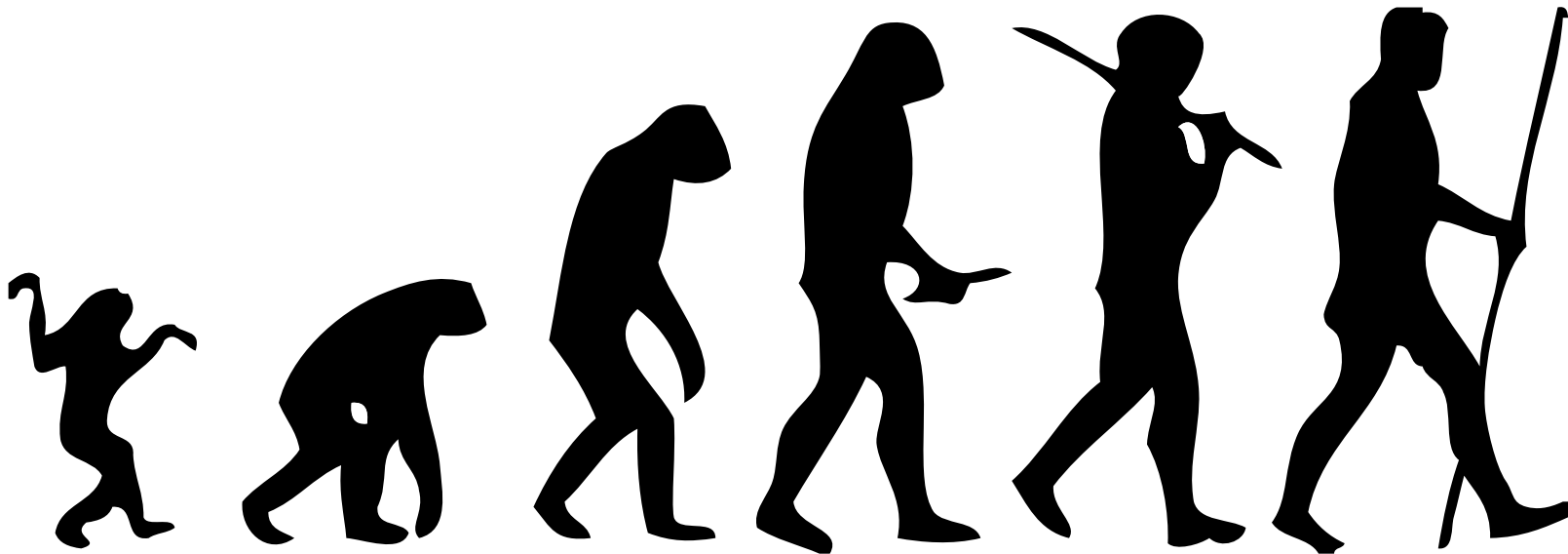




UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Conceptos básicos sobre software para microcontroladores ARM Cortex-M



2013/04/18



armcortexm.blogs.upv.es



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



etsinf



Contenido

- Objetivos
- El lenguaje C es lo adecuado
- El caos de las bibliotecas
- CMSIS: La solución
- CMSIS y el St STM32F(4)
- Proyectos
 - Creándolo con Keil MDK-ARM
 - A partir de una plantilla
- “Hola Mundo” (Al servicio de depuración)
- El arranque: del vacío al main()



Objetivos

- Entender la filosofía de desarrollo
- Tener nociones de CMSIS
- Aprender a crear un proyecto mínimo
- Aprender a usar plantillas y bibliotecas externas



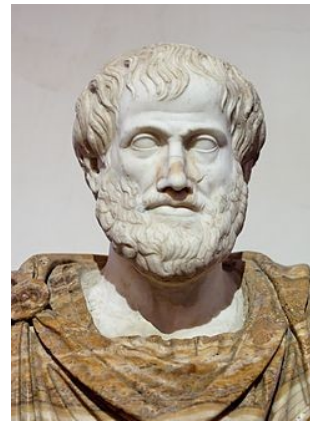
El lenguaje C es lo adecuado

- Esta arquitectura está pensada para desarrollo en lenguaje C
 - Usar ensamblador no tiene sentido (ni aquí ni en otros sitios)
 - (Un programador experto conoce ensamblador y le da ventaja respecto al resto de programadores)
- Hay otras opciones:
 - Pascal, Basic, C++, C#, java, Ada, ...
 - Son lenguajes marginales en el ámbito de los microcontroladores
- Como ingeniero, elige ...

