DISCA
DEPARTAMENT D'INFORMÀTICA DE SISTEMES I COMPUTADORS

Sistemas Embebidos

Programación ARM Cortex-M

Ángel Perles

r08

Contenido

- Objetivos
- Arquitectura
- El arranque
- Ajustes del proyecto
  - stack y heap
  - optimización C
  - target
  - módulos HAL
- Depuración
  - la sonda
  - un ejemplo
  - Cagada: volatile

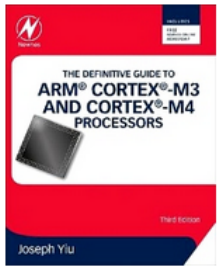
Objetivos

- Conocer el ciclo de arranque de un ARM Cortex-M
- Saber configurar el entorno de ejecución C
- Conocer el sistema de depuración de ARM Cortex-M

- (Rehacer el próximo curso)

Arquitectura

- Visto básicamente en asignaturas previas
- Un vistazo rápido al mapa de memoria
- ¿Más información sobre la arquitectura?



The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors

3rd Edition

★★★★★ 1 Review

View on ScienceDirect

Google Preview

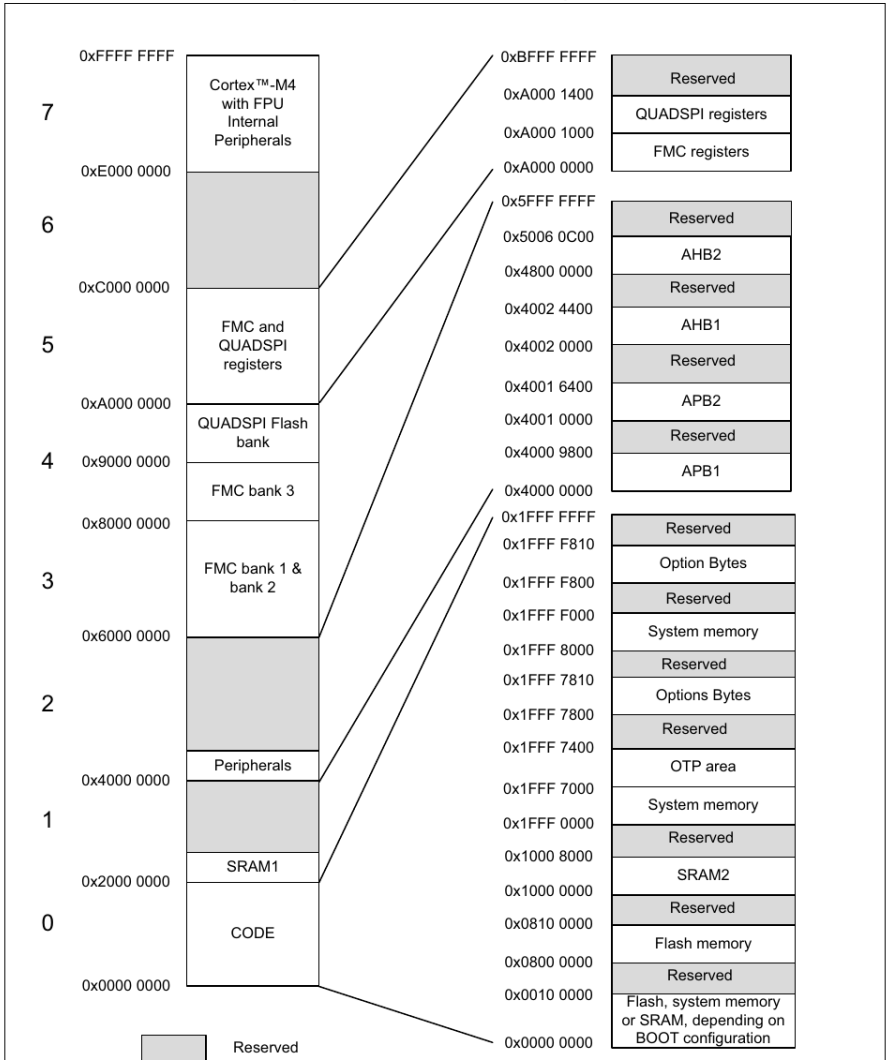
Author: Joseph Yiu

Paperback ISBN: 9780124080829
eBook ISBN: 9780124079182
Hardcover ISBN:

Imprint: Newnes
Published Date: 18th October 2013

Page Count: 864

Figure 17. STM32L476xx memory map

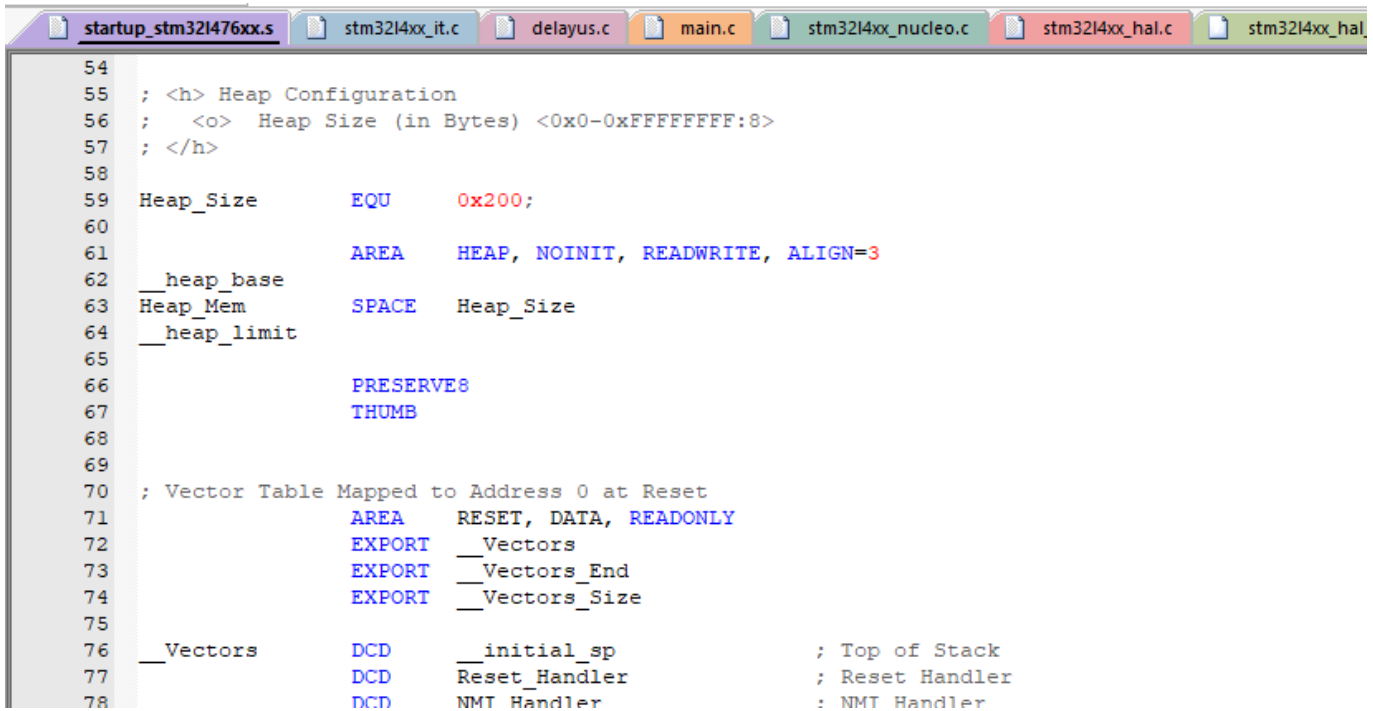


El arranque (o “BigBang”)

- En general, un microcontrolador está vació de software
- Por tanto, nuestro programa debe responsabilizarse de
  - gestionar la operación de “reset”
  - ajustar las zonas de memoria donde colocar cosas: *heap*, *stack*, *code* ... a estas zonas se las llama *segmentos*
  - configurar relojes (osciladores) que marcan el ritmo del sistema
  - otras milongas (reguladores, controladores de memoria y bus en general)
  - y, por fin, proporcionar la función estándar de C *main()*
    - *(es mentira)*

