Sistemas Empotrados II
Trabajo final de la asignatura

David Pintiel Paños - 755727

Enero de 2021

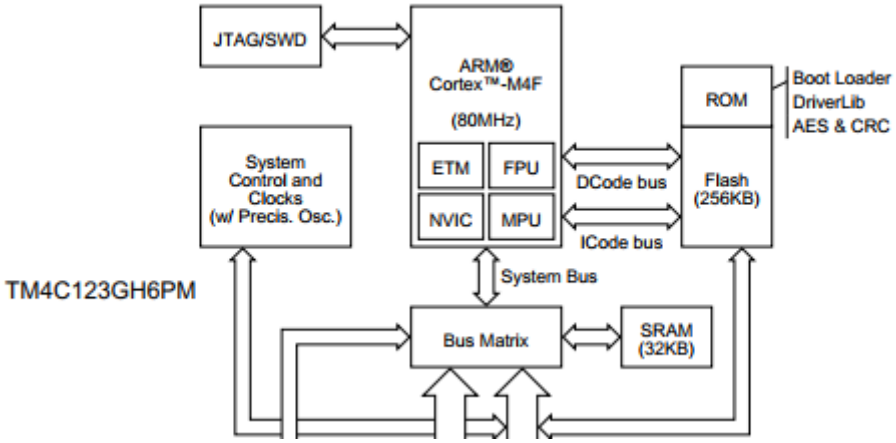
Contenido

Introducción

En este trabajo se va a realizar un estudio de la librería de drivers de periféricos (Peripheral Driver Library) de la familia de microcontroladores Tiva-C Launchpad de Texas Instruments, en concreto el TMC4123. Esta librería es una API para la gestión de los diferentes periféricos incorporados en cada microcontrolador de forma sencilla.

Los microcontroladores cuentan con diferentes memorias que presentan las siguientes características:

- Memoria RAM: memoria volátil y rápida. Se almacenan las zonas de memoria de datos (.bss, .data, .stack, .vecs).
- Memoria flash programable: Por defecto almacenan las secciones estáticas del programa (.text, .const, .cinit). No volátil.
- Memoria ROM: memoria preprogramada de solo lectura y no volátil. Almacena la librería de drivers de los periféricos, el bootloader, las tablas criptográficas del Advanced Encryption Standart (AES) y una implementación de CRC.



Como se ha mencionado, la ROM aloja la librería de drivers de periféricos, lo que es una característica particular. Sin embargo, puede haber otros microcontroladores de la familia que no tengan esta memoria ROM, por lo que se deberán almacenar en la memoria flash. Por tanto, se da la opción al programador de alojar los drivers en la memoria flash o hacer uso de los mismos en la ROM si están disponibles.

El objetivo de alojar dichos drivers en la ROM es ahorrar espacio en la memoria flash del dispositivo.

El objeto de estudio del presente trabajo es comparar el tiempo de ejecución de la librería de drivers alojada en ROM frente a estar almacenada en flash, así como el ahorro de espacio en memoria flash que supone el ubicar la librería en la memoria ROM.

Metodología

Uso de la librería de drivers de periféricos

Texas Instruments proporciona diferentes formas de acceder a los periféricos mediante las funciones de sus APIs. Para utilizar las funciones primero es necesario incluir sus respectivos ficheros de cabeceras, ubicados en el directorio driverlib.

> **Nota**: <nombre función> es el nombre de la función de la librería de un periférico determinado a usar.

- <nombre función >: se invoca a la función por defecto, que se alojará en flash. Para utilizarla basta incluir la cabecera del driver del periférico a utilizar.
- ROM_<nombre función>: se invoca explícitamente a una función mapeada en la memoria ROM. Si el dispositivo no tiene el respectivo driver alojado en la memoria o prescinde de la misma, saltará un error de compilación.
  Para utilizarlas es necesario incluir las cabeceras del driver del periférico, y el fichero driverlib/rom.h.
- MAP_<nombre función >: se invoca a la función en ROM en caso de estar disponible. En caso contrario, la función se habrá mapeado en flash y se accederá mediante esta memoria.
  Tiene como objetivo dar portabilidad al código entre diferentes microcontroladores de Texas Instruments. De este modo, siempre se accederá a las funciones en ROM si es que están disponibles en un determinado dispositivo.
  Para poder utilizar estas funciones es necesario incluir las cabeceras driverlib/rom.h, driverlib/rom_map.h y la correspondiente al periférico deseado.