Inglés

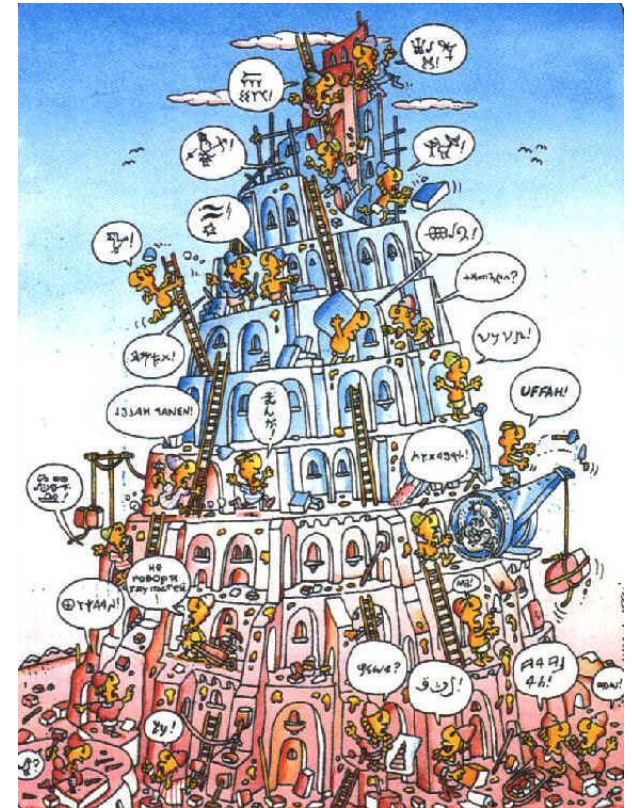


Griego



El caos de las bibliotecas

- Los microcontroladores ARM Cortex-M son complejos
- Primera solución
 - Cada fabricante crea sus bibliotecas C para simplificar
- Pega (bueno para unos, malo para otros)
 - Incompatibles entre fabricantes
 - Dificulta la migración entre fabricantes
- A ARM no le convenía esto y puso orden
 - Si te das cuenta, a ti tampoco te conviene