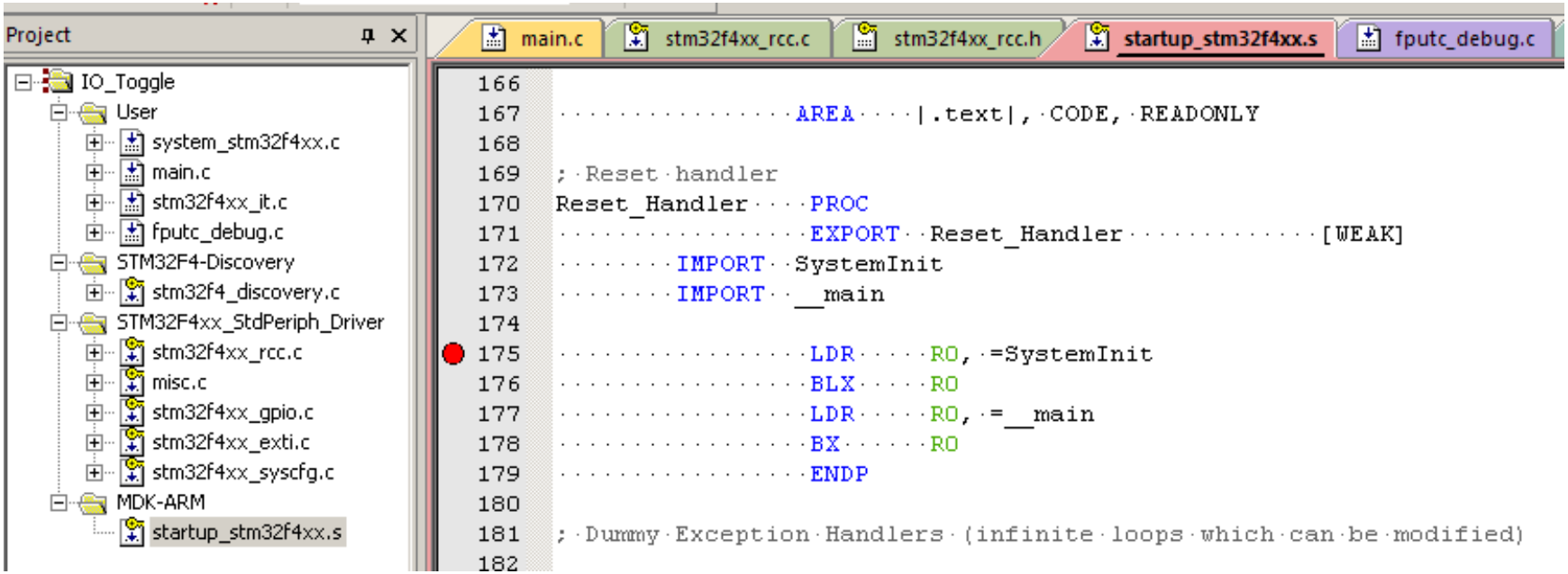
Arranque (o “BigBang”)

- En un ARM Cortex-M, tras un RESET se hace
  - Ajuste del puntero STACK (la pila) con valor en dir 0x00000000
  - Ejecución del vector de RESET cuya dir. está en 0x00000004
- La manera fácil de verlo es abrir un proyecto y el archivo startup*.s



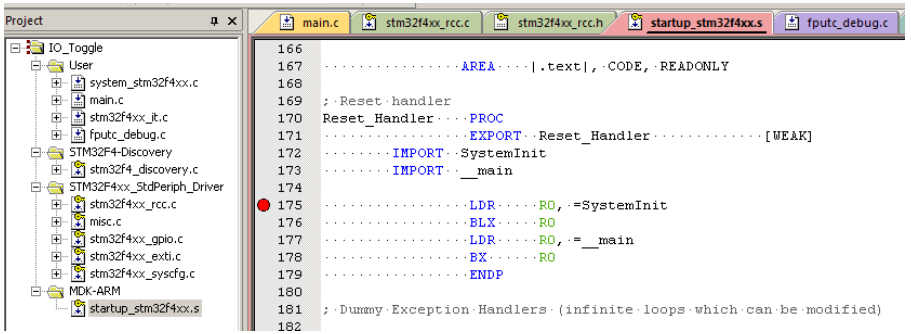
Arranque (o “BigBang”)

- En un ARM Cortex-M, el *reset* es una petición de interrupción
- Por tanto, suponiendo que trabajamos en lenguaje C
  - En el *handler* estará donde empieza todo
  - Lo primero es *SystemInit()*, el verdadero comienzo de un programa C
  - Y después *main()*



Arranque (o “BigBang”)

- Lo vemos en directo



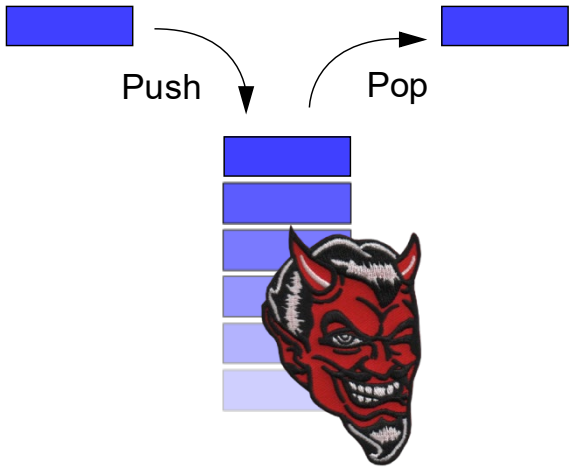
Ajustes del proyecto: stack/heap

- El *stack* se emplea para almacenar/recuperar el estado de los registros en las llamadas a funciones
- Cuanto más anidadas estén las funciones, más *stack* se consume
- Cuidadín, las interrupciones anidan mucho las funciones
- Fuente típica de fallos difíciles e localizar
  - SOLUCIÓN: más *stack* … pero sin pasarse -> necesita más RAM
  - Ejemplo ajuste en Keil: archivo startup_xxx.s

```
; Amount of memory (in bytes) allocated for Stack
; Tailor this value to your application needs
; <h> Stack Configuration
;   <o> Stack Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>

Stack_Size      EQU     0x400

                AREA    STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem       SPACE   Stack_Size
__initial_sp
```



- Lo vemos en directo …

Ajustes del proyecto: stack/heap

- El *heap* se emplea para el mecanismo de petición/liberación de memoria dinámica
- En C, por ejemplo, *malloc()*/*free()*

```
#define NUM_MEASURES 20
fl32_t* pmoistures;

pmoistures = malloc(NUM_MEASURES*sizeof(fl32_t));
```

- En C++ para la creación dinámica de objetos

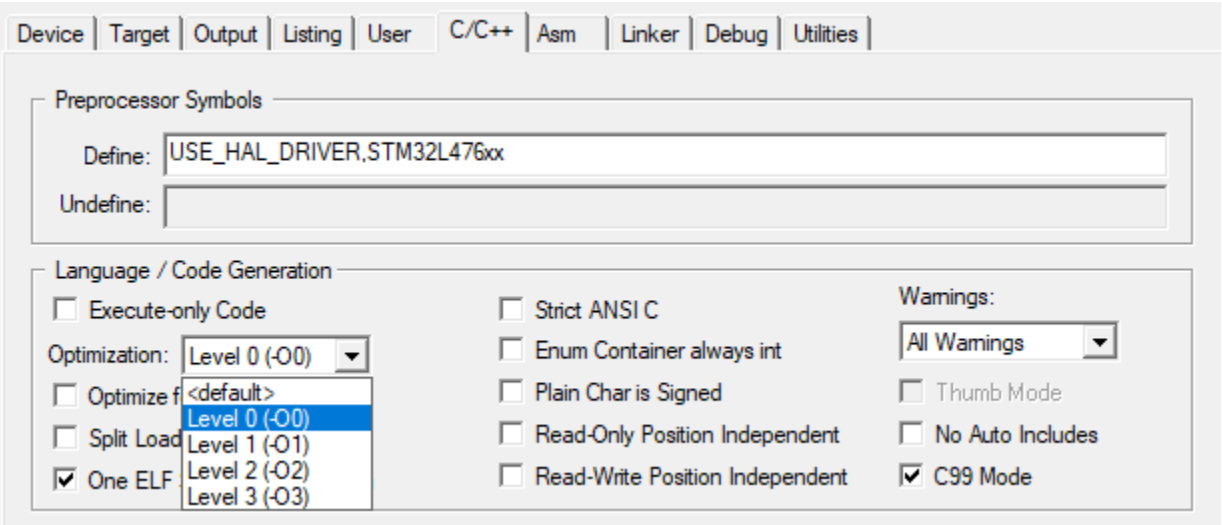
```
#define NUM_SERVOS 30
servo_t *my_servos = new servo_t[NUM_SERVOS];

my_servos[0]->SetTimer(TIM1);
my_servos[0]->SetChannel(1);

my_servos[0]->SetPosition(-900);
```

