que se tiene en cuenta al inicio de un nuevo proyecto es la reutilización del Hardware y Software de un proyecto previo pero por lo que respecta a este último, este tipo de enfoque tiene la siguientes desventajas:

- El grado de reusabilidad podría ser más alto.
- El código fuente reutilizado no es suficientemente maduro ya que ha sido readaptado (no utilizado en su forma original).
- El código fuente resultante no es óptimo.
- El código resultante readaptado no debería ser susceptible de otra readaptación para un nuevo proyecto.
- La calidad del código fuente se ve rebajada.

En cambio, una Arquitectura para Sistemas Embebidos (ESSA de ahora en adelante) representa un beneficio a medio plazo para una empresa cuyos productos están basados en microprocesadores/procesadores.

Conceptos básicos de las ESSA

La variedad de factores que han de tenerse en cuenta en una ESSA son no solo las necesidades del sistema durante su etapa de desarrollo sino también durante los procesos de Producción y Servicio Postventa, ya que el objetivo de una ESSA es obtener una solución completa donde todos los actores antes enumerados salgan beneficiados.

Una vez están claros los objetivos y cuáles son los recursos disponibles en el procesador hay que considerar una serie de aspectos tales como Tiempo Real, Utilización de Memoria, Tipos de Datos, Almacenamiento, Sistema de Interrupciones, Temporizadores, Interfaz Shell, etc.

Beneficios aportados por las ESSA

Existen varias ESSA comerciales listas para su uso y algunas de ellas incorporan el Hardware, la documentación y ejemplos de aplicación. Ésta supone una forma fácil de empezar y realizar la primera aproximación a una ESSA. Otra forma puede ser desarrollar una ESSA propia, lo cual representa una inversión a medio plazo y la ocupación de algunos recursos de carácter técnico durante algún tiempo.

Si la empresa dispone de un equipo de I+D limitado se recomienda la primera opción, pero como solución a medio/largo plazo la segunda opción representa una mejor adaptación a las necesidades de desarrollo del producto, facilita su fabricación así como los procedimientos del Servicio Postventa, lo que en definitiva es el mayor beneficio de las ESSA.

Sea como sea, independientemente de cual de las dos opciones se escoja, se obtendrá un desarrollo de Software Embebido basado en ESSA con todos los beneficios que ello aporta.

Partes constituyentes de las ESSA

En nuestra opinión, las partes más importantes a considerar en una ESSA son:

- Un Sistema Operativo capaz de ejecutar Tareas y tener un control de Tiempo.
- Uso de memoria RAM.
- Tipos de datos.
- Conjunto de Tareas.
- Capa de Abstracción Hardware.
- Drivers
- Interfaz Shell.
- Software de PC capaz de interactuar con la ESSA a través de la interfaz Shell.

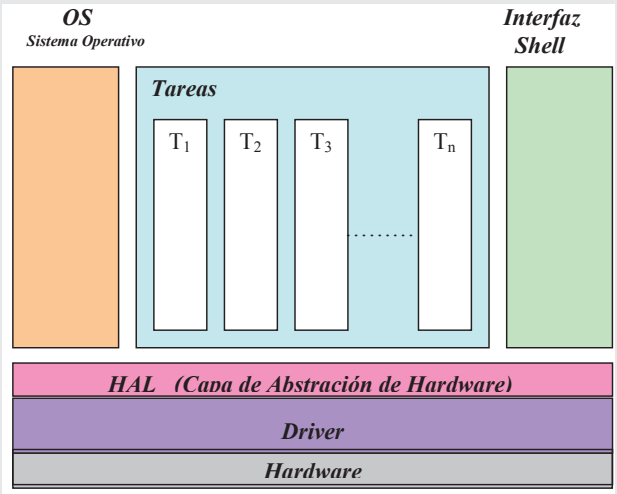

Figura 1: Ejemplo diagrama de bloques ESSA

El Sistema Operativo

Se considera que conjuntamente con la interfaz Shell y la Capa de Abstracción de Hardware (HAL), es la base que sustenta la ESSA. Su principal propósito es ejecutar las diferentes Tareas y proporcionar al sistema el grado de determinismo temporal necesario. Éste puede además dotar a la ESSA de un conjunto de recursos comunes tales como Control de Tiempo, Control de Eventos, Sistema de Ficheros, Control de Accesos, Reloj/Calendario, etc. Cualquier Tarea en la Aplicación puede demandar al Sistema Operativo (OS) el uso de uno de estos recursos y liberarlo cuando ya no lo necesite pudiendo entonces ser usado por otra Tarea.