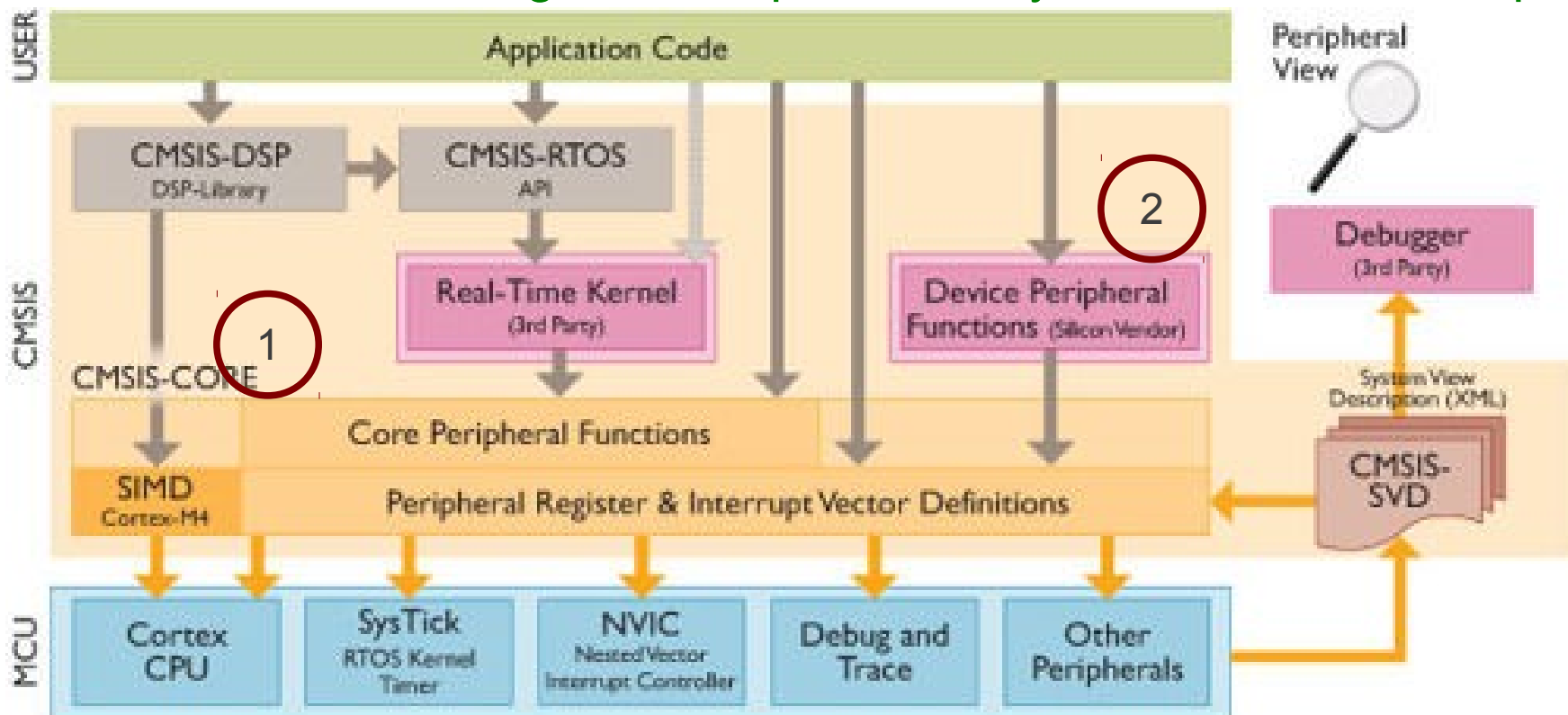
CMSIS: La solución

- CMSIS: Cortex Microcontroller Software Interface Standard
- Fruto de la coordinación entre ARM, fabricantes, desarrolladores de entornos, desarrolladores de bibliotecas, ...
- Pretende establecer abstracción del hardware común a todos los fabricantes de procesadores ARM Cortex-M



CMSIS: La solución

- Arquitectura de CMSIS (bueno, de CMSIS-Core)
 - Funcionalidades para arranque del sistema
 - Acceso características específicas del núcleo y periféricos básicos
 - Visión consistente registros de periféricos y servicios de interrupción



Nota: La imagen corresponde a CMSISv3



CMSIS: La solución

- ¿Recuerdas el ejemplo del LED?
 - Y 2 se apoya en 1

```
#include "stm32f4xx_rcc.h"  
#include "stm32f4xx_gpio.h"
```

```
void led_inicializar(void) {
```

```
    GPIO_InitTypeDef  GPIO_InitStructure;
```

```
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
```

```
    GPIO_InitStructure.GPIO_Pin    = GPIO_Pin_15;  
    GPIO_InitStructure.GPIO_Mode  = GPIO_Mode_OUT;  
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;  
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;  
    GPIO_InitStructure.GPIO_PuPd  = GPIO_PuPd_NOPULL;
```

```
    GPIO_Init(GPIOA, &GPIO_InitStructure);
```

```
}
```

1

2




CMSIS: La solución

- CMSIS se basa en el sentido común. Por ejemplo:
 - Cumple MISRA 2004 (Sí, busca en Google)

```
if (sonda_gamma > 45.3)
    CatalizadorOn();
else
    CatalizadorOff();
```

A Juan lo echaron de la Volkswagen por esto



- El código se debe documentar con doxygen (y el desarrollador se ahorra hacer el manual)

```
/**
 * @brief Calcular el azimut ortogonal
 * @param alfa la consistencia
 * @param delta el espesor
 * @returns pues eso, el azimut
 */
double AzimutOrtogonal(double alfa, double delta) {
```



CMSIS: La solución

- CMSIS se basa en el sentido común. Por ejemplo:
 - Emplea la definición de enteros <stdint.h>
 - ejemplo: un `int` es distinto en un PIC 12F, un Cortex-M o un x86
 - El siguiente ejemplo compila bien en un PIC, un 8051, un Cortex-M, un PC x86, una GPU Nvidia o un IBM Blue Gene/Q

```
#include <stdint.h>
```