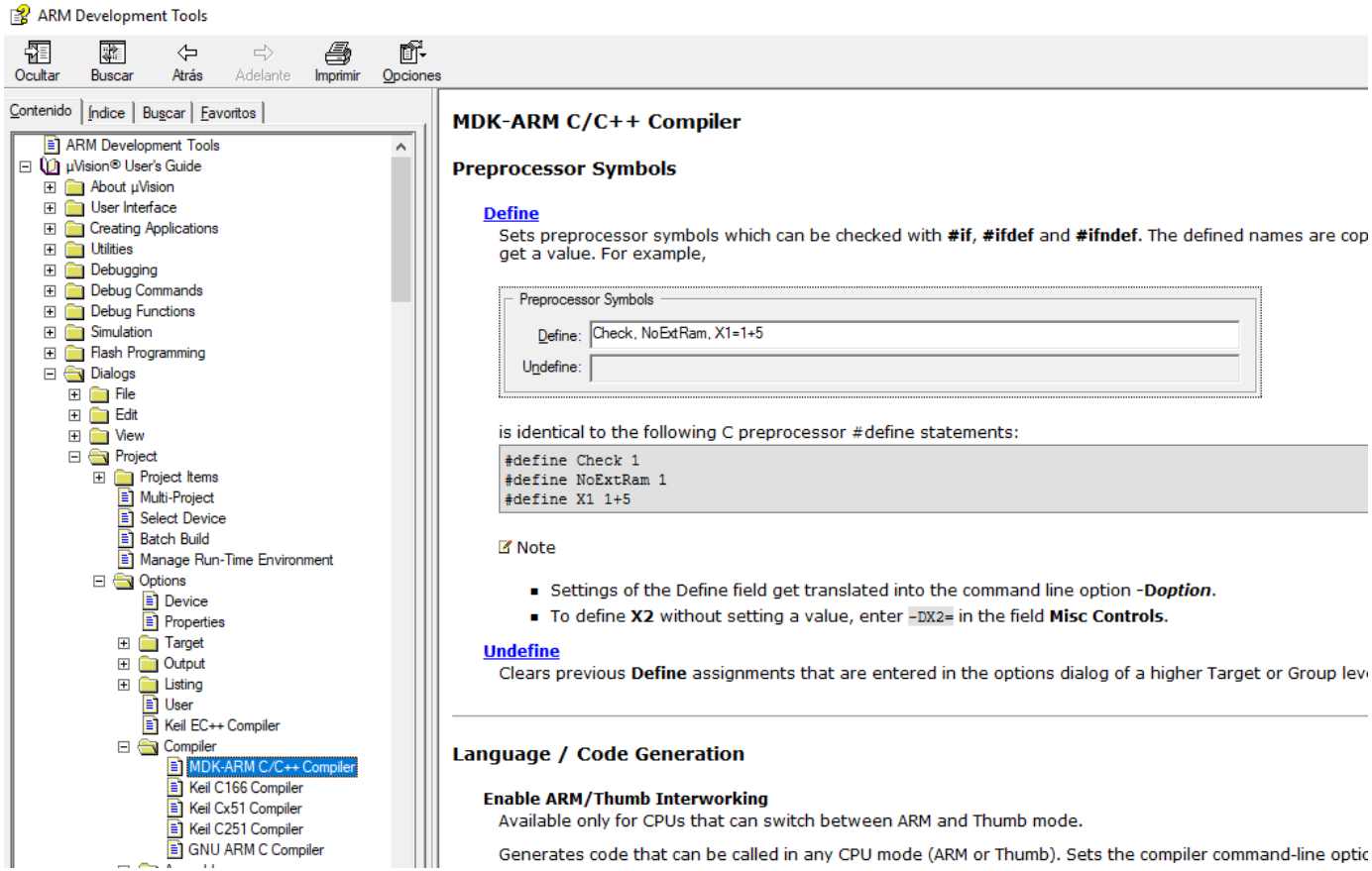
Ajustes del proyecto: optimización C

- El optimizador permite “optimizar el código haciéndolo más rápido y/o pequeño”
- De 0 (no optimiza) … a X (máxima optimización)
- En Keil está aquí



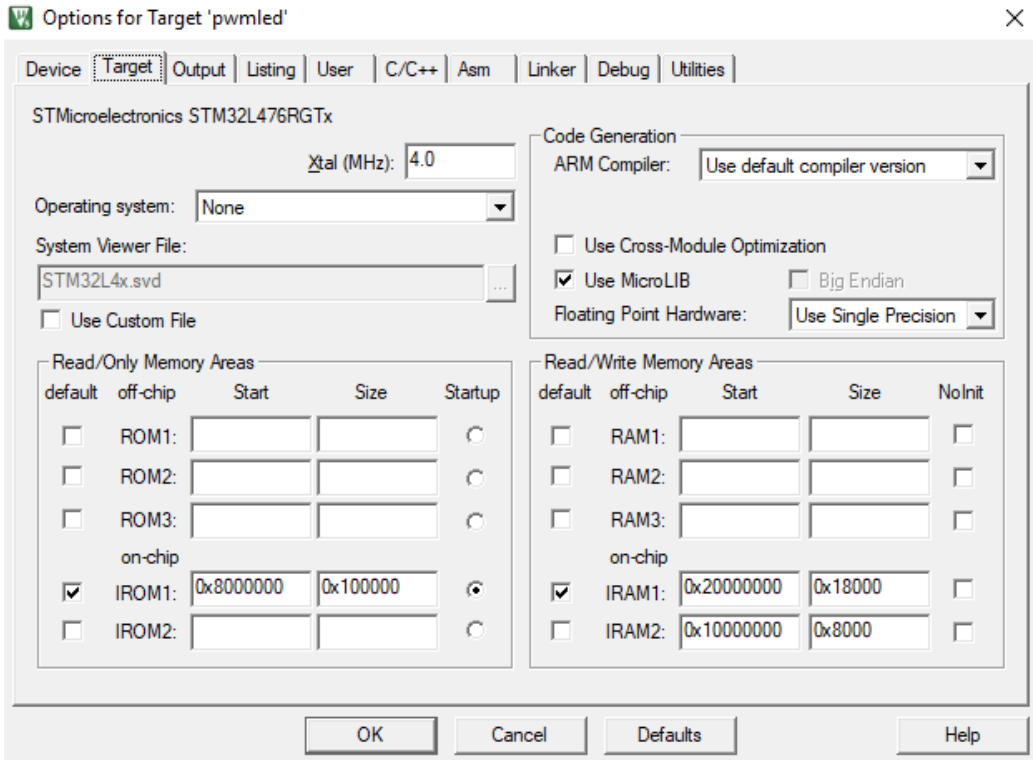
Ajustes del proyecto: optimización C

- Qué hace el optimizador … id al manual del compilador



Ajustes del proyecto: target

- Ajustes asociados al chip usado … y más



- Microlib -> una reducción de la biblioteca estándar C
- Floating point por HW -> hay que habilitar la unidad FPU!

