Comentario Técnico: CTC-076
Título: **Utilización de I/O en ARM mediante CMSIS**
Autor: Sergio Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	30/10/15	

En este comentario técnico describimos brevemente el entorno CMSIS y la forma de utilización en I/O para varios procesadores ARM Cortex-M3.

CMSIS
Se trata de un conjunto de reglas y un grupo de archivos que conforman un nivel de abstracción que permite acceder a micros de diferentes fabricantes de una forma común; siempre que utilicen un core de la serie Cortex-M, es decir, Cortex-M3, Cortex-M4, Cortex-M0, Cortex-M0+, Cortex-M7[1]. Este tipo de interfaz independiente del dispositivo se basa fundamentalmente en proveer un punto de acceso abstracto para que ya sea kernels de RTOSes, aplicaciones de usuario, o incluso middleware; puedan operar sobre cualquiera de estos micros en particular, prácticamente sin cambios. Las reglas de programación aseguran que las interfaces provistas por diferentes fabricantes tengan una filosofía y una forma común, de modo de poder cambiar de uno a otro sin mayores inconvenientes. El usuario gana así la libertad de poder migrar de fabricante, tanto para el micro como para el compilador (y a partir de CMSIS 3.0 también de RTOS), con una mínima intervención: las diferencias naturales entre periféricos, minimizadas por la estructura y lineamientos dados por ARM.

En la versión 1, ARM proveía todo el paquete completo de soporte, unificando la información de los diferentes fabricantes de micros y compiladores en un único archivo. A partir de la versión 2, ARM provee solamente los archivos necesarios para acceder al NVIC y para poder utilizar las características distintivas del micro con diversos compiladores. El soporte para cada micro en particular es provisto por cada fabricante, ya sea del micro y/o del compilador. La versión 3 remueve algunas inconsistencias que se arrastraban desde la versión 1.

Gracias a las definiciones y nomenclatura que reglamenta CMSIS, cada fabricante provee un conjunto de archivos que permite acceder a los periféricos del micro de modo uniforme mediante estructuras en C. Los elementos de dichas estructuras se acceden de forma indexada, de modo de aprovechar el direccionamiento óptimo de ARM, y los punteros definidos respetan una nomenclatura del tipo

<abreviatura del micro>_<nombre del periférico>

Así, para escribir en el primer puerto de I/O de un Toshiba escribimos:

```
TOS_PA->DATA = byte;
```

en un Spansion:

```
FM3_GPIO->PDOR1 = byte;
```

y en un Holtek:

```
HT_GPIOA->DOUTR = byte;
```

Cada periférico de cada fabricante puede tener su estructura particular, pero más allá de la nomenclatura, todos responden a una serie de lineamientos generales por lo que a la hora de migrar de un fabricante a otro no debemos reescribir todo sino que realizamos pequeños cambios arquitectónicos. Salvando las distancias, con los tres micros mencionados se ha desarrollado un ejemplo de manejo de un display color; pueden observarse las diferencias en el código revisando las distintas encarnaciones de la CAN-094[2].

Entornos de desarrollo (IDEs)

1 y los siguientes a la fecha de publicación de este documento...
2 que podés solicitar a tu vendedor.

El entorno de desarrollo en un Cortex-M no es una elección del fabricante, sino del desarrollador. Si bien existen algunos casos particulares que son todo lo contrario, siempre existe la posibilidad de recurrir a un proveedor no atado con el fabricante, disponer de software gratuito, o incluso hasta *open-source*.
Tal vez el compilador C más utilizado, aunque probablemente no el más conocido, sea *gcc*.
Eclipse es un entorno de desarrollo basado en Java, extensible mediante una serie de plugins. Actualmente es un proyecto más de la Eclipse Foundation, http://www.eclipse.org/, fundación sin fines de lucro soportada por sus miembros que desarrolla software open-source.
Lamentablemente, la mayoría de los entornos de desarrollo que vamos a detallar no tienen versiones para GNU/Linux, debiendo el entusiasta independiente utilizar el compilador gcc, motor de todo lo conocido sobre esta plataforma, con o sin Eclipse. Una excepción a esta regla es la *community edition* del compilador que provee ARM, que soporta toda la línea de procesadores.
A continuación describimos los más comúnmente empleados para el desarrollo con microcontroladores dentro del ámbito de análisis de este libro, entre los que soportan y son soportados por, la mayor cantidad de micros y características standard; sin quedar, como hemos dicho, atados a ningún fabricante en particular, ya sea del micro o del compilador.

Keil

La firma Keil es desde hace unos años "una empresa del grupo ARM". Como tal, el compilador es algo así como "el oficial", al menos para este entorno de microcontroladores. El entorno de desarrollo utilizado es uVision, el nombre del producto es ARM-MDK (Microcontroller Development Kit), del que puede obtenerse en página web una versión sin costo con límite de 32KB de código generado.
El soporte para CMSIS en esta herramienta es por defecto, no sólo para la compilación sino que los include files que provee el MDK son del formato utilizado y recomendado en CMSIS.