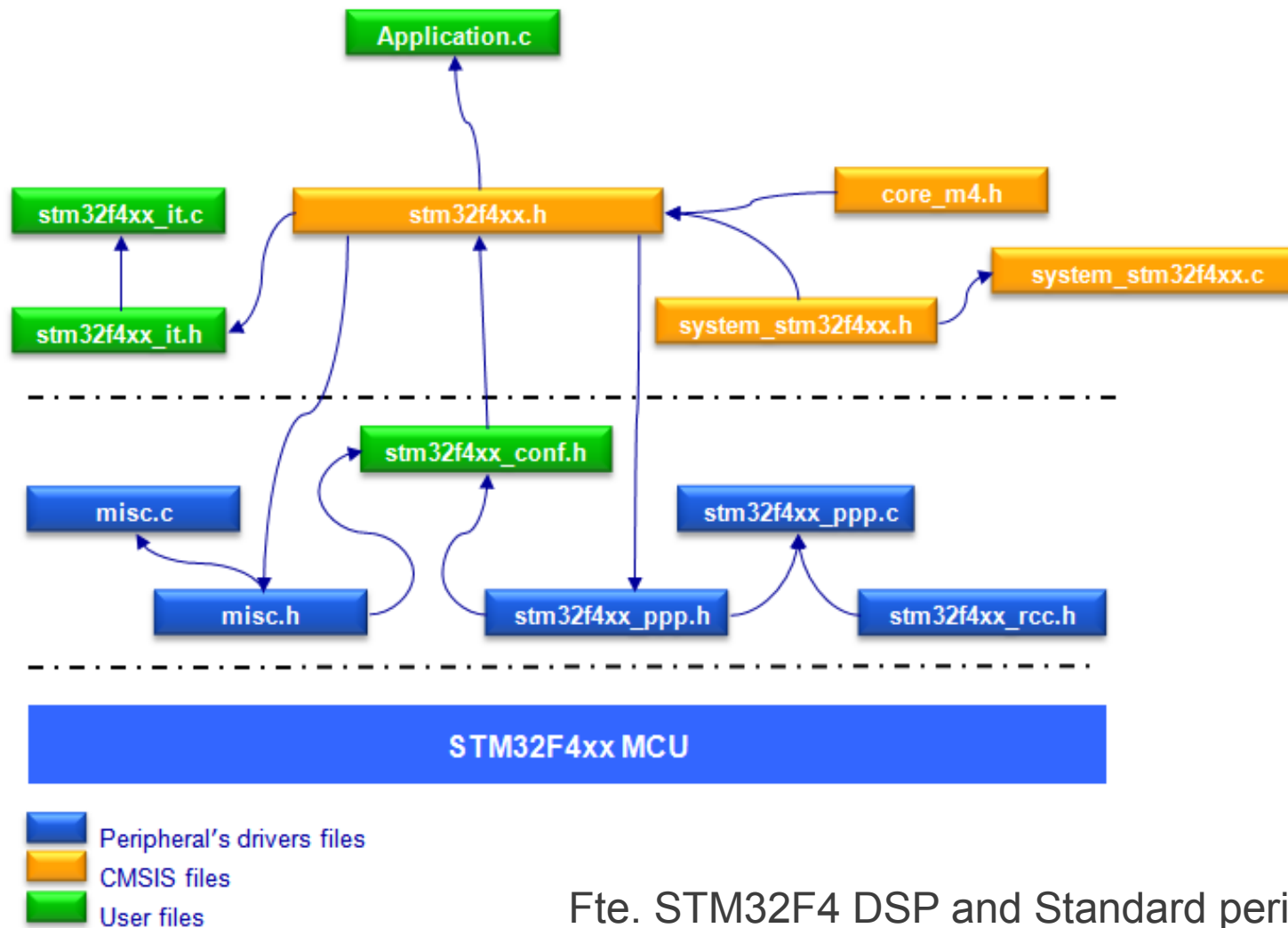
entero de 16 bits sin signo

```
uint8_t cds_DAQReadQuadratureIn(uint8_t subdevice, uint16_t *value) {  
  
    if (subdevice != SUBDEVICE_QUADRATURE_INPUT_1) {  
        return(CD_Error_DAQBadSubdevice); /* bad subdevice */  
    }  
  
    *value = TIM1->CNT; // leer contador  
    return(CD_Error_NoError);  
}
```



CMSIS y el St STM32F(4)

