



Programación ARM Cortex-M

Àngel 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



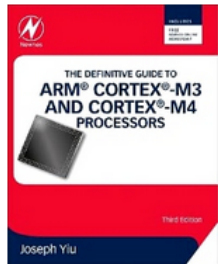
Objetivos

- 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

Author: Joseph Yiu

Paperback ISBN: 9780124080829

eBook ISBN: 9780124079182

Hardcover ISBN:

Imprint: Newnes

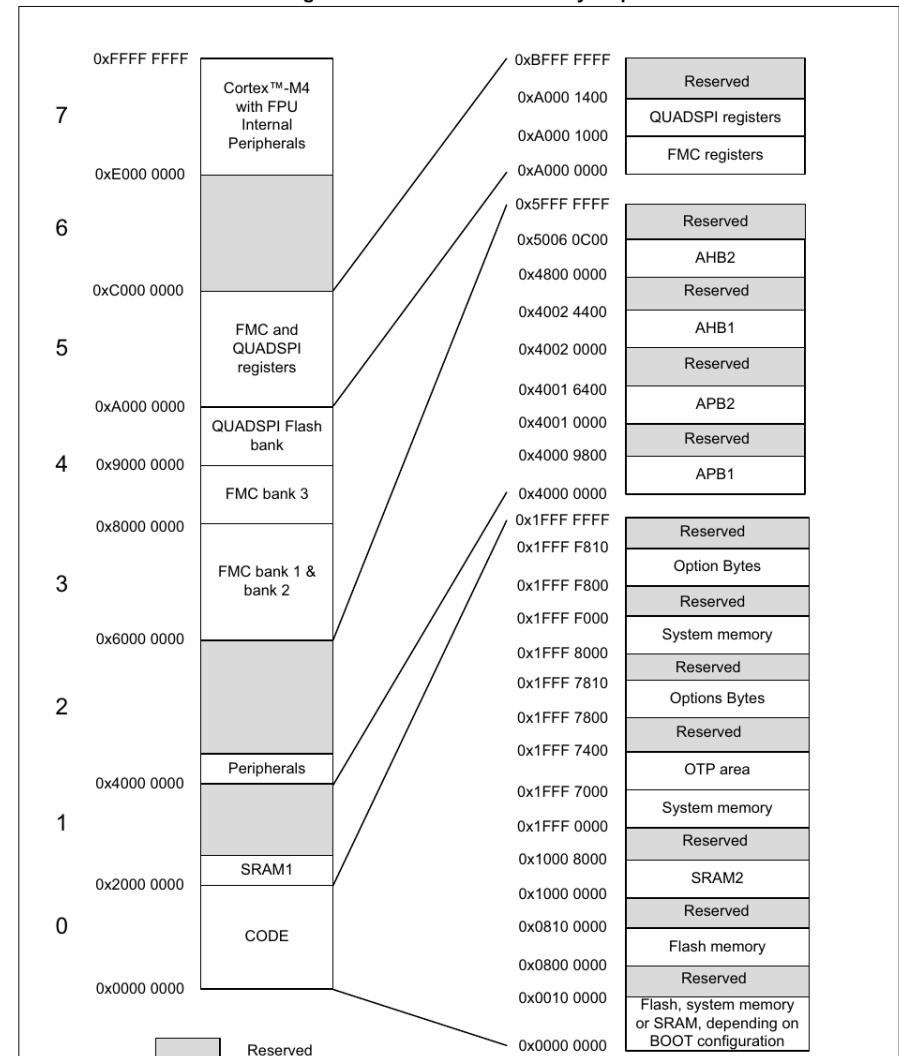
Published Date: 18th October 2013

Page Count: 864

[View on ScienceDirect](#)



Figure 17. STM32L476xx memory map



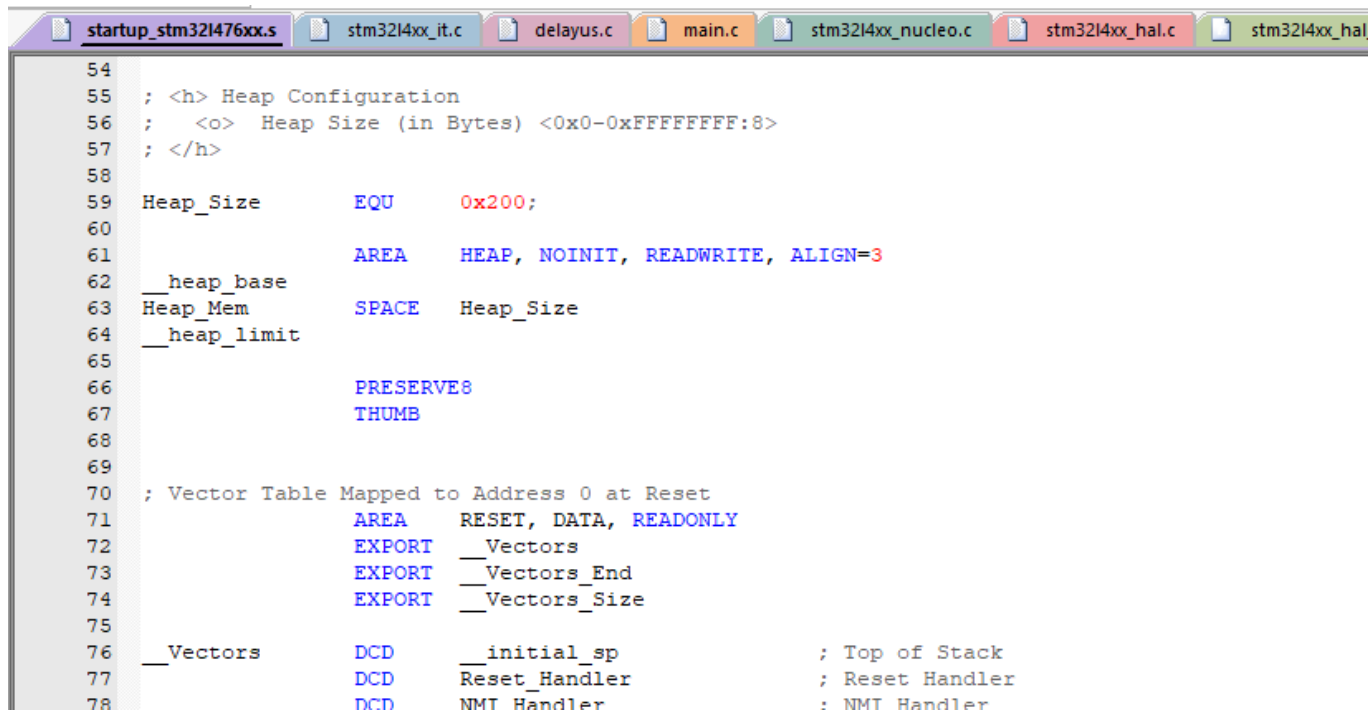
El arranque (o “BigBang”)