Cabe destacar que en el fichero driverlib/rom.h hay una serie de cláusulas que determinarán en tiempo de compilación que librerías de periféricos podrán usarse en ROM según el dispositivo para el que se compile. Sin embargo, puede ser que haga falta añadir un define previo que defina el dispositivo para el que se usa. En este caso, ha sido necesario añadir la claúsula que define el dispositivo destino de compilación.

Ejecución en flash frente a ROM

Se va a evaluar que ventajas o desventajas tiene la ejecución en ROM frente a flash. Por el hecho de que una memoria ROM suele ser más lenta que las memorias flash en general, se puede intuir que el acceder a las funciones de librería en ROM añadirá penalización.

Se ha diseñado una prueba sencilla en la que se ejecutará un bucle un determinado número de veces y se hará la media de tiempo que ha tomado cada ejecución, con el objetivo de disminuir el error. En bucle se accederán a unas pocas funciones de librería repetidamente. Los códigos utilizados han sido los siguientes:

```
void task_FLASH(UArg a0, UArg a1){
    for(;;){
        Log_write1(UIABenchmark_start, (xdc_IArg)"WCET");
        GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_2,GPIO_PIN_2) ;
        int i = 0;
        for(i; i < 100000; i++){
               SysTickPeriodSet(0x1000*i);
               SysTickEnable();
               SysTickDisable();
           }
        GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_2,0) ;
        Log_write1(UIABenchmark_stop, (xdc_IArg)"WCET");
        Task_sleep(2);
    }
}

void task_ROM(UArg a0, UArg a1){

    for(;;){
        Log_write1(UIABenchmark_start, (xdc_IArg)"WCET");
        GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_2,GPIO_PIN_2) ;
        int i = 0;
        for(i; i < 100000; i++){
               ROM_SysTickPeriodSet(0x1000*i);
               ROM_SysTickEnable();
               ROM_SysTickDisable();
           }
        GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_2,0) ;
        Log_write1(UIABenchmark_stop, (xdc_IArg)"WCET");
        Task_sleep(2);
    }
}
```

Ahorro de memoria en flash

Para dilucidar el ahorro en memoria que supone alojar la librería en ROM frente a flash hay que tener en cuenta:

- Número de funciones utilizadas.
- Tamaño de cada función.

El fichero .map del proyecto en CCS muestra lo que ocupa cada parte del código. Si se utiliza una determinada función en flash, el tamaño de la memoria que ocupa la driverlib aumentará. Si se utiliza en ROM, se observa como disminuye. De hecho, el objeto de la librería deja de figurar en el fichero .map, ya que no requiere de mapeo puesto que ya están alojadas en ROM.

El espacio total posible por ahorrar no ha sido calculado. Para poder conocerlo, sería necesario utilizar todas las funciones de librería en el código, una tarea, en consideración propia, de proporciones bastante desmesuradas.

Resultados

Problemas en el uso de la librería de periféricos

Ha sido verdaderamente difícil conseguir ejecutar todos los tipos de funciones de manera precisa. Aunque en el presente trabajo se deja bastante claro como usar la driverlib desde ROM, es difícil usarla y verificar su ejecución con los manuales del fabricante. Texas Instruments proporciona mucha información en sus manuales sobre el uso de las APIs, pero es en general inconexa.

Además, hay que socavar información de varios manuales, en vez de tener centralizada en un punto concreto toda información referente al uso de la driverlib desde ROM. Esto complica una tarea que debería ser muy sencilla y sin problemas en un quebradero de cabeza. Si no se conoce muy bien el bosque, puede dar pie al programador a no entender porque se está haciendo mucho más uso del debido en la memoria flash por estar alojándose allí funciones que se creían ubicadas en ROM.

En experiencia personal, ha costado hacer funcionar las funciones MAP_<nombre función>. Por pequeños errores, las funciones se mapeaban en flash y no en ROM, dando pie a confusión y dudar del correcto funcionamiento de las APIs de TI.

Ahorro de memoria en FLASH

Las siguientes pruebas han sido realizadas con un nivel optimización en la compilación de 4 en CCS, y un balance entre espacio y velocidad de ejecución de 1.