- Módulos C típicos para CMSIS con microcontrolador STM32F4

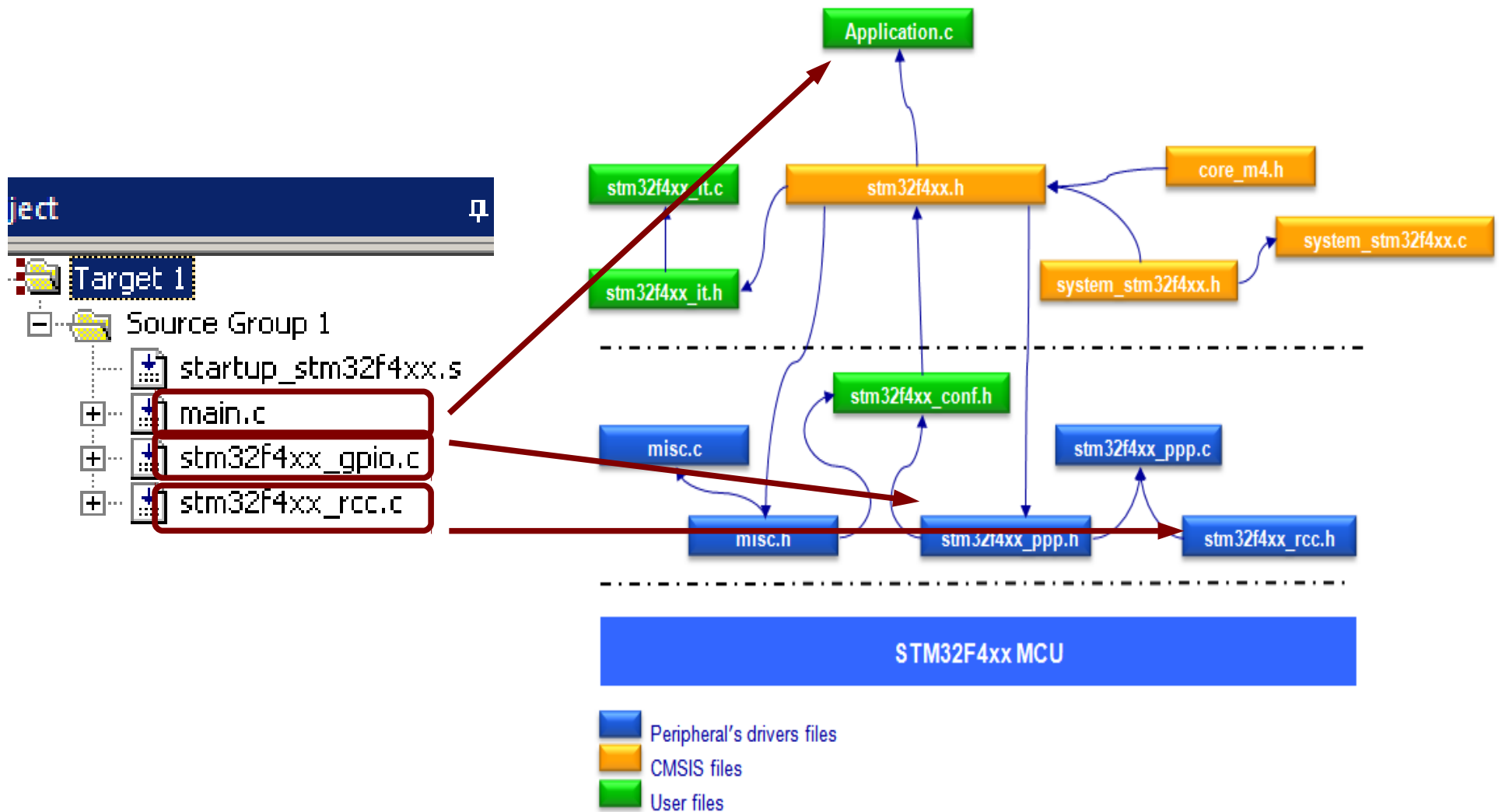


Fte. STM32F4 DSP and Standard peripheral library



CMSIS y el St STM32F(4)

- En nuestro ejemplo mínimo del LED



CMSIS y el St STM32F(4)

- Los cometidos de cada archivo son

| Archivo | Descripción |
|-----------------------|---|
| startup_stm32f4xx.s | STM32F4xx devices startup file. Es un archivo en ensamblador y punto de arranque del microcontrolador. En C, antes de llegar al <code>main()</code> , se pasa por aquí. |
| system_stm32f4xx.h/.c | CMSIS Cortex-M4F STM32F4 devices peripheral access layer system. Son las funciones del core de ARM común a todos los fabricantes adheridos. |
| stm32f4x_???.h /.c | Cabecera y código del driver de un dispositivo ???. Por ejemplo, <code>stm32f4x_spi.h</code> es la cabecera para el driver del SPI. |
| stm32f4xx_conf.h | Peripheral's drivers configuration file. Configurar que cabeceras de periféricos se añaden. No lo toques al principio. |
| stm32f4xx.h | CMSIS Cortex-Mx STMFx device peripheral access. Incluyendo esta cabecera en nuestros módulos debería ser suficiente para acceder a los periféricos. |
| stm32f4xx_it.h/.c | Cabecera y plantillas para todos los servicios de interrupción. Basta añadir nuestro código de servicio dentro. |