- En general, un microcontrolador está vacío 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



```

54
55 ; <h> Heap Configuration
56 ; <o> Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
57 ; </h>
58
59 Heap_Size      EQU      0x200;
60
61               AREA     HEAP, NOINIT, READWRITE, ALIGN=3
62 __heap_base
63 Heap_Mem      SPACE   Heap_Size
64 __heap_limit
65
66               PRESERVE8
67               THUMB
68
69
70 ; Vector Table Mapped to Address 0 at Reset
71               AREA     RESET, DATA, READONLY
72               EXPORT   __Vectors
73               EXPORT   __Vectors_End
74               EXPORT   __Vectors_Size
75
76 __Vectors     DCD      __initial_sp          ; Top of Stack
77               DCD      Reset_Handler        ; Reset Handler
78               DCD      NMI_Handler          ; NMI Handler

```



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()*

```

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

```



Arranque (o “BigBang”)

- Lo vemos en directo

```

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
  
```



Ajustes del proyecto: stack/heap

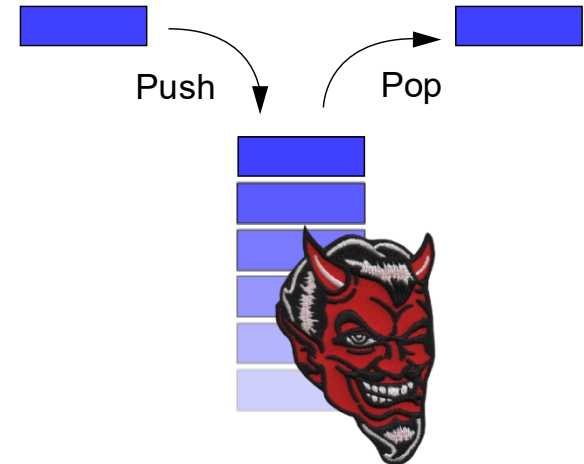
- El *stack* se emplea para almacenar/recuperar el estado de los registros en las llamadas a funciones
- Cuanto más anidadas estén las funciones, más *stack* se consume
- Cuidadín, las interrupciones anidan mucho las funciones
- Fuente típica de fallos difíciles e localizar
 - SOLUCIÓN: más *stack* ... 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 ];

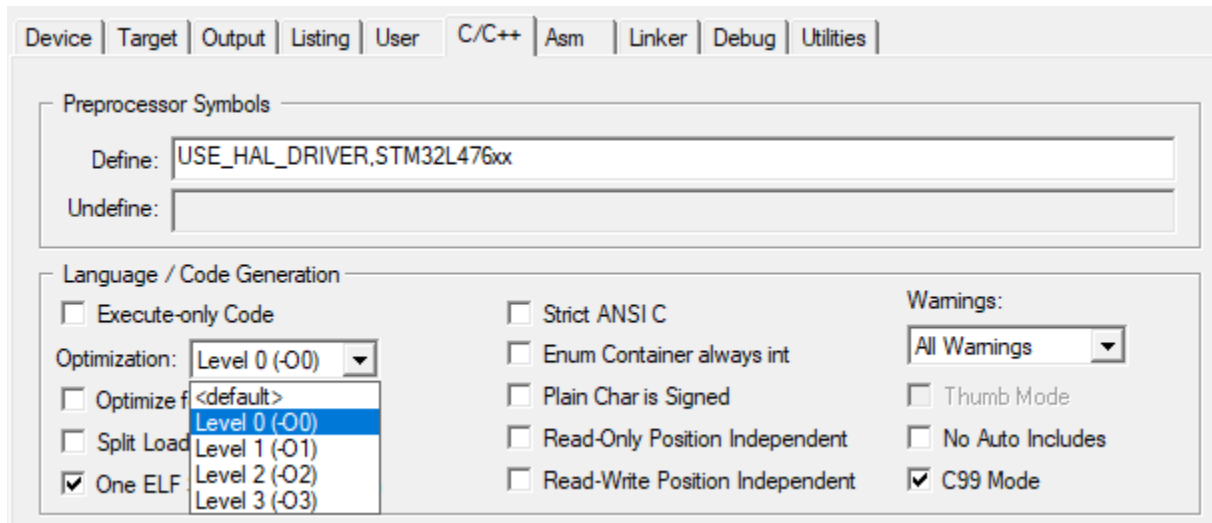
my_servos[0]->SetTimer(TIM1);
my_servos[0]->SetChannel(1);