En el código se han ido añadiendo funciones de librería y observando como aumenta en espacio ocupado en flash.

Número de funciones	**Espacio ocupado en flash (Bytes)**
7	1072
6	638
5	586
4	336

Teniendo en cuenta que hay cerca de 1800 funciones, el ahorro puede llegar a ser importante. Suponiendo que cada función de media ocupe 100 bytes, se estima un ahorro de 180 kB en caso de usar toda la driverlib. Teniendo en cuenta que hay disponibles 256 kB disponibles en memoria flash, es una cantidad muy relevante.

Tiempo de ejecución

Las pruebas realizadas han sido con un balance entre espacio y velocidad de ejecución de 1.

Los resultados obtenidos quedan reflejados en la siguiente tabla:

Alojamiento de los drivers	**ROM -O0 (ms)**	**FLASH -O0 (ms)**	**ROM -O3 (ms)**	**FLASH -O3 (ms)**
Tiempo de ejecución	437,634	334,683	347,548	283,213
SpeedUp	1,3		1,227	

Se puede afirmar que hay una mejora al acceder a la librería desde flash frente a ROM. No obstante, aunque la mejora sea sustancial, hay que recordar que la gran mayoría de funciones de librería son accedidas con baja frecuencia por lo general y en contexto de interrupción, lo que puede implicar que no sea muy relevante la mejora que hay. La decisión de alojar una función en flash o en ROM dependerá mucho del uso que se le quiera dar a la dicha función.

Conclusiones

Si se quiere llegar a obtener el mejor rendimiento posible, según las necesidades de la aplicación, se puede combinar ambos tipos de funciones. Por ejemplo, en periféricos que vayan a interrumpir con una frecuencia mucho más superior que el resto, podría ser interesante alojar sus funciones de librería en flash.

Por otro lado, puede que este cambio no sea suficientemente relevante: cualquier aplicación lo suficientemente compleja probablemente tenga cuellos de botella en otros puntos antes que en este. Además, existe el hecho de que puede premiar mucho antes el tener disponible el máximo espacio posible en flash antes que acceder a algunas funciones de forma un poco más rápida.

También esta la parte del uso de la librería, que verdaderamente es complicada de usar con la documentación aportada por el fabricante. Es complicado verificar que estás ejecutando sobre ROM, lleva varios quebraderos de cabeza hasta que conoces bien todo el ecosistema de Texas Instruments. Es bastante probable que se estén utilizando la funciones MAP y por algún pequeño fallo no ejecutarlas desde ROM aun estando disponibles, entre otros problemas.

Por último, alegar que el alojar la driverlib en ROM es una buena idea a priori, puede suponer un ahorro importante sin perjudicar apenas el rendimiento de la aplicación. Sin embargo, el hacer uso de driverlib desde ROM no es trivial, y la documentación del fabricante es poco clara y confusa, lo que dificulta una tarea sencilla que no debería suponer tiempo.

Anexos

Anexo 1: material utilizado

El material hardware y software utilizado para este estudio ha sido:

- El microcontrolador TMC4123. Dicho microcontrolador cuenta con la opción
- El IDE Code Composer Studio y sus herramientas, en especial la herramienta Duration para las mediciones de tiempos de ejecución.

Anexo 2: resultados de las pruebas

-Drivers en flash compilados en -O3

	Source	Count	Min	Max	Average	Total	Percent
1	CORTEX_M4_0, WCET	23	283212937	283213188	283,213,139.0	6,513,902,196	100.0

-Drivers en ROM compilados en -O3

	Source	Count	Min	Max	Average	Total	Percent
1	CORTEX_M4_0, WCET	23	347548500	347548875	347,548,695.2	7,993,619,990	100.0

-Drivers en ROM compilados en -O0

	Source	Count	Min	Max	Average	Total	Percent
1	CORTEX_M4_0, WCET	23	437634063	437634500	437,634,410.3	10,065,591,438	100.0

-Drivers en flash compilados en -O0

	Source	Count	Min	Max	Average	Total	Percent
1	CORTEX_M4_0, WCET	23	334683563	334683687	334,683,573.8	7,697,722,197	100.0

Espacio ocupado en flash de 7 funciones

```
../driverlib.lib
   sysctl.obj                     582     108       0
   gpio.obj                       434     0         0
   systick.obj                    56      0         0
+--+------------------------------+-------+---------+---------+
   Total:                         1072    108       0
```