Ajustes del proyecto: módulos HAL

- En archivo stm32l4xx_hal_conf.h y similares
  - Configura, por ejemplo, los módulos activos
  - (Acordarme de meterlo también en el libro un día de estos)

```
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32L4xx_HAL_CONF_H
#define __STM32L4xx_HAL_CONF_H

#ifdef __cplusplus
 extern "C" {
#endif

/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/

/* ########################## Module Selection ############################## */
/**
  * @brief This is the list of modules to be used in the HAL driver
  */

#define HAL_MODULE_ENABLED
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_COMP_MODULE_ENABLED */
#define HAL_CORTEX_MODULE_ENABLED
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DFSDM_MODULE_ENABLED */
#define HAL_DMA_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_I2C_MODULE_ENABLED
/* #define HAL_IWDG_MODULE_ENABLED */
/* #define HAL_LCD_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
/* #define HAL_OPAMP_MODULE_ENABLED */
#define HAL_PWR_MODULE_ENABLED
/* #define HAL_QSPI_MODULE_ENABLED */
#define HAL_RCC_MODULE_ENABLED
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
```

Depuración

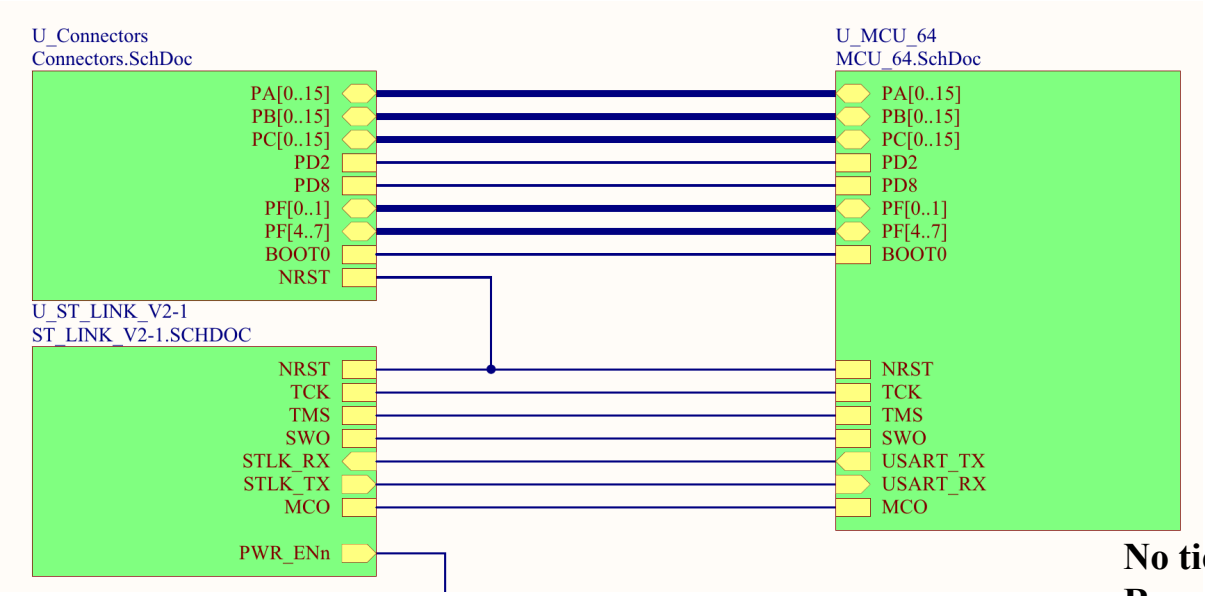
- Diferencia a un juguete (Arduino, p.e.) de algo profesional
- Fundamental para localizar errores
- ARM Cortex-M es la pera en cuanto a depuración

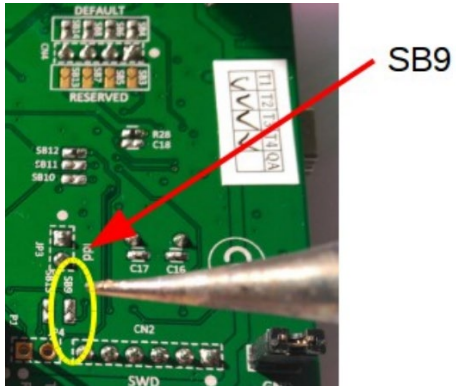
Depuración

- Dependiendo del tipo de ARM Cortex-M tendremos más opciones o menos
  - ARM Cortex-M0 tiene menos opciones
  - Interfaz JTAG clásico y SW (Serial-Wire) para ahorrar líneas de salida
  - CoreSight debug: el estado de la CPU y de la memoria puede ser accedido en vivo (con la CPU funcionando)
  - 6 breakpoints y 4 watchpoints
  - ETM (Exxx) para traza de instrucciones y datos basado en DWT
  - Excepciones (bus fault, memory fault, …) facilitando enormemente la captura de fallos raros
  - ITM para emular el “printf()” típico de depuración de código del año de la pera

Depuración

- Es real: cómo se conecta físicamente en la placa de prácticas
  - La “sonda de depuración” es “ST_LINK_V2-1”
  - NRST: reset del micro
  - TMS(SWDIO)/TCK(SWCLK): SW bus (PA13 y PA14 se pierden)
  - SWO: bus ITM para printf()
  - USART_TX/USART_RX:,puerto serie redirigido al puerto de depuración





No tiene porque estar todo conectado. Recordad placa con pantalla gráfica de II2 y el puente que hicimos.

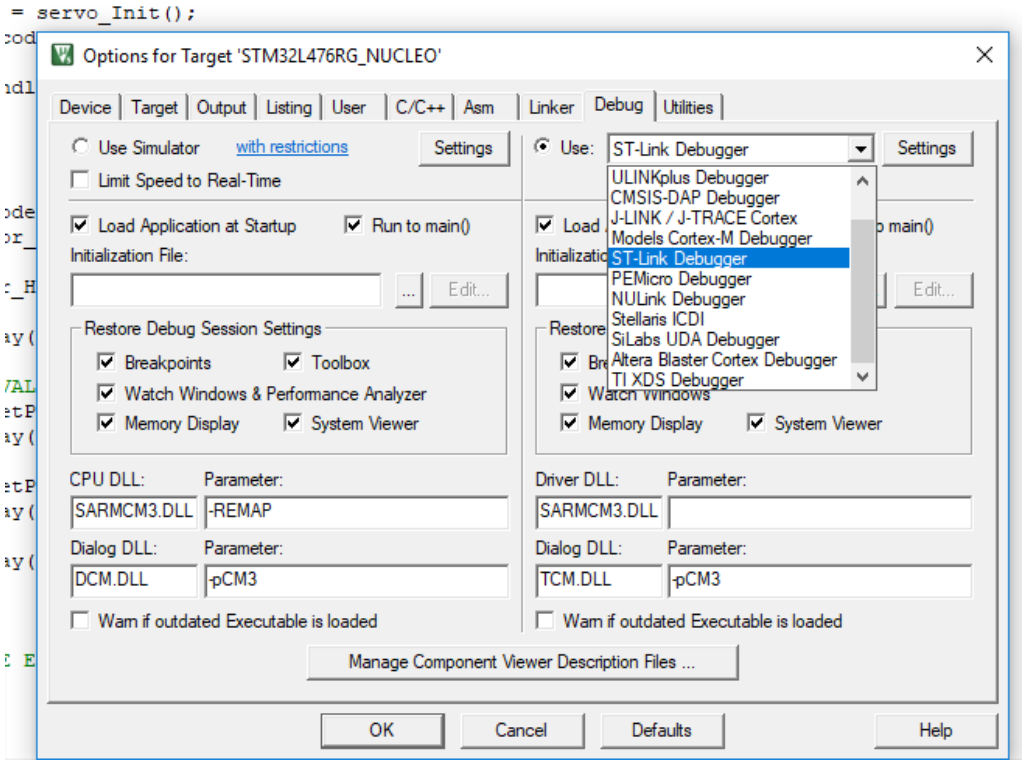
Depuración: sonda

- La “sonda de depuración” es crítica
  - De baratitas incorporadas como la St-Link -> limitada
  - A “top” que permiten explotar más mecanismos de depuración
- Ejemplo “semi-top” Segger J-Trace
- Ejemplo “semi-top” ARM ULINK plus
- Echemos un vistazo …