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



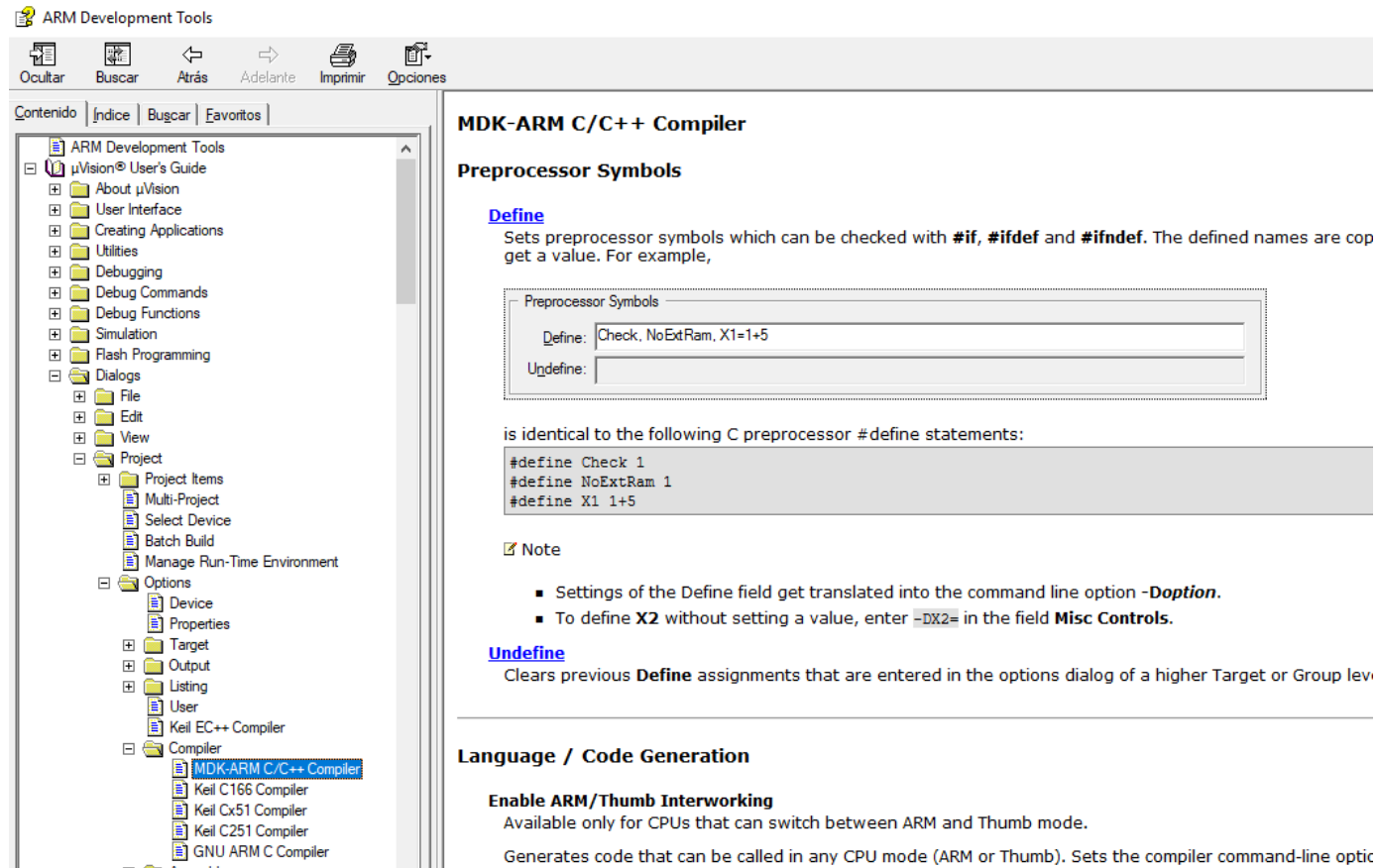
Ajustes del proyecto: optimización C

- 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



MDK-ARM C/C++ Compiler

Preprocessor Symbols

[Define](#)
Sets preprocessor symbols which can be checked with **#if**, **#ifdef** and **#ifndef**. The defined names are got a value. For example,

Preprocessor Symbols

Define: Check, NoExtRam, X1=1+5

Undefine:

is identical to the following C preprocessor #define statements:

```
#define Check 1
#define NoExtRam 1
#define X1 1+5
```

Note

- Settings of the Define field get translated into the command line option **-Doption**.
- To define **X2** without setting a value, enter **-DX2=** in the field **Misc Controls**.

[Undefine](#)
Clears previous **Define** assignments that are entered in the options dialog of a higher Target or Group level