IAR Systems

La empresa sueca IAR Systems desarrolla el producto Embedded Workbench for ARM (EWARM), cuya versión sin costo con límite de 32KB de código generado puede ser descargada de su página web. Esta empresa tiene muchos años de trayectoria y ha provisto compiladores y entornos de desarrollo de excelente calidad para gran cantidad de micros.
El soporte para incluir los archivos de CMSIS provistos se encuentra a partir de la versión 6.2, y debe activarse manualmente en las opciones del proyecto. Esto sólo aplica a los archivos genéricos para el core, los include files de los dispositivos que provee IAR responden a su propio formato, que es diferente del utilizado en CMSIS.

CooCox

Dicho nombre hace referencia a **Co**operate **o**n **Cor**te**x**, un grupo de desarrolladores que provee un entorno de distribución gratuita, CoIDE, basado en Eclipse y un compilador externo también gratuito. Afortunadamente a pesar de estar basado en Eclipse es relativamente rápido, siendo órdenes de magnitud más lento para toda acción que las opciones tanto de Keil como de IAR.
Este grupo de desarrolladores provee además código para muchos periféricos, una capa de abstracción (CoX), y hasta un RTOS (CoOS), todo de forma gratuita. Es posible obtenerlo desde su página web.
El soporte para incluir los archivos de CMSIS provistos debe activarse manualmente en las opciones del proyecto. El formato de los include files de los dispositivos es el que generalmente proveen los fabricantes, compatible con CMSIS, similar al provisto por Keil pero sin el prefijo que identifica al fabricante del procesador (por ejemplo `HT_` para el caso de Holtek).

Leer un switch, prender un LED

El archivo de software que acompaña a este comentario provee ejemplos de ambas funciones para algunos micros con core ARM Cortex-M3 de las firmas Holtek, Toshiba y Spansion (ex Fujitsu), tanto operando de forma tradicional en paralelo sobre los I/O como utilizando bit banding. En el caso de Holtek, el detalle de utilización de bit banding se encuentra explicado en el CTC-074.

Si bien los ports de I/O tienen nombres distintos y arquitecturas distintas, todos realizan en esencia la misma función y se controlan de forma muy similar:

Inicialización de Toshiba, switch en PH1, LED en PI1

```
TSB_PH->CR &= ~(1<<1);          // PH1 defined as Input
TSB_PH->PUP |= (1<<1);          // Enable pull-up in PH1
TSB_PH->IE |= (1<<1);           // Enable input circuit in PH1
TSB_PI->CR |= (1<<1);           // PI1 defined as Output
TSB_PI->DATA |= (1<<1);         // LED off
```

Inicialización de Spansion FM3, switch en P6.8 (pull-up externo), LED en P3.8

```
FM3_GPIO->PFR6 &= ~(1<<8);      // P6.8 defined as I/O
FM3_GPIO->DDR6 &= ~(1<<8);      // P6.8 defined as Input
FM3_GPIO->PFR3 &= ~(1<<8);      // P3.8 defined as I/O
FM3_GPIO->DDR3 |= (1<<8);       // P3.8 defined as Output
FM3_GPIO->PDOR3 |= (1<<8);      // LED off
```

Inicialización de Holtek, switch en PB11, LED en PB12

```
HT_CKCU->APBCCR0 |= (1<<17);  // 17=PBEN, enable APB clocks for GPIOB
HT_GPIOB->DIRCR &= ~(1<<11);  // PB11 defined as Input
HT_GPIOB->INER |= (1<<11);    // PB11 input circuit enabled
HT_GPIOB->DIRCR |= (1<<12);   // PB12 defined as Output
HT_GPIOB->DOUTR |= (1<<12);   // PB12=1, LED off
```

Como pudimos observar, algunos poseen puertos de I/O y registros de control de periféricos de 8-bits, otros de 16-bits y otros de 32-bits.

Existe además un “driver” o biblioteca de funciones provista por cada fabricante para acceder a funciones de configuración y control. El inconveniente, desde una óptica ingenieril en hardware, es que nos aleja demasiado del hardware, y además no suele ser compatible con el standard de CMSIS y requiere la utilización de archivos propios del fabricante del micro en lugar de los del compilador, dificultando además la eventual migración a otro procesador en el futuro.

Apéndice: Bit banding

Tanto el área de RAM interna, como la de periféricos están internamente divididas en cuatro secciones:

1. Un sector de 1MB (*Zona de bit band)*
2. Otro de 31MB (lo que resta para sumar 32MB)
3. Un sector de 32MB (*Bit band alias)*
4. Otro de lo que resta para sumar 512MB

En estas secciones se incorpora un esquema denominado *de bit banding*, que permite direccionar cada bit por separado o dentro de un byte, half-word (16-bits), o word (32-bits). En este esquema, cada bit de la *Zona de bit band*, se corresponde con una palabra (32-bits) en la zona *Bit band alias* . De este modo, una lectura de un byte en la *Zona de bit band*, setear un bit y volver a escribir (operación típica *read-modify-write* requerida para realizar un OR), puede reemplazarse por una escritura en la zona *Bit band alias* en la palabra correspondiente a dicho bit (una palabra por cada bit), dado que existe una correspondencia biunívoca entre ambas zonas.
Esto resulta sumamente útil para trabajar con periféricos, es más eficiente, y elimina los clásicos inconvenientes del acceso múltiple y colisiones entre diversas tareas que comparten el uso bidireccional de los pines de un puerto de I/O.