https://store.developer.arm.com/store/debug-probes/ulinkplus-debug-adapter

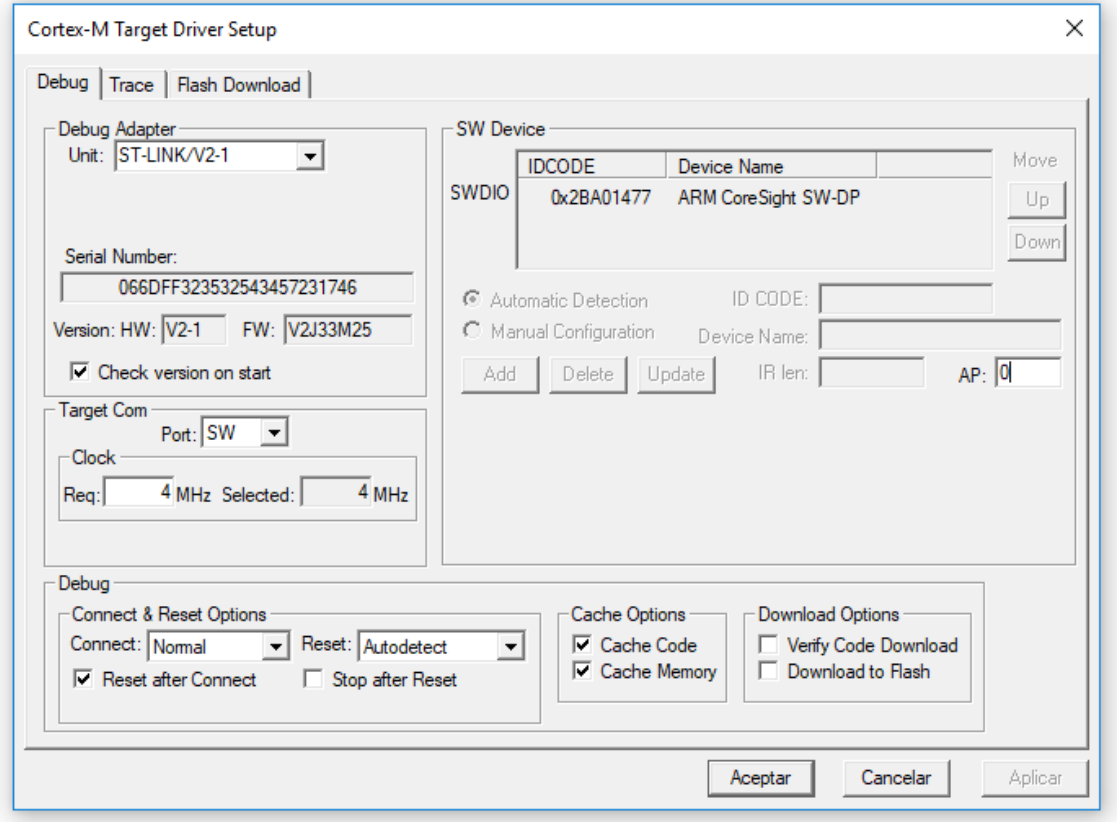
Depuración: sonda St-link

- Puesta a punto de la sonda
  - 1 - Descarga de drivers de St e instalación (siguiendo el libro y ya lo habéis hecho
  - 2 – Configuración en el entorno de desarrollo: p.e. Keil



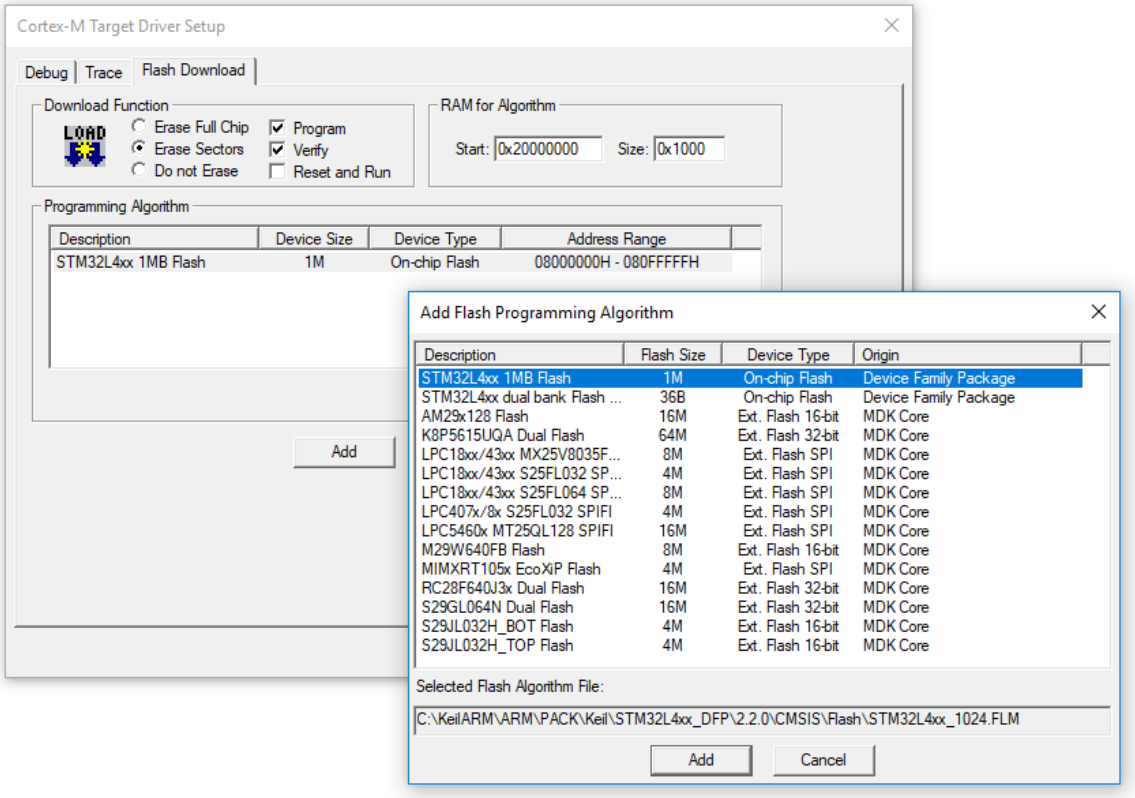
Depuración: sonda St-link

- Puesta a punto de la sonda: elegir “Settings”
  - Debug, Trace, Flash download
  - La sonda tiene que estar conectada (en este caso la placa Nucleo)
  - Jugamos …



