STM32H745 Timer & DMA & DAC

Setting up with STM32CubeIDE

Timer (TIM6)

Mode

- Runtime Context: Cortex-M7
- Activated: Yes (Checked)
- One Pulse Mode: No (Unchecked)

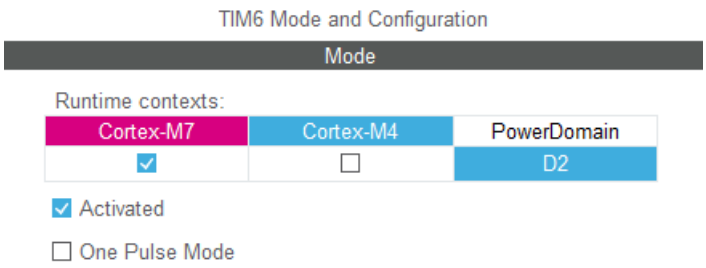

Figure 1: TIM6 Mode

Parameter and Settings

- Prescaler:480
- Counter Mode: Up
- Counter Period (AutoReload Register): 10
- Auto-Reload Preload: Disable
- Trigger Output (TRGO) Trigger Event Selection: Update Event

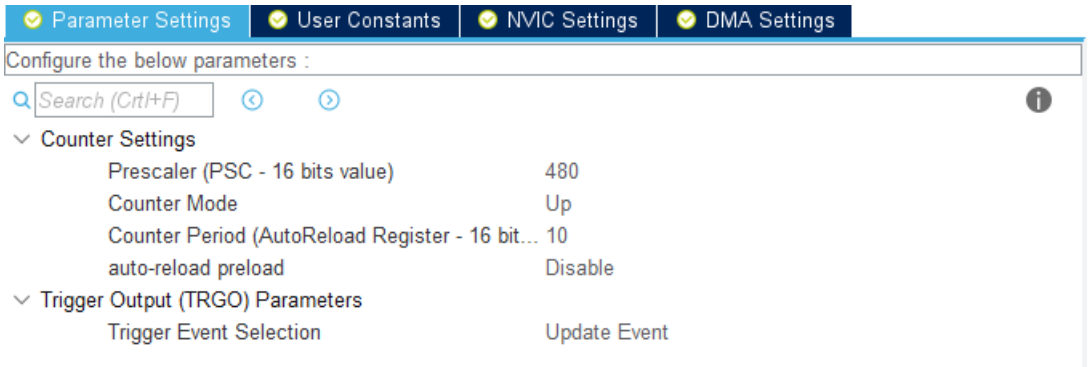

Figure 2: TIM6 Parameter

User Constants

None.

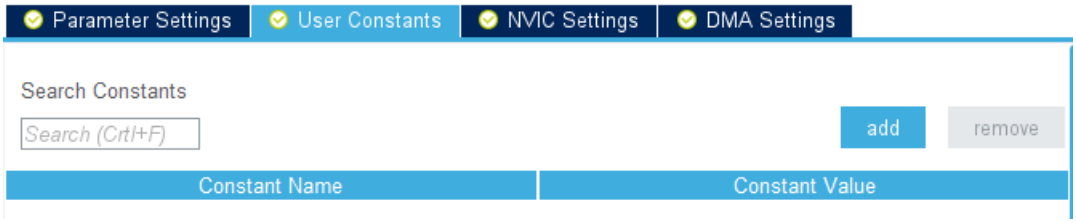

Figure 3: TIM6 User Constants

NVIC Settings

None.

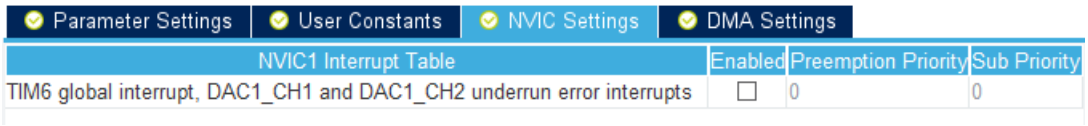

Figure 4: TIM6 NVIC Settings

DMA Settings

None.