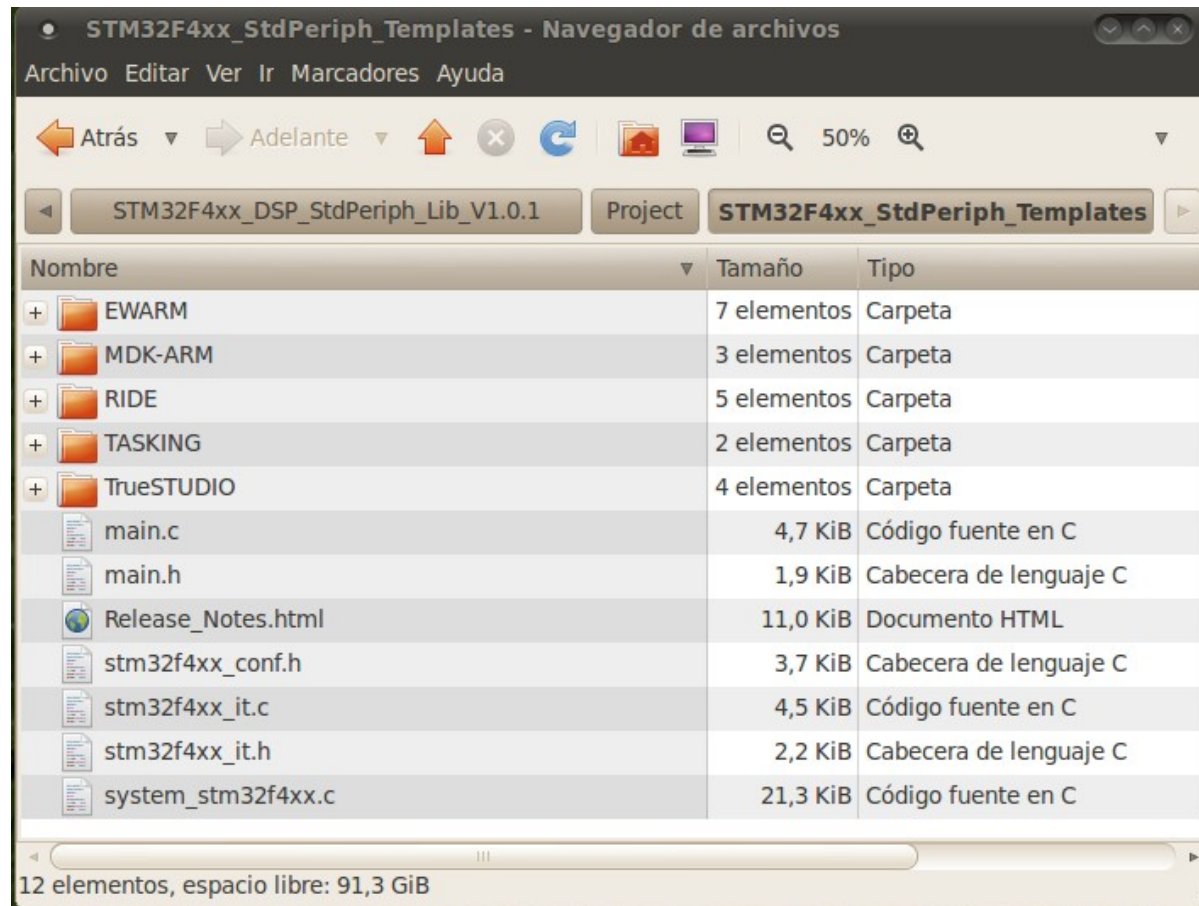
Proyectos: Creándolo con Keil MDK-ARM

- Keil es muy flojillo en esto, cualquier otro entorno es mejor
- La plantilla de la primera sesión se desarrolló con estos pasos
- Actividad:
 - Crea tu mismo el proyecto siguiendo el punto “Crear la primera aplicación” de la guía de iniciación de la STM32F4 Discovery de este curso.
 - Echa un vistazo al interior de los archivos creados. Identifícalos con los de la tabla anterior