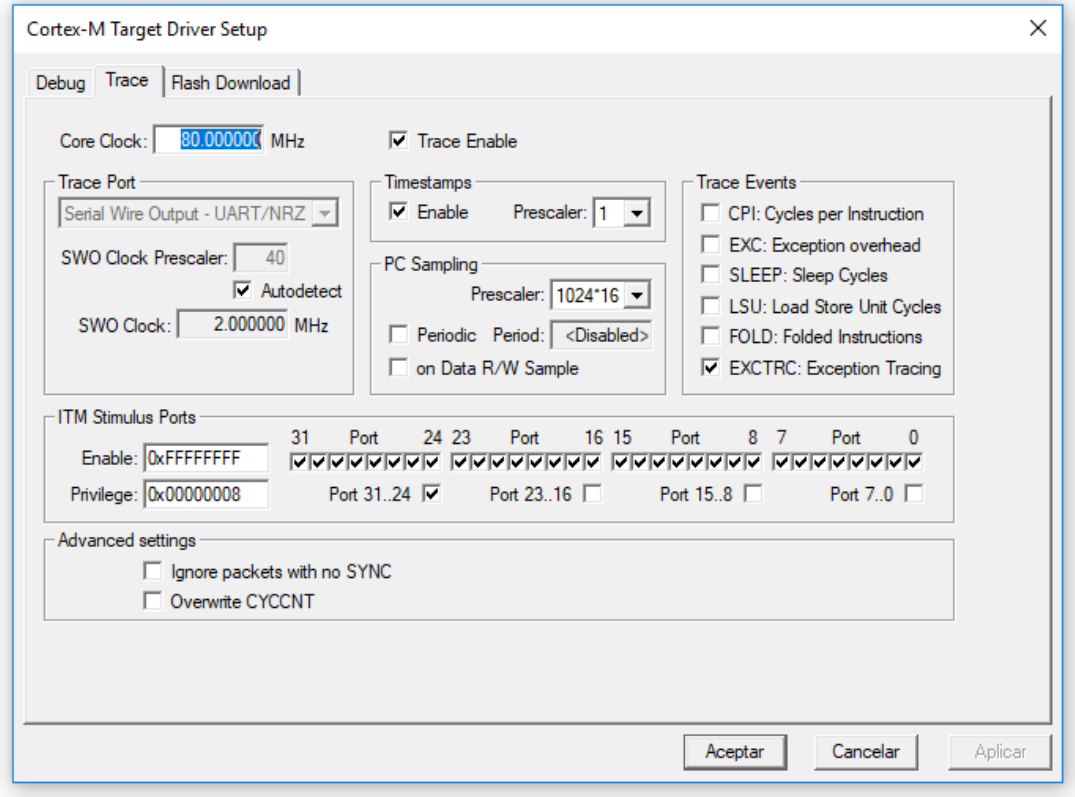
Depuración: sonda St-link

- Puesta a punto de la sonda: elegir “Settings”
  - Debug, Trace, Flash download
  - La sonda se puede encargar también de grabar la flash …



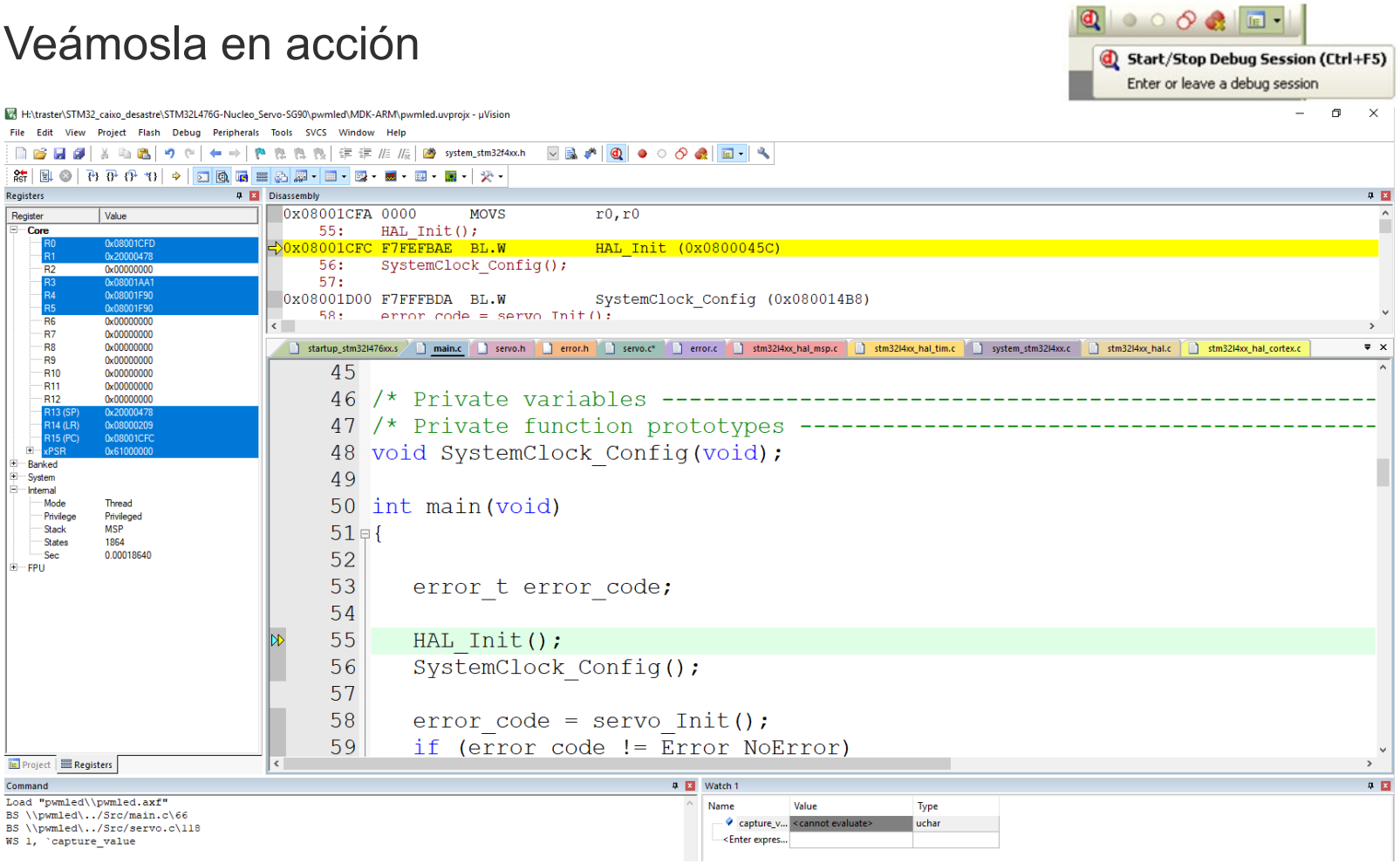
Depuración: sonda St-link

- Puesta a punto de la sonda: elegir “Settings”
  - Debug, Trace, Flash download
  - Qué y cómo se recoge la información de traza … casi ni idea
  - Cuida “Core Clock”