Algunos OS también proporcionan capacidades de multitarea y en este sentido es importante determinar si ésta es una característica a considerar o si por el contrario su impacto supone sobredimensionar la ESSA o infrautilizarla.

Un aspecto clave es que como todas las capacidades y recursos arriba mencionados son parte del OS, cualquier desarrollo futuro que utilice la ESSA podrá servirse de ellos sin la necesidad de volverlos a implementar. Esto aportará al código un alto grado de reutilización y madurez.

En otro orden de cosas, el OS solo se relaciona con el Hardware en el momento de obtener la base de tiempos, lo cual se hace a través de una OSApi para enlazar el OS con el dispositivo físico. Esto facilita la portabilidad del sistema entre distintas plataformas Hardware.

El Uso de la Memoria RAM

Un factor a ser considerado en una ESSA es el uso de la memoria RAM. Aparte de la cantidad de memoria RAM que las diferentes Tareas necesitan para cumplir sus objetivos, las ESSA tienen también sus propias necesidades de memoria. Las ESSA están intrínsicamente basadas en capas software y por tanto usan más espacio de memoria de pila que los "Sistemas Clásicos". Además, dependiendo del tipo de OS, aún se necesita más memoria. Esto puede ser visto como un inconveniente, pero en nuestra opinión, si se escoge el tipo adecuado de OS, esto será compensado por los beneficios obtenidos. Otro factor que ayuda a mitigar este inconveniente es el hecho que diariamente aparecen en el mercado nuevos procesadores con una mejor relación precio/cantidad de RAM.

Tipos de datos

Las ESSA deben implementar los tipos de datos estándar y del OS pero pueden contar también con tipos de datos corporativos. Si los Sistemas Embebidos de la empresa usan un determinado tipo de dato, la ESSA debería incorporar este tipo de dato y sus métodos. Esto cumple con la política de las ESSA de reutilización del código y la consecuente maduración de éste.

El Nivel de Tareas

Independientemente de su complejidad, cualquier Software Embebido está dividido en varios módulos dependiendo de su función. Esta programación modular es, por supuesto, una práctica bien conocida que no solo facilita la implementación de la aplicación sino que también aporta beneficios en cuanto al mantenimiento del software y reusabilidad del código con un cierto número de modificaciones.

En una ESSA, los módulos pueden ser considerados como Tareas independientes y la práctica modular toma entonces aún más importancia. Aquí, escoger la modularidad correcta es primordial y por tanto hay que preguntarse:

- ¿Puede el módulo llegar a ser una parte permanente de la ESSA?
- ¿Que importancia tiene ser capaz de depurar el módulo?
- ¿Cuál es el nivel de interdependencia entre módulos?
- ¿Cuáles son las necesidades de tiempo del módulo?
- ¿Necesita ejecución en Tiempo Real?
- ¿Existe algún tipo de interacción entre el módulo y el Hardware?
- ¿Necesita el módulo de algún tipo de recursos del cual los otros módulos se puedan beneficiar?

Todas estas preguntas (y algunas más) deberían ser respondidas antes de decidir cual deberá ser el aspecto del conjunto del Aplicativo y como éste se relacionará con el OS y sus recursos.

En el entorno de una ESSA, las Tareas se pueden considerar como partes de software que se ejecutan de forma periódica y que no tienen relación directa entre ellas. Si la comunicación entre Tareas es necesaria, el sistema debe proporcionar una forma de hacerlo. Sea cual sea el sistema de comunicación, una regla debe tenerse en cuenta "Si se extrae un módulo (Tarea) del sistema, la compilación del software no debe fallar nunca". Si eso pasase, querría decir que hay algo en el sistema que es de alguna forma dependiente de ese módulo.

Para conseguir que el Aplicativo sea fácilmente adaptable a diferentes procesadores (en términos de Hardware), las Tareas deben evitar cualquier referencia directa al Hardware. Si se necesita interactuar con él, esto debe hacerse a través de la HAL, la cual debe proporcionar los métodos para hacerlo posible.

Al final, el objetivo de toda esta complejidad añadida es conseguir un sistema con los siguientes beneficios:

- Permitir al sistema parar/arrancar cualquier Tarea para facilitar la depuración del código.
- Facilitar la mejora/resolución de errores de una Tarea sin afectar a otras.
- Añadir fácilmente nuevas características al sistema añadiendo solo nuevas Tareas que realicen las nuevas funciones con una mínima modificación de éste.
- Permitir la comprobación de los Drivers/Hardware parando previamente la/s Tarea/s relacionadas con ellos y manejando entonces los Drivers/Hardware a través de la interfaz Shell.