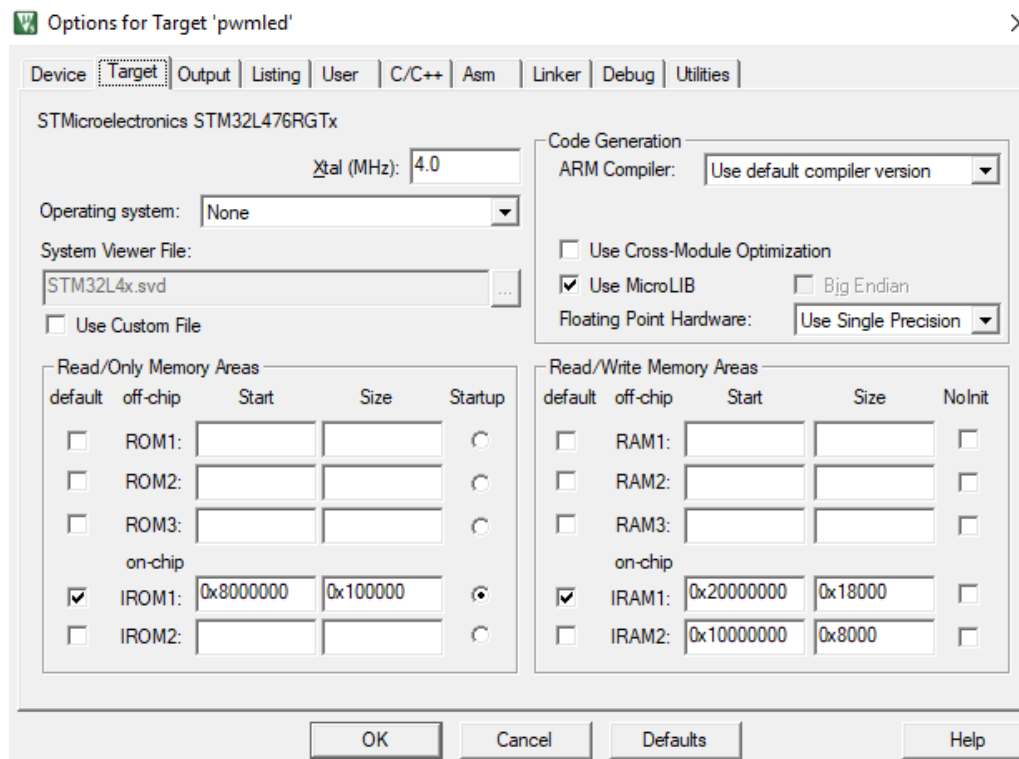
Language / Code Generation

Enable ARM/Thumb Interworking
Available only for CPUs that can switch between ARM and Thumb mode.
Generates code that can be called in any CPU mode (ARM or Thumb). Sets the compiler command-line option



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)

```

40 /* Define to prevent recursive inclusion -----*/
41 #ifndef __STM32L4xx_HAL_CONF_H
42 #define __STM32L4xx_HAL_CONF_H
43
44 #ifndef __cplusplus
45 extern "C" {
46 #endif
47
48 /* Exported types -----*/
49 /* Exported constants -----*/
50
51 /* ##### Module Selection ##### */
52 /**
53  * @brief This is the list of modules to be used in the HAL driver
54  */
55
56 #define HAL_MODULE_ENABLED
57 /* #define HAL_ADC_MODULE_ENABLED */
58 /* #define HAL_CAN_MODULE_ENABLED */
59 /* #define HAL_CAN_LEGACY_MODULE_ENABLED */
60 /* #define HAL_COMP_MODULE_ENABLED */
61 #define HAL_CORTEX_MODULE_ENABLED
62 /* #define HAL_CRC_MODULE_ENABLED */
63 /* #define HAL_Cryp_MODULE_ENABLED */
64 /* #define HAL_DAC_MODULE_ENABLED */
65 /* #define HAL_DFSDM_MODULE_ENABLED */
66 #define HAL_DMA_MODULE_ENABLED
67 #define HAL_FLASH_MODULE_ENABLED
68 /* #define HAL_NAND_MODULE_ENABLED */
69 /* #define HAL_NOR_MODULE_ENABLED */
70 /* #define HAL_SRAM_MODULE_ENABLED */
71 #define HAL_GPIO_MODULE_ENABLED
72 #define HAL_I2C_MODULE_ENABLED
73 /* #define HAL_IWDG_MODULE_ENABLED */
74 /* #define HAL_LCD_MODULE_ENABLED */
75 /* #define HAL_LPTIM_MODULE_ENABLED */
76 /* #define HAL_OPAMP_MODULE_ENABLED */
77 #define HAL_PWR_MODULE_ENABLED
78 /* #define HAL_QSPI_MODULE_ENABLED */
79 #define HAL_RCC_MODULE_ENABLED
80 /* #define HAL_RNG_MODULE_ENABLED */
81 /* #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



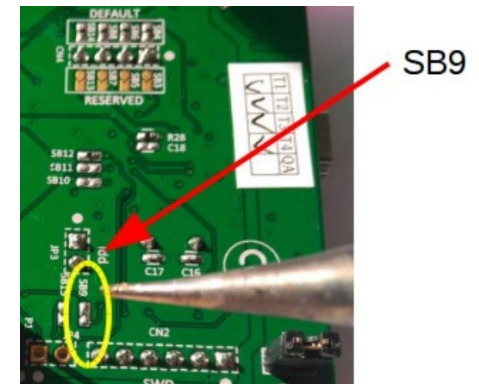
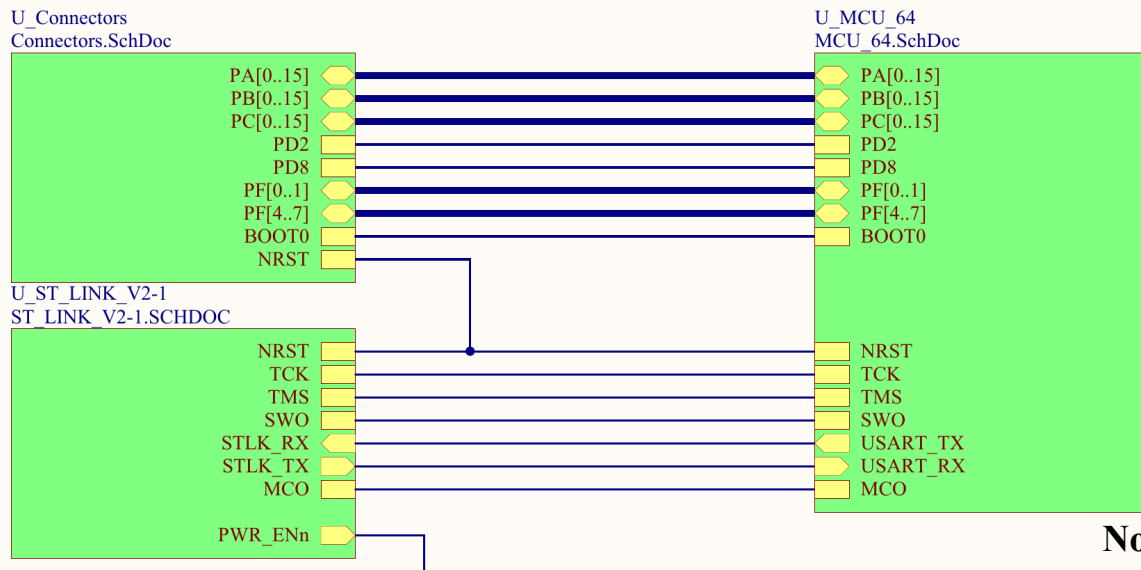
Depuración

- 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 baratas 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 ...



Function	Source Coverage	Inst. Coverage	Load
CRYPTO_ECC_ModMul	71.4% (5/7)	72.1% (31/43)	51.27%
CRYPTO_ECC_ModSquare	71.4% (5/7)	70.7% (29/41)	28.66%
CRYPTO_EC_KillPoint	100.0% (6/6)	100.0% (23/23)	0.41%
CRYPTO_EC_InitPoint	100.0% (6/6)	100.0% (28/28)	0.37%
CRYPTO_RSA_InitPublicKey	100.0% (4/4)	100.0% (18/18)	0.21%