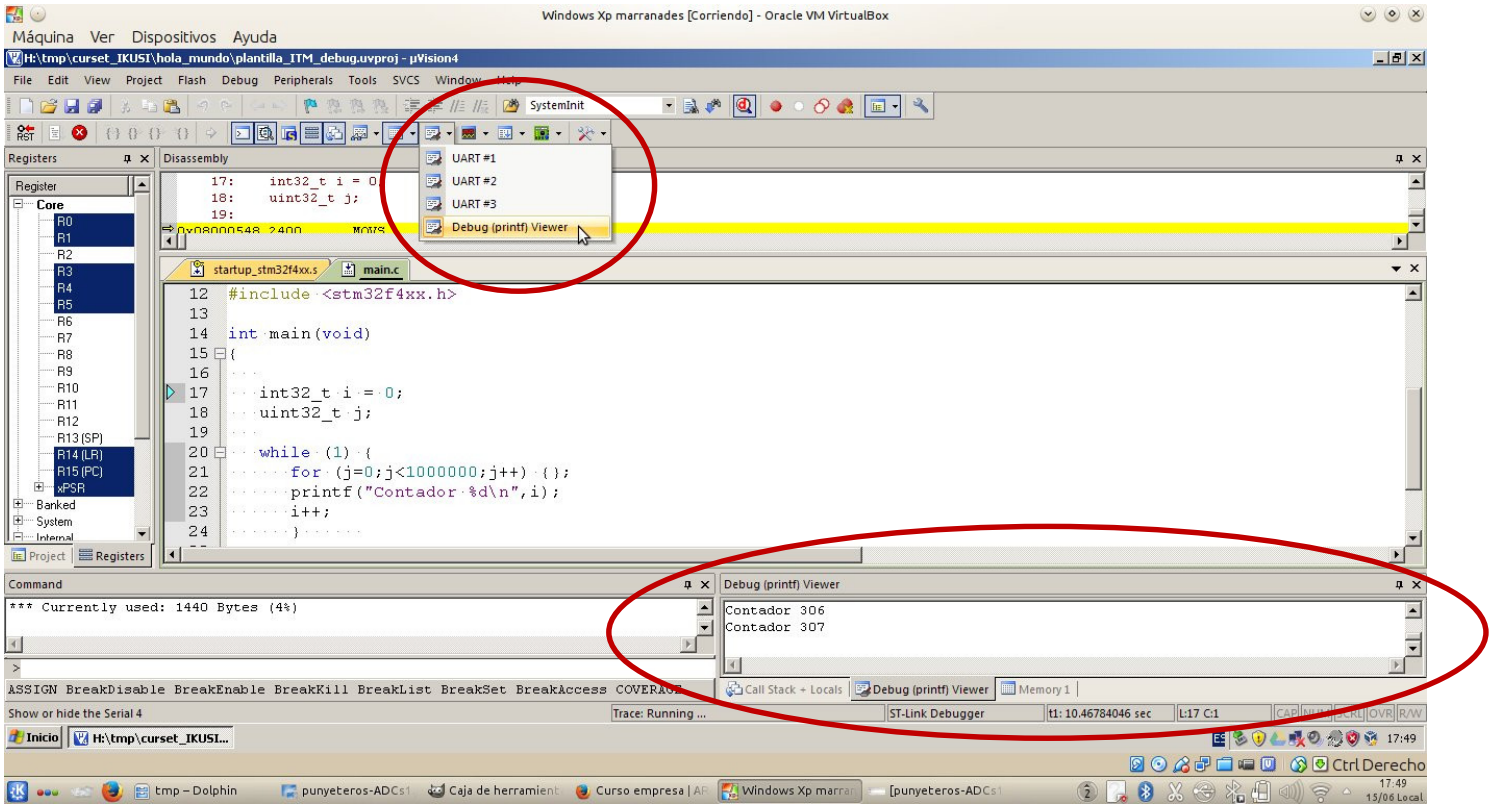
Depuración: un ejemplo

- A PARTIR DE HOY NADA DE DARLE AL BOTÓN “RESET”
- Veámosla en acción



Depuración: printf()

- La plantilla está preparada para usar SWO
  - Redireción del flujo de salida estándar (putc(), printf() …)



https://aperles.blogs.upv.es/2012/06/25/stm32f4-discovery-and-printf-redirection-to-debug-viewer-in-keil/

Depuración: pifias

- Deberíamos usar siempre nivel optimización 0 durante el desarrollo
- O lo sufriremos en la depuración, condiciones de carrera, funcionamiento extraño …
- La herramientas automáticas suelen ponerlo a 3 -> ¡Cámbialo!

-O0

```
90
91  GPIO_InitStruct.Pin = GPIO_PIN_5;
92  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
93  GPIO_InitStruct.Pull = GPIO_NOPULL;
94  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
95  GPIO_InitStruct.Alternate = GPIO_AF1_TIM2;
96  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct); // th
97
98  HAL_TIM_PWM_Start(&htim,TIM_CHANNEL_1);
99
100 return Error_NoError;
101 }
```

-O3

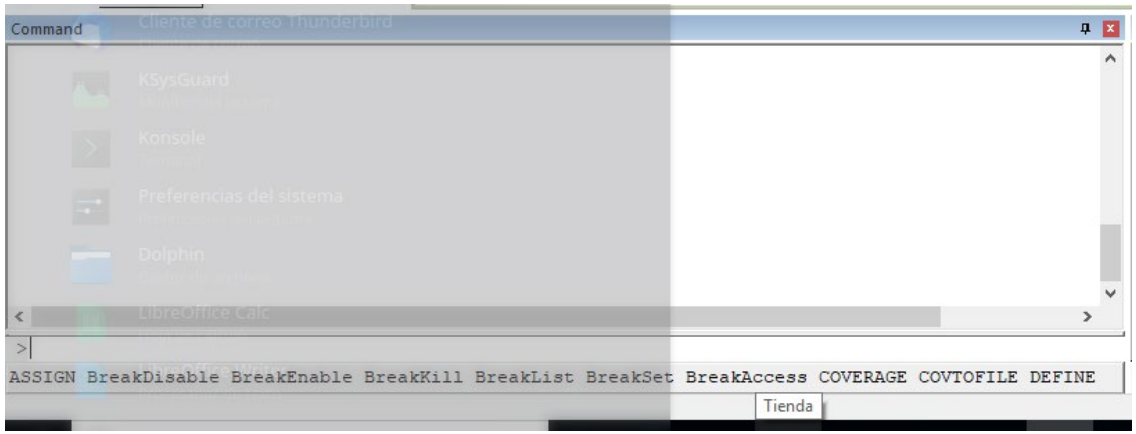
```
89  GPIO_InitTypeDef GPIO_InitStruct;
90
91  GPIO_InitStruct.Pin = GPIO_PIN_5;
92  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
93  GPIO_InitStruct.Pull = GPIO_NOPULL;
94  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
95  GPIO_InitStruct.Alternate = GPIO_AF1_TIM2;
96  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct); // th
97
98  HAL_TIM_PWM_Start(&htim,TIM_CHANNEL_1);
```

- (Nivel de optimización y el “volatile” e ilustrarlo con el examen y el contador de pulsaciones)

Depuración: step-by-step

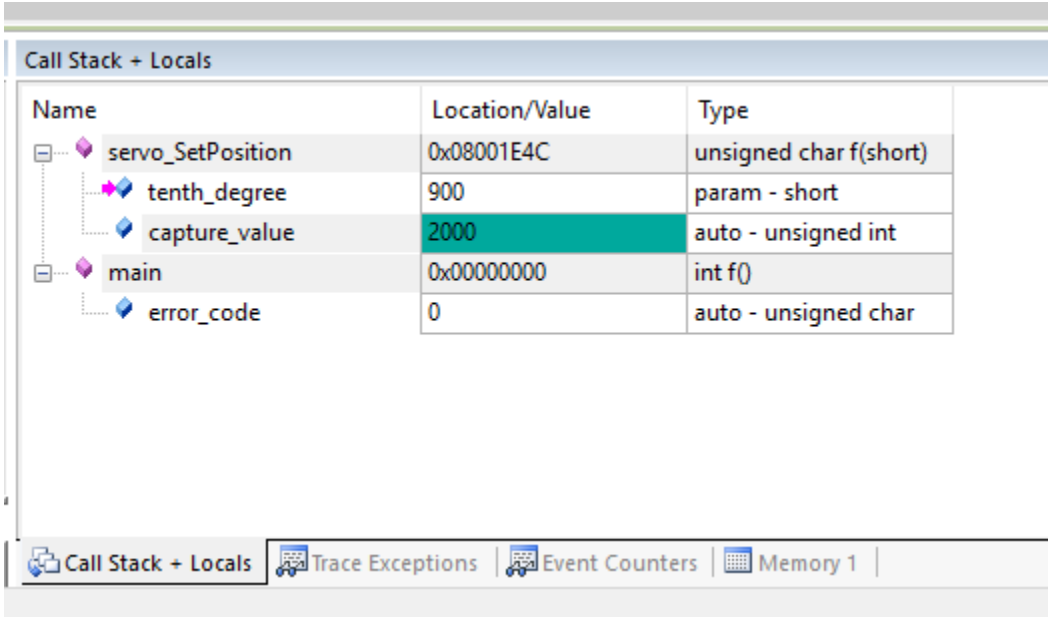
- Botones de control
  - Reset, run, step, step-over, run to cursor, step out of the function …

  - Command window (ni idea de cómo se usa, pero parece útil)



Depuración: step-by-step

- Call-stack and locals
  - De dónde venimos y las variables automáticas creadas
  - Podemos cambiar la representación entre hexadecimal/decimal/…
  - ¡Podemos tocar las variables!