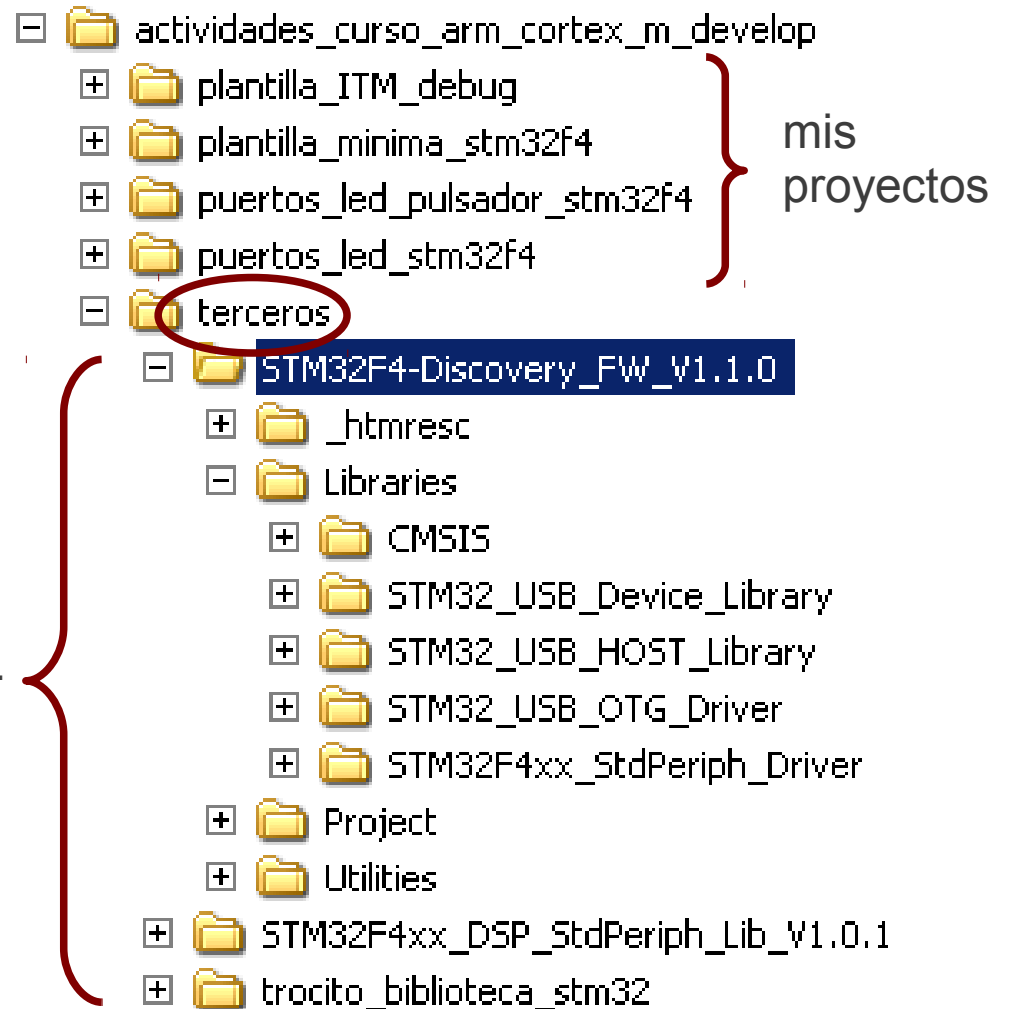
Proyectos: A partir de una plantilla

- Una plantilla es otra opción
 - La proporcionan los fabricantes del chip, los entornos, ...
 - St la proporciona con la “STM32F4 DSP and Standard peripheral library”



Proyectos: A partir de una plantilla

- Un proyecto ARM se debe apoyar en montones de bibliotecas
 - Hay que ser organizados
 - Una plantilla suele usar rutas preconfiguradas
 - En la rutas, evitar, espacios y símbolos raros



Proyectos: A partir de una plantilla

- Con la demo de la Discovery no viene plantilla
 - No problem. Hemos preparado una.
- Actividad: Usar la plantilla
 - 1 - Descomprime la “plantilla con printf() ITM debug” en el directorio de trabajo
 - 2 - Descarga “STM32F4DISCOVERY board firmware package”
 - 3 - Descomprímelo en el subdirectorio “terceros”
 - 4 - Con Keil, abre el proyecto “plantilla_ITM_debug.uvproj”
 - 6 - Prueba a compilar