```

CRYPTO_EC_ 35 689 242 int CRYPTO_ECC_ModMul (CRYPTO_MPI
          35 689 243 if (pCurve->pfReduce) {
          35 689 244     CRYPTO_CHECK(CRYPTO_MPI_Mul()
          35 689     0002EFB8 683A LDR
          35 689     0002EFBA 683A LDR
          35 689     0002EFBC 683A BL
          35 688     0002EFBE 683A DR
          35 688     0002EFC0 683A MP
          0
          35 688 245 }
          246 }
          0 247 CR
          248 }
    
```

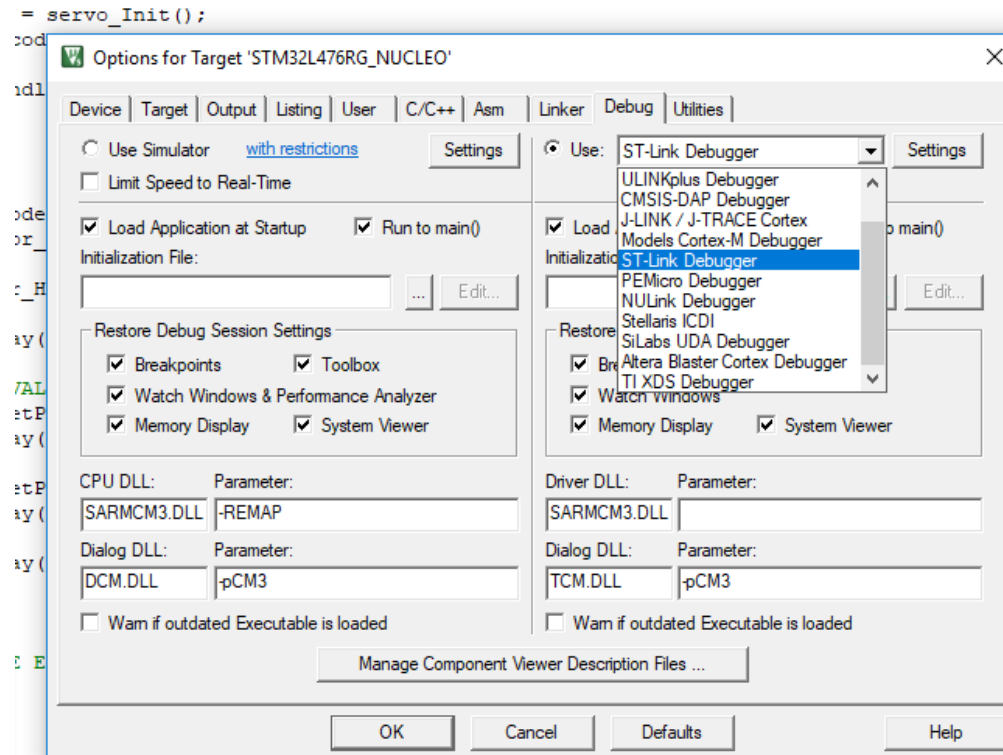


<https://store.developer.arm.com/store/debug-probes/ulinkplus-debug-adapter>



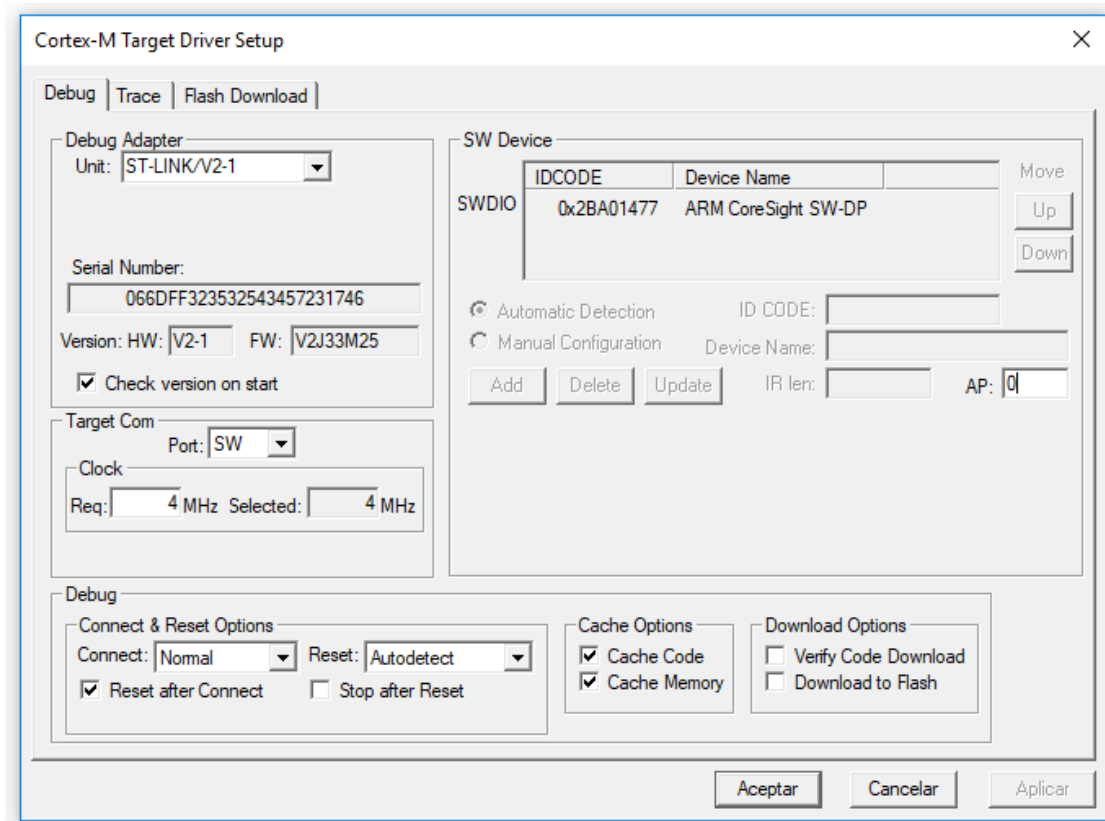
Depuración: sonda St-link

- 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