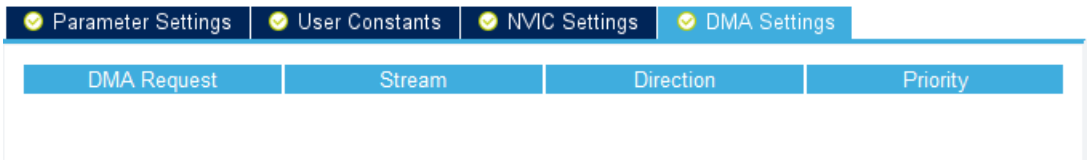

Figure 5: TIM6 DMA Settings

DAC (DAC1)

Mode

- Runtime Context: Cortex-M7
- OUT1 mode: Connected to external pin only
- OUT1 mode: Disable
- External Trigger: No (unchecked)

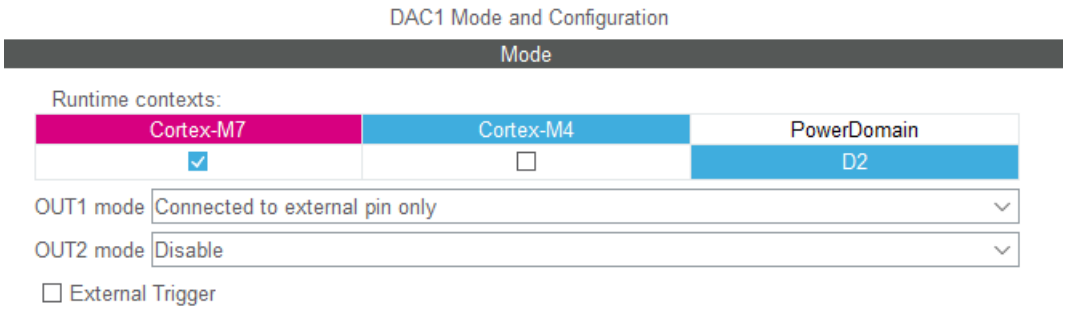

Figure 6: DAC1 Mode

Parameter Settings (DAC Out 1 Settings) :

- Output Buffer: Enable
- Trigger: Timer 6 Trigger Output Event
- Wave generation mode: Disabled
- User Trimming: Factory Trimming
- Sample and Hold: Disabled

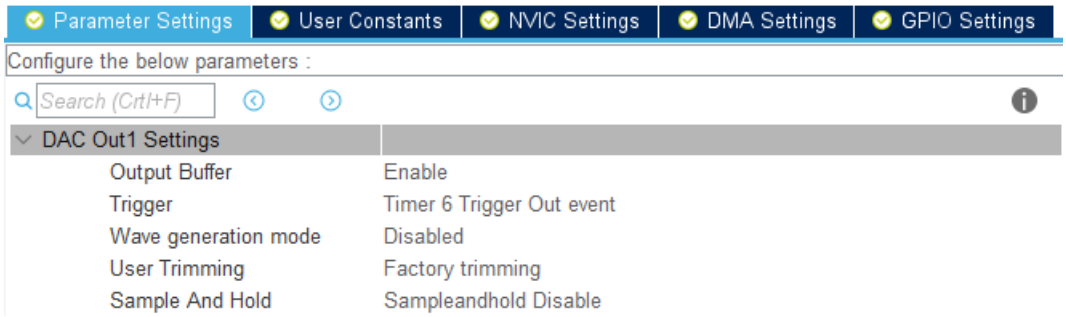

Figure 7: DAC1 Parameter Setting

User Constant :

None.