“Hola Mundo”: al servicio de depuración

- Modifica main.c para incorporar el siguiente código
 - y sigue la guía de activar servicio depuración del blog (English)

```
#include "stm32f4_discovery.h"
#include <stdio.h>

void retardo(uint32_t cuenta);

int main(void)
{
    uint32_t i = 0;

    while (1)
    {

        retardo(20000000);
        printf("Contador %d\n", (int)i);
        i++;
    }
}

void retardo(uint32_t cuenta)
{
    while(cuenta--) {};
}
```



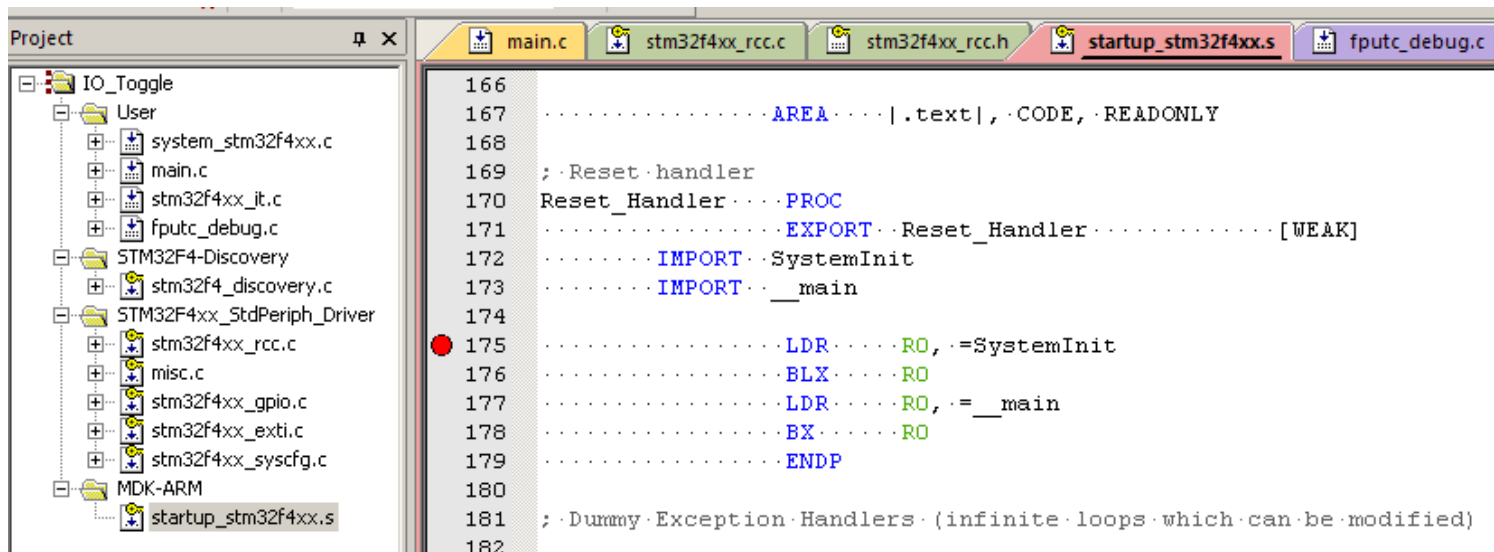
El arranque: del vacío al main()

- En general, un microcontrolador está vacío de software
- Por tanto, nuestro programa será responsable de
 - gestionar la operación de “reset”
 - configurar relojes (osciladores) que marcan el ritmo del sistema
 - ajustar las zonas de memoria donde colocar cosas: heap, stack, code ...
 - proporcionar la función estándar de C main()



El arranque: del vacío al main()

- En general, el arranque lo produce un “reset”
- Aquí, el reset es una petición de interrupción/servicio
- Actividad:
 - Abrir cualquier proyecto y localizar el “handler” del reset. Pensar dónde puede estar a partir de la tabla de módulos
 - Seguir las funciones hasta llegar a main()



```
166
167 ..... AREA ..... |.text|, .CODE, .READONLY
168
169 ; .Reset .handler
170 Reset_Handler ..... PROC
171 ..... EXPORT ..Reset_Handler ..... [WEAK]
172 ..... IMPORT ..SystemInit
173 ..... IMPORT ..__main
174
175 ..... LDR ..... RO, .=SystemInit
176 ..... BLX ..... RO
177 ..... LDR ..... RO, .=__main
178 ..... BX ..... RO
179 ..... ENDP
180
181 ; .Dummy .Exception .Handlers .(infinite .loops .which .can .be .modified)
182
```



Deberes

- Puesss, vemos como sale la sesión