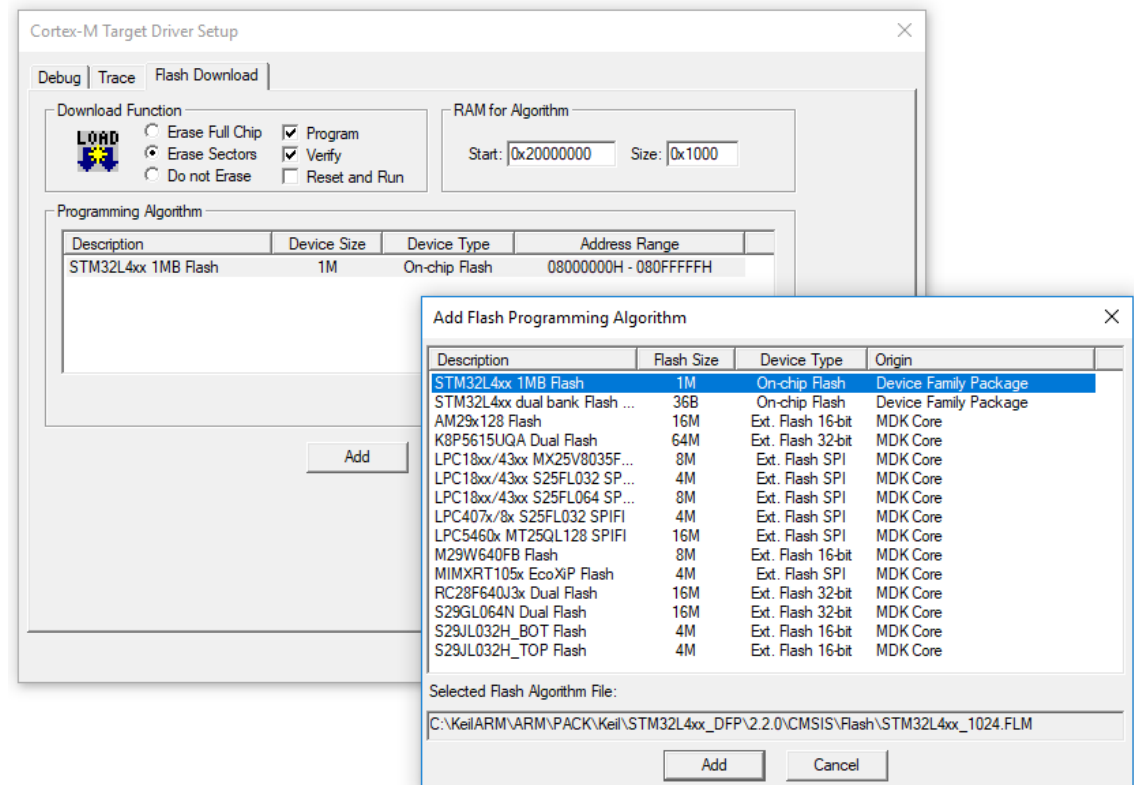
Depuración: sonda St-link

- 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 ...



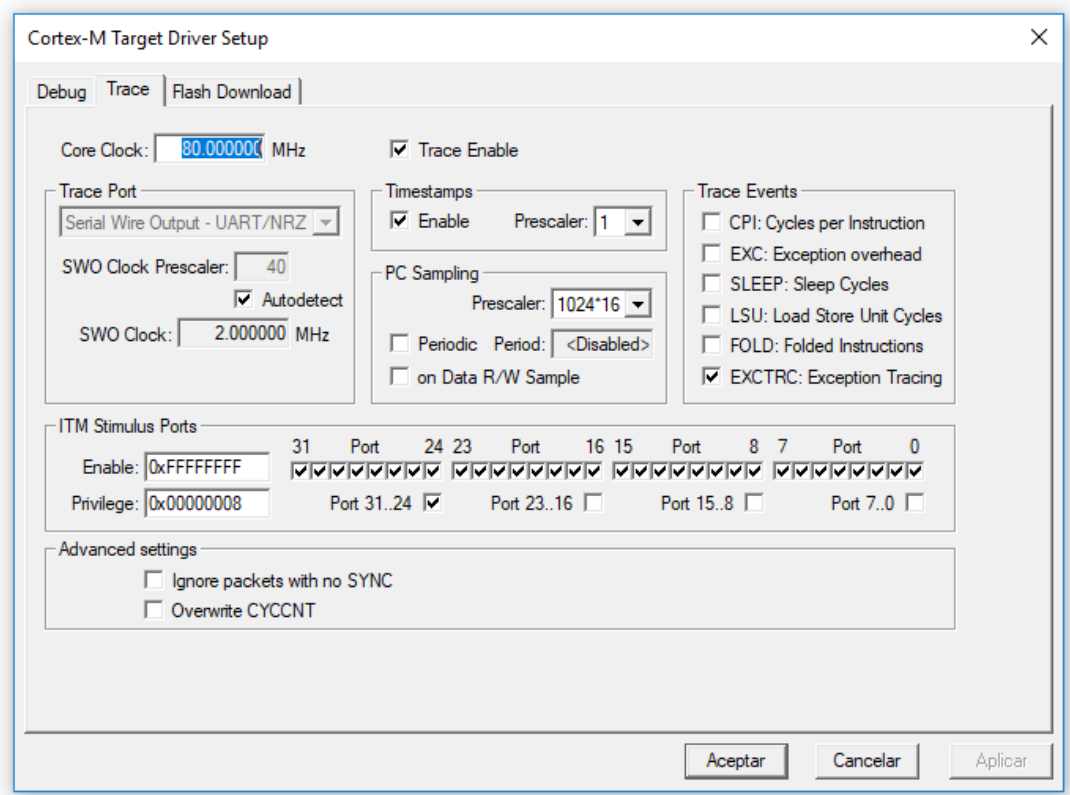
Depuración: sonda St-link

- 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



The screenshot shows an IDE window with the following components:

- Registers:** A table showing core registers (R0-R15, xPSR) and their values. R0 is 0x08001CFD, R1 is 0x20000478, R2 is 0x00000000, R3 is 0x08001AA1, R4 is 0x08001F90, R5 is 0x08001F90, R6-R12 are 0x00000000, R13 (SP) is 0x20000478, R14 (LR) is 0x08000209, R15 (PC) is 0x08001CFC, and xPSR is 0x61000000.
- Disassembly:** Shows assembly instructions:


```

0x08001CFA 0000 MOVS      r0,r0
55: HAL_Init();
->0x08001CFC F7FEFBAE BL.W    HAL_Init (0x0800045C)
56: SystemClock_Config();
57:
0x08001D00 F7FFFBD8 BL.W    SystemClock_Config (0x080014B8)
58: error_code = servo_Init();
      
```
- Source Code:** Shows C code for `main.c`:


```

45
46 /* Private variables -----
47 /* Private function prototypes -----
48 void SystemClock_Config(void);
49
50 int main(void)
51 {
52     error_t error_code;
53
54     HAL_Init();
55     SystemClock_Config();
56
57     error_code = servo_Init();
58     if (error code != Error NoError)
59
      
```
- Command Window:** Shows the following commands:


```

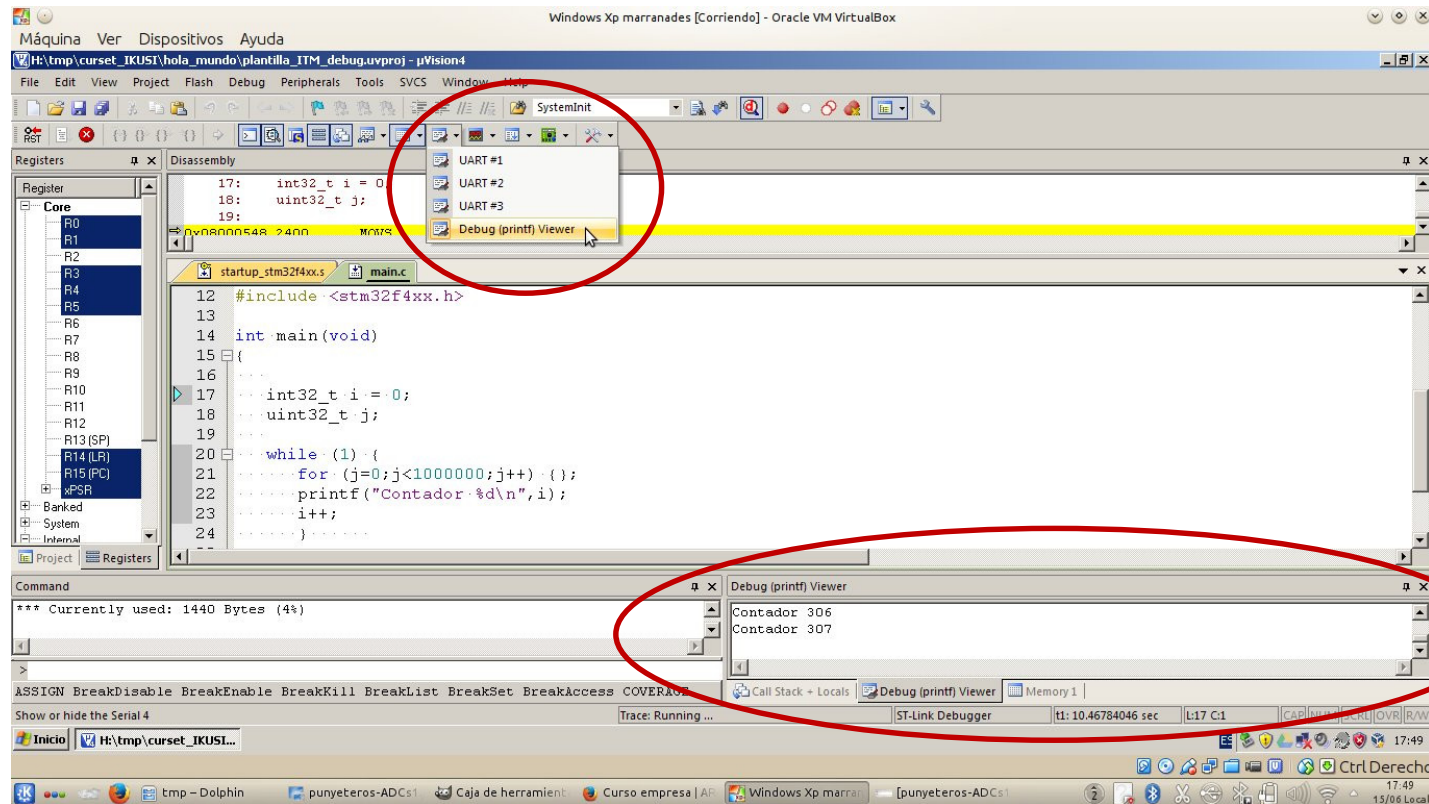
Load "pwmlcd\pwmlcd.axf"
BS \pwmlcd\..\Src/main.c:66
BS \pwmlcd\..\Src/servo.c:118
WS 1, 'capture_value
      