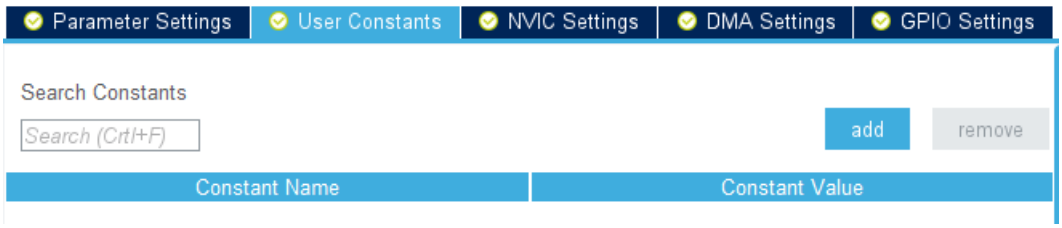

Figure 8: DAC1 User Constants

NVIC Settings:

- DMA1 stream0 global interrupt: Enabled, Preemption Priority = 0 ; Sub Priority 0
- TIM6 global interrupt, DAC1_CH1 and DAC1_CH2 underrun and error interrupts: Disabled, Preemption Priority = 0 ; Sub Priority 0

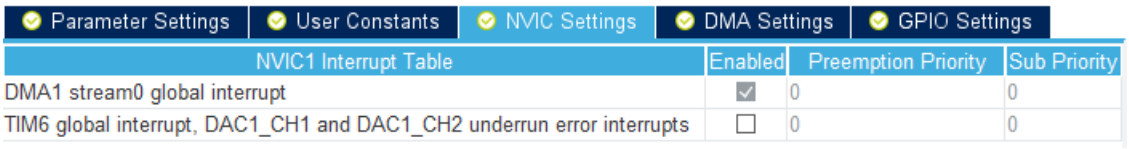

Figure 9: DAC1 NVIC Settings

DMA Settings

- One Channel:
  - DMA Request: DAC1_CH1 ;
  - Stream : DMA1 Stream 0 ;
  - Direction: Memory to Peripherial ;
  - Priority : Low

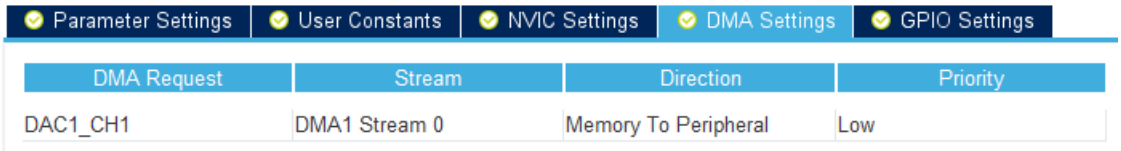

Figure 10: DAC1 DMA Settings

GPIO Settings

- DAC_CH1 Settings:
  - Pin Name: PA4
  - Signal on Pin: DAC1_OUT1
  - Pin Context Assignment: ARM Cortex M7
  - GPIO Output Level: n/a
  - GPIO Mode: Analog Mode
  - GPIO Pull-up/Pull-down: No Pull-up and no Pull-down
  - Maximum Output Speed: n/a
  - Fast Mode: n/a
  - User Label: Empty
  - Modified: Yes (Checked)

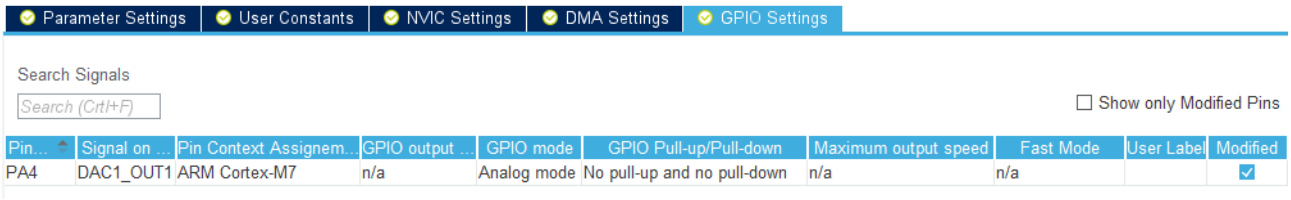

Figure 11: DAC1 GPIO Settings

DMA

Check if the Settings in the DMA Sections are the same as managed by the DAC-Section.