El Nivel de Abstracción de Hardware (HAL)

Seria fácil decir que hace lo que su nombre indica, que es lo mismo que decir que independiza el Nivel de Tareas con respecto al Hardware. De todas formas, si se observa más de cerca se puede ver que la HAL se puede utilizar de otras maneras que son también beneficiosas. De hecho, la HAL no es solo el canal natural que las Tareas o el OS usan para interactuar con el Hardware del dispositivo a través de los Driver, sino que también puede mejorar las posibilidades de depurar las Tareas, los Drivers y hasta comprobar el Hardware del procesador en la línea de producción. Todos estos beneficios se pueden conseguir si la ESSA está dotada de una Interfaz Shell capaz de interactuar con la HAL.

El Nivel de Drivers

El nivel de Drivers se encarga de permitir la interacción del sistema con el mundo exterior.

Como que los periféricos del procesador varían de un dispositivo a otro, los Drivers deben proporcionar un conjunto de procedimientos estándar para permitir a las Tareas y el OS interactuar con ellos a través de la HAL. Procedimientos estándar como inicializar, arrancar, parar, leer y escribir son muy comunes pero se pueden añadir otros, como por ejemplo leer estado, si se necesita conocer el estado del periférico.

Además, otorgando al OS el control del arranque y parada de los Drivers, se añade la capacidad de comprobar el comportamiento de las Tareas. Esto se puede llevar a cabo parando los Drivers que interactúan con la Tarea y emulando el funcionamiento de los Drivers/Hardware a través de la interfaz Shell.

Finalmente, el Nivel de Drivers es el único que cambia de una plataforma a otra, pero si el número de dispositivos diferentes normalmente usados es limitado, se puede desarrollar el mismo número de conjuntos de Drivers que pueden ser empleados cada vez que se use la misma plataforma.

La Interfaz Shell

La Interfaz Shell (Shell) proporciona grandes beneficios en el ámbito de la Fabricación y el Servicio Postventa. La Shell dota a la ESSA de un canal donde se puede obtener información del sistema o ejecutar diversos comandos para configurar y comprobar la Aplicación. Los más habituales son:

Readregister	WriteRegister	ReadRegisterType
HALdisconnect	HALConnect	Login
StopApplications	StartApplications	HaltApplications

A parte de estos, se recomienda extender las posibilidades de la Shell con los siguientes comandos:

StopSingleTask	StartSingleTask	HaltSingleTask
HALSingleConnect	HALSingleDisconnect	ShowMeTask
ShowMeHAL	ReadMinRegValue	ReadMaxRegValue
GiveMeLastLog	FlushLog	whoami
Boot	unboot	Hi
ProductIdentification		

Teniendo en cuenta este conjunto de comandos, es fácil ver que la Shell puede resultar difícil de usar, por lo que se recomienda la utilización de un Software de PC como interprete de los comandos para la depuración y comprobación del sistema.

Una vez más, como la Shell pertenece a la ESSA, el índice de reusabilidad y madurez de su código es alto.

Finalmente, añadir que la salida física del la Shell hacia el mundo real puede variar desde una simple interfaz RS-232 hasta un puerto USB.

La ESSA y las Aplicaciones Compatibles para PC

Considerando lo dicho en el apartado anterior, queda clara la complejidad de interactuar con la ESSA a través de la línea de comandos. De ahí que se recomiende utilizar una Aplicación de PC compatible con ella.