Espacio ocupado en flash de 6 funciones

```
../driverlib.lib
   sysctl.obj                     582     108       0
   systick.obj                    56      0         0
+--+------------------------------+-------+---------+---------+
   Total:                         638     108       0
```

Espacio ocupado en flash de 5 funciones

```
../driverlib.lib
   sysctl.obj                     530     108       0
   systick.obj                    56      0         0
+--+------------------------------+-------+---------+---------+
   Total:                         586     108       0
```

Espacio ocupado en flash de 4 funciones

```
../driverlib.lib
   sysctl.obj                     280     108       0
   systick.obj                    56      0         0
+--+------------------------------+-------+---------+---------+
   Total:                         336     108       0
```

Anexo 3: Código completo de las pruebas

```
#include <stdbool.h>
#include <xdc/std.h>
#include <xdc/runtime/System.h>
#include <xdc/runtime/Log.h>
#include <ti/uia/events/UIABenchmark.h>
#include <ti/sysbios/BIOS.h>
#include <ti/sysbios/knl/Clock.h>
#include <ti/sysbios/knl/Task.h>

#include <xdc/runtime/Types.h>
#include "inc/hw_memmap.h"

/*Comentar o descomentar en función de lo que se quiera probar*/
//#define prueba_ROM
//#define prueba_FLASH
#define prueba_tamanyo

#ifdef prueba_ROM
#define TARGET_IS_TM4C123_RA3
#include "driverlib/rom.h"
#endif

#ifdef prueba_FLASH
#define TARGET_IS_TM4C123_RA3
#include "driverlib/rom.h"
#include "driverlib/rom_map.h"
#endif

#include "driverlib/adc.h"
#include "driverlib/aes.h"
#include "driverlib/can.h"
#include "driverlib/comp.h"
#include "driverlib/cpu.h"
#include "driverlib/crc.h"
#include "driverlib/des.h"
#include "driverlib/eeprom.h"
#include "driverlib/emac.h"

#include "driverlib/flash.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/systick.h"
#include "driverlib/sysctl.h"
#include "driverlib/shamd5.h"

/*Variables globales*/
Task_Handle task;

/*Fucniones de prueba*/
#ifdef prueba_ROM
void task_ROM(UArg a0, UArg a1){

    for(;;){
        Log_write1(UIABenchmark_start, (xdc_IArg)"WCET");
        GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_2,GPIO_PIN_2) ;
```

```
        int i = 0;
        for(i; i < 100000; i++){
               ROM_SysTickPeriodSet(0x1000*i);
               ROM_SysTickEnable();
               ROM_SysTickDisable();
           }
        GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_2,0) ;
        Log_write1(UIABenchmark_stop, (xdc_IArg)"WCET");
        Task_sleep(2);
    }
}
#endif

#ifdef prueba_FLASH
void task_FLASH(UArg a0, UArg a1){
    for(;;){
        Log_write1(UIABenchmark_start, (xdc_IArg)"WCET");
        GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_2,GPIO_PIN_2) ;
        int i = 0;
        for(i; i < 100000; i++){
               SysTickPeriodSet(0x1000*i);
               SysTickEnable();
               SysTickDisable();
           }
        GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_2,0) ;
        Log_write1(UIABenchmark_stop, (xdc_IArg)"WCET");
        Task_sleep(2);
    }
}
#endif

#ifdef prueba_tamanyo
void task_tamanyo(UArg a0, UArg a1){
    for(;;){
        SysTickPeriodSet(0x1000);
        SysTickEnable();
        SysTickDisable();
        SysCtlClockGet();
        //SysCtlClockSet(0x1000);
        //SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
        //GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, GPIO_PIN_2);

    }
}
#endif

int main(){

#ifdef prueba_FLASH
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    ROM_GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, GPIO_PIN_2);
    task = Task_create(task_FLASH, NULL, NULL);
#endif

#ifdef prueba_ROM
```

```
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    ROM_GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, GPIO_PIN_2);
    task = Task_create(task_ROM, NULL, NULL);
#endif

#ifdef prueba_tamanyo
    task = Task_create(task_tamanyo, NULL, NULL);
#endif

    BIOS_start();
}
```