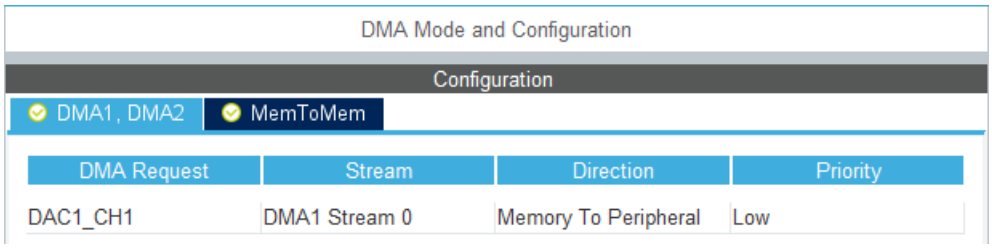

Figure 12: DMA Mode and Configuration

Memory Layout, Linker-Scripts, Startupcode(Memory Init) and Implementation

Memory Layout

The Memory coupled to the CPU is very fast, but can not be accessed by the DMA. Therefore the large AXI-Memory with it's 512kByte is used in this example, being a very comfortable size for a Waveform LUT.

- Address: 0x24000000 – 0x2407FFFF
- Size 512kByte

Figure 1. System architecture for STM32H745/55/47/57xx devices

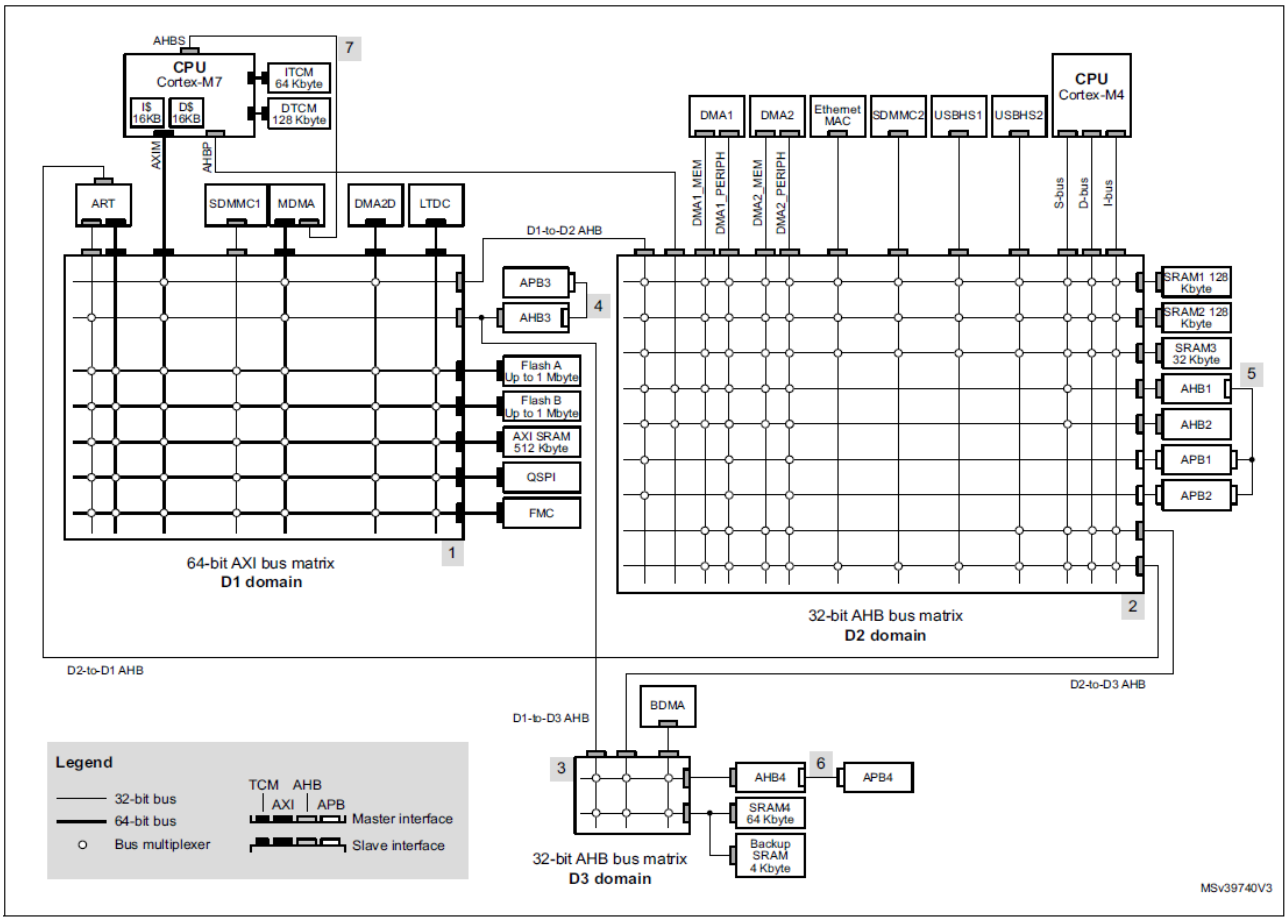

Figure 13: Reference Manual RM0399 Rev 3, Page 109

Linker Script

In this example the Code is loaded from Flash, so the Linkerfile that needs to be modified is STM32H745ZITX_FLASH.ld , which can be found at ..\workspace\Nucleo745_project\CM7\ STM32H745ZITX_FLASH.ld .

First the memory areas need to be defined, add the line marked in blue, you'll find it at approximately line 50, if you use the project default from the code-generator:

```
/* Specify the memory areas */
MEMORY
{
FLASH (rx)      : ORIGIN = 0x08000000, LENGTH = 1024K
RAM (xrw)      : ORIGIN = 0x20000000, LENGTH = 128K
ITCMRAM (xrw)      : ORIGIN = 0x00000000, LENGTH = 64K
RAM_D2 (xrw)      : ORIGIN = 0x30000000, LENGTH = 128K
RAM_AXI (xrw)      : ORIGIN = 0x24000000, LENGTH = 512K
}
```

Second, global symbols need to be defined and the object-files parsed for the default-values. You'll find it at approximately line 140, if you use the project default from the code-generator:

```
/* This is a custom section, created by TilenM, ST */