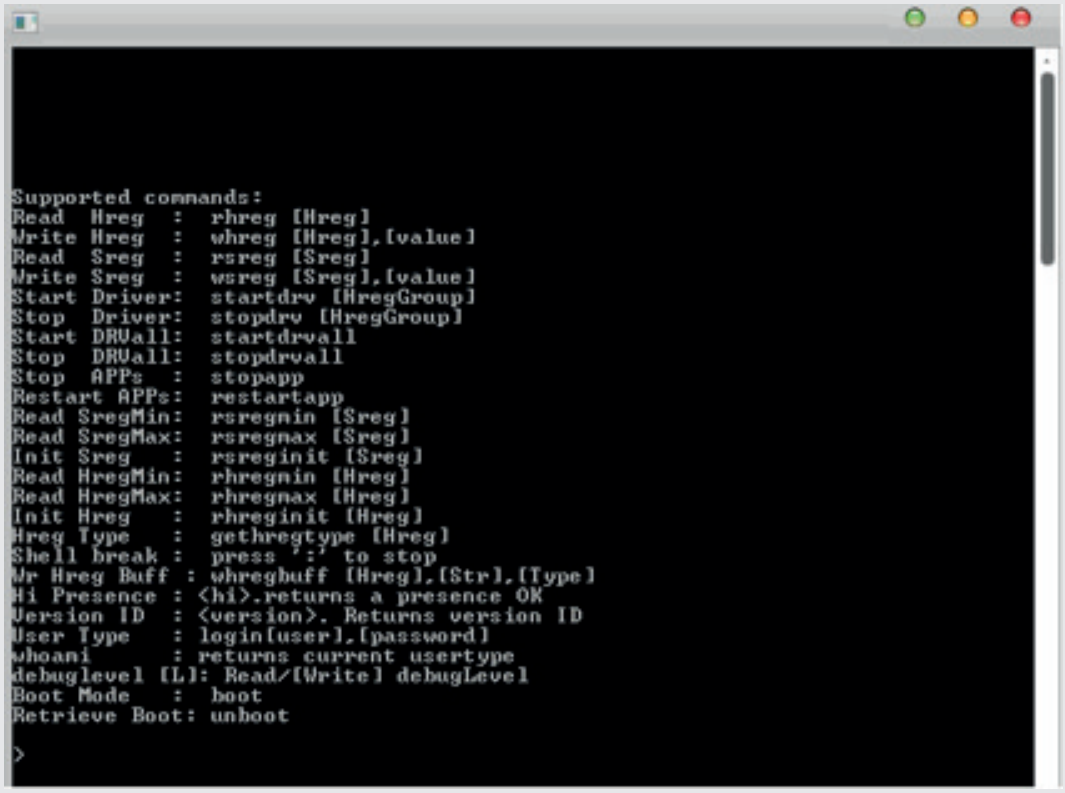

Figure 2: Ejemplo de comandos Shell

En este sentido, hay diversas aplicaciones disponibles en el mercado, pero para sacarle todo el partido, se recomienda nuevamente implementar una aplicación propia y personalizarla con el objeto de satisfacer las necesidades de Producción/Servicio Postventa. Esta Aplicación debería ser una herramienta de fácil manejo para interactuar con la ESSA con los siguientes propósitos:

- Comprobar fácilmente los valores de los datos internos y detectar errores.
- Configurar el producto de forma fácil.
- Facilitar/Automatizar la copia de seguridad de la configuración del producto.
- Facilitar/Automatizar el clonado de la configuración del producto.
- Compartir la configuración del producto de forma fácil (I+D / Fabricación / Servicio Postventa)
- Manipular y modificar fácilmente la configuración del producto en base a algunos parámetros. Ejemplo: Modificar un parámetro interno dependiendo del Número de Serie (Típica demanda de los Servicios Postventa).
- Almacenar de forma fácil la configuración del producto en la base de datos de la empresa.

Es importante mencionar que esta aplicación de PC será la misma para todos los productos basados en la misma ESSA. Si la aplicación es capaz de detectar a qué tipo producto está conectado, resulta muy fácil reconfigurarla para adaptarla a las características de éste.

Las ESSA: una puerta abierta a la IoT y la IIoT

La interfaz Shell es por definición la puerta natural de acceso de la ESSA al mundo exterior y viceversa. Si este acceso se protege de forma adecuada, los productos basados en esta arquitectura se convierten fácilmente en dispositivos IoT (del inglés, Internet of Things) y, por tanto, conectables a la gran red de redes, con múltiples posibilidades; entre ellas:

· Integración del producto en la red de datos del cliente.
· Monitorización remota de los equipos basados en la ESSA.
· Tele gestión de estos.
· Contribución a su mantenimiento predictivo.
· Mejora del servicio al cliente final.

Por otra parte, si hablamos de IoT, hay que referirse inevitablemente a su translación en el campo de la producción industrial, el cual que se ha venido a denominar Industrial Internet of Things (IIoT), y que está fuertemente relacionada con la nueva Revolución Industrial 4.0 de la que tanto se habla.

Los dispositivos basados en una ESSA se pueden conectar a la red de datos de la propia cadena de producción y colaborar con ella durante su proceso fabricación, simplificándolo y aportando una serie de datos que dotarán al producto de un mayor grado de trazabilidad y que además servirán para mejorar su proceso productivo.