```
- Watch Window:** Shows a single watch item:

Name	Value	Type
capture_v...	<cannot evaluate>	uchar



Depuración: printf()

- La plantilla está preparada para usar SWO
 - Redirección 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

-O3

```

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 }
  
```

```

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)

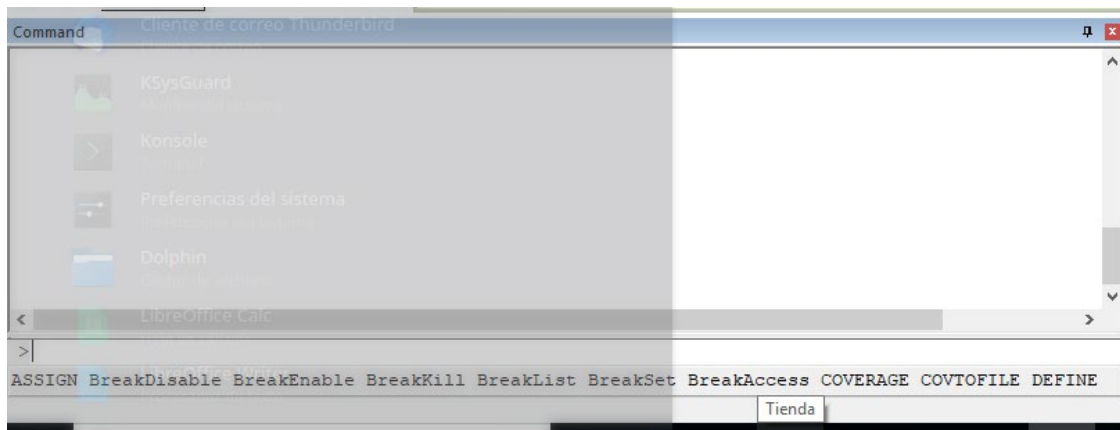


Depuración: step-by-step

- 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!

Name	Location/Value	Type
servo_SetPosition	0x08001E4C	unsigned char f(short)
tenth_degree	900	param - short
capture_value	2000	auto - unsigned int
main	0x00000000	int f()
error_code	0	auto - unsigned char

Call Stack + Locals | Trace Exceptions | Event Counters | Memory 1



Depuración: step-by-step

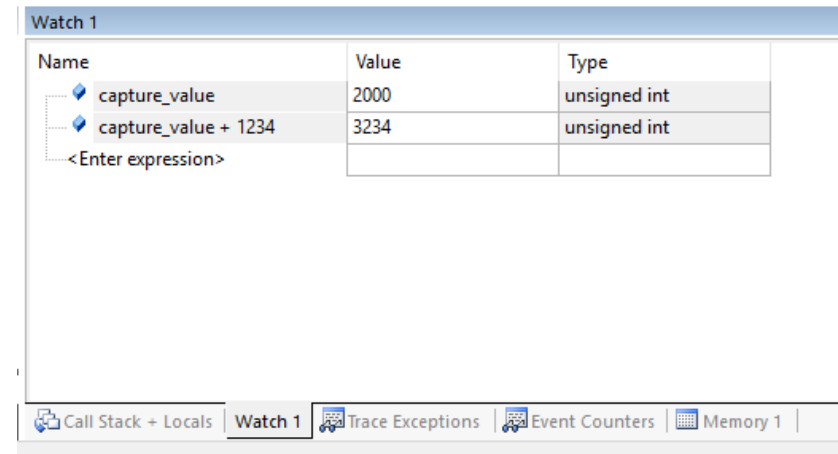
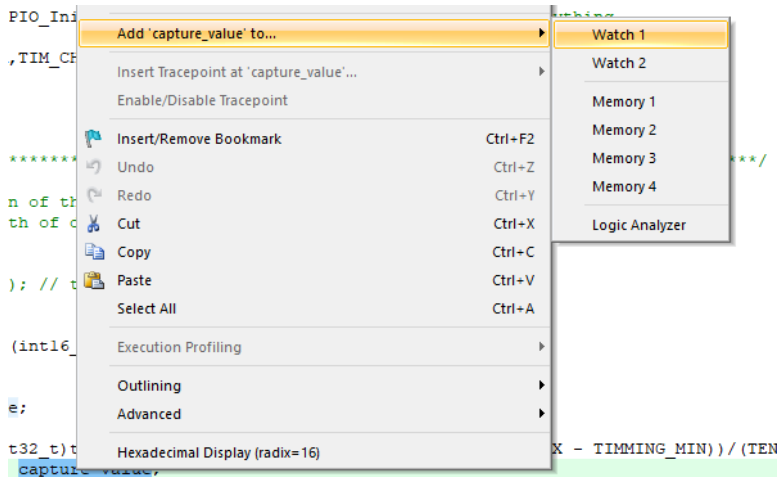
- Podemos ojear variables

```

lue;

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

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



Depuración: periféricos

- Estilo “Reference manual” (registros internos)

The image shows a debugger interface with two main panes. The left pane, titled 'System Viewer', displays a tree view of peripherals. The right pane, titled 'TIM2', shows a table of internal registers and their values.

Property	Value
CR1	0x00000001
CR2	0
SMCR	0
DIER	0
SR	0x0000001F
EGR	0
CCMR1_Output	0x00000068
CCMR1_Input	0x00000068
CCMR2_Output	0
CCMR2_Input	0
CCER	0x00000001
CNT	0x000046BD
CNT_H	0x0000
CNT_L	0x46BD
PSC	0x00000004
ARR	0x00004E20
CCR1	0x000007D0
CCR2	0
CCR3	0
CCR4	0
DCR	0
DMAR	0x00000001
OR	0



Depuración: trazas

- Próximo curso
- TO BE CONTINUED ...

Trace Exceptions

EXCTRC: Exception Tracing Timestamps Enable

#	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						

Call Stack + Locals | Watch 1 | Trace Exceptions | Event Counters | Memory 1





(Cajón de sastre)

- Bip, bip