/* Let's have all DMA buffers in AXI domain. If buffers have default values, copy values from flash first */
_si_dma_data = LOADADDR(.dma_buffer);
.dma_buffer :
{
  . = ALIGN(4);
  _s_dma_data = .;  /* create a global symbol at data start */
  *(.dma_buffer)  /* Parse all object files, find any .dma_section and place it here */
  *(.dma_buffer*)  /* Parse all object files (*), find any sub dma_sections and place it here */
  _e_dma_data = .;  /* define a global symbol at data end */
} >RAM_AXI AT> FLASH  /* Add "AT> FLASH" to indicate copy from flash to RAM to initialize variables to def values */
```

Startup Code

If a variable is initialized in C, that requires the copying the values from Flash into the RAM. The file required to be modified is **startup_stm32h745zitx.s** , which can be found at ..\workspace\ Nucleo745_project\CM7\Core\Startup\startup_stm32h745zitx.s .

```
/* Copy the data segment initializers from flash to SRAM */
  movs  r1, #0
  b  LoopCopyDataInit

CopyDataInit:
  ldr  r3, =_sidata
  ldr  r3, [r3, r1]
  str  r3, [r0, r1]
  adds  r1, r1, #4

LoopCopyDataInit:
  ldr  r0, =_sdata
  ldr  r3, =_edata
  adds  r2, r0, r1
  cmp  r2, r3
  bcc  CopyDataInit

  movs  r1, #0
  b  LoopDMACopyDataInit
CopyDMADataInit:
  ldr  r3, =_si_dma_data
  ldr  r3, [r3, r1]
  str  r3, [r0, r1]
  adds  r1, r1, #4

LoopDMACopyDataInit:
  ldr  r0, =_s_dma_data
  ldr  r3, =_e_dma_data
  adds  r2, r0, r1
  cmp  r2, r3
  bcc  CopyDMADataInit

  ldr  r2, =_sbss
  b  LoopFillZerobss
/* Zero fill the bss segment. */
```