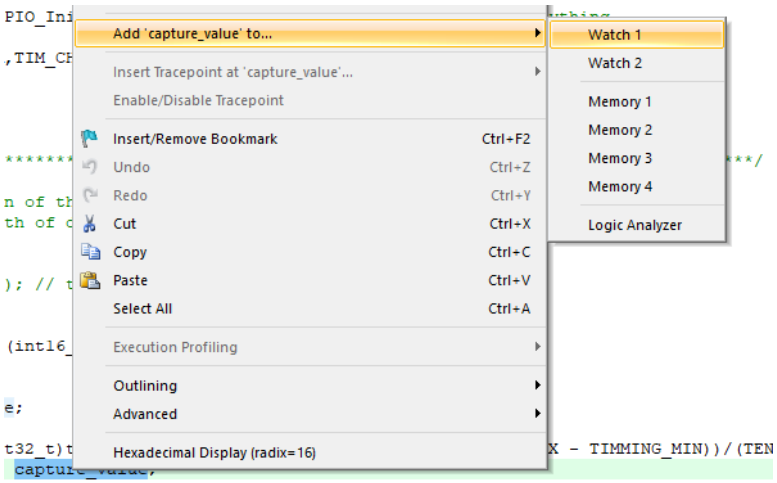
Depuración: step-by-step

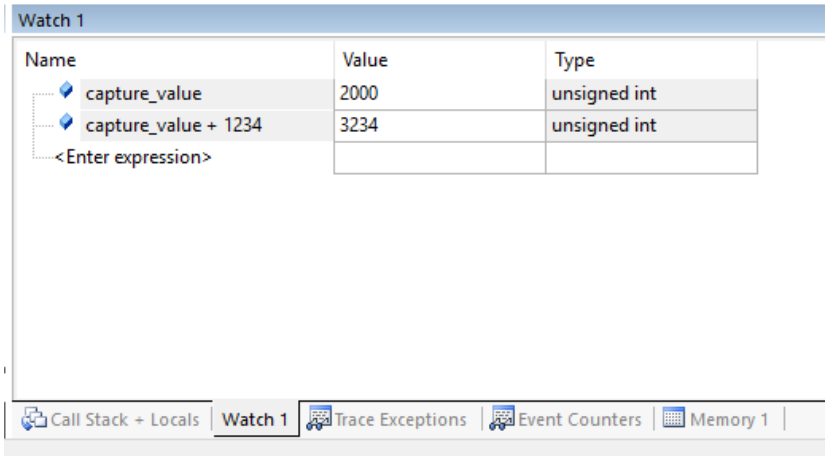
- Podemos ojear variables

```
lue;

int32_t)tenth_degree - TENTH_DI
 = capture_value;
      capture_value = 2000
r;
```

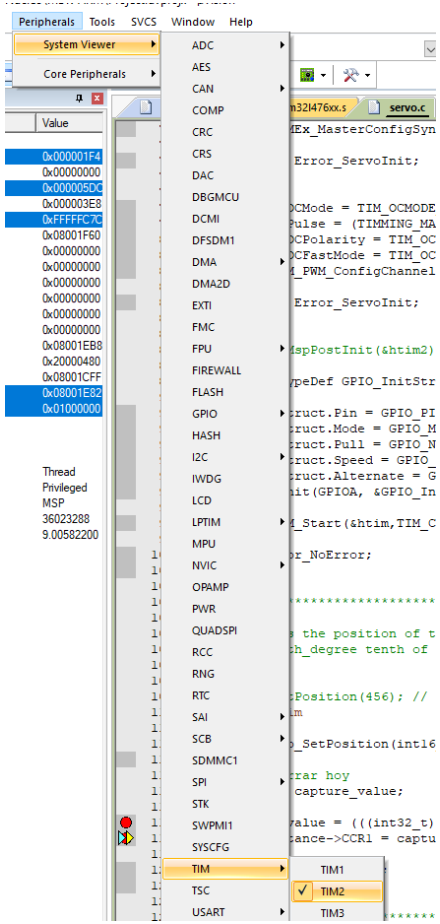
- O vigilarlas con un “watch”, “memory” y “lógic analyzer”

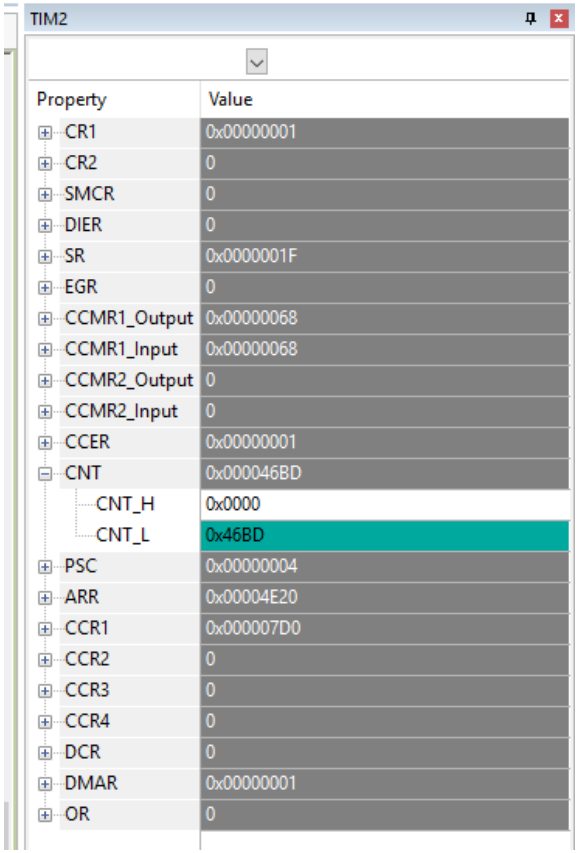




Depuración: periféricos

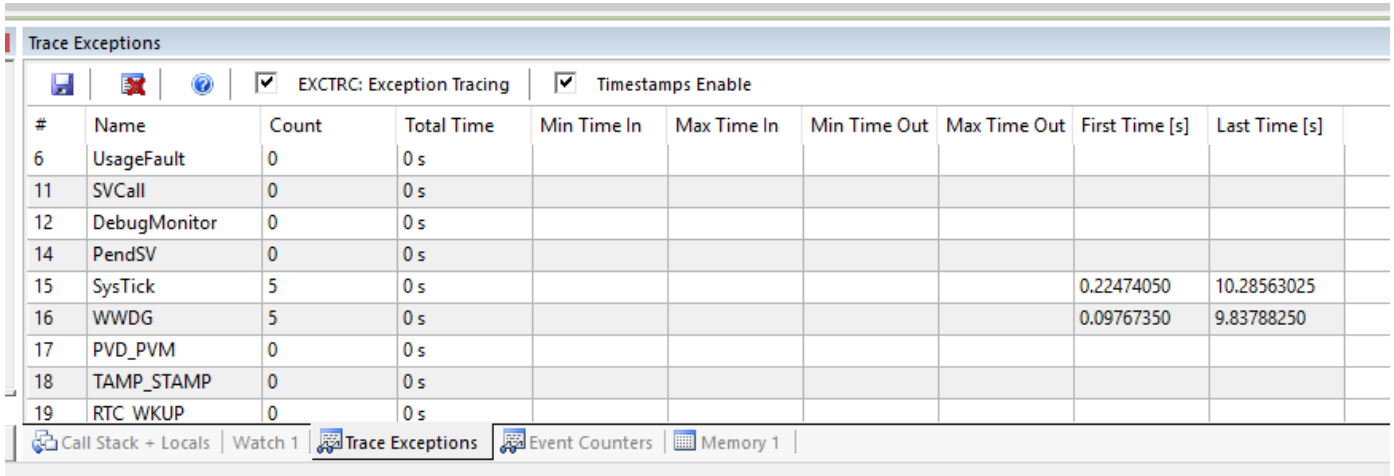
- Estilo “Reference manual” (registros internos)





Depuración: trazas

- Próximo curso
- TO BE CONTINUED …



#	Name	Count	Total Time	Min Time In	Max Time In	Min Time Out	Max Time Out	First Time [s]	Last Time [s]
6	UsageFault	0	0 s						
11	SVCall	0	0 s						
12	DebugMonitor	0	0 s						
14	PendSV	0	0 s						
15	SysTick	5	0 s					0.22474050	10.28563025
16	WWDG	5	0 s					0.09767350	9.83788250
17	PVD_PVM	0	0 s						
18	TAMP_STAMP	0	0 s						
19	RTC_WKUP	0	0 s						


The End

(Cajón de sastre)

- Bip, bip