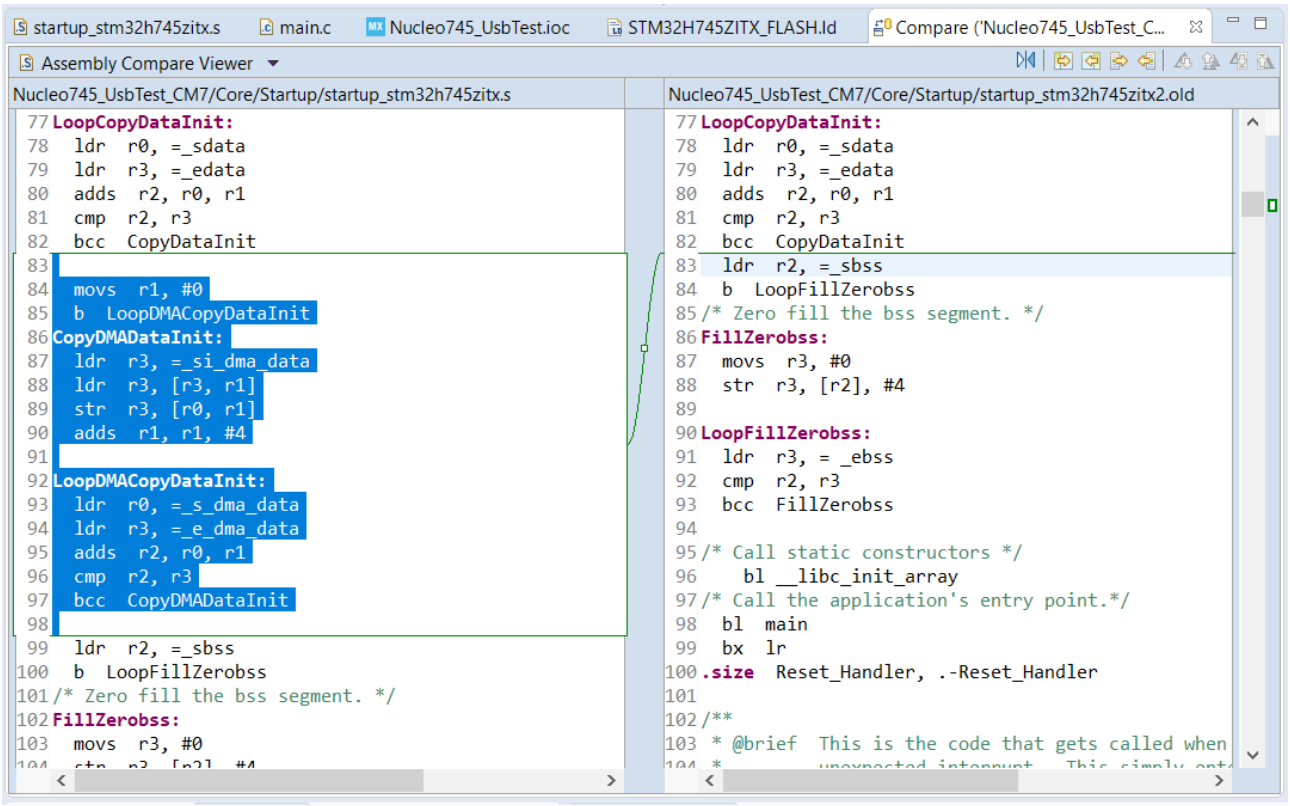

Figure 14: Diff of the startup_stm32h745zitx.s with added initialization.

Implementation

In this example the implementation is in main.c . In order to place a variable into the AXI it requires to mark it with the attribute `__attribute__((section(".dma_buffer")))` , in this example it is defined with DMA_Buffer.

```
/* Private macro
-------------------------------------------------------------*/
/* USER CODE BEGIN PM */

#if defined( __ICCARM__ )
  #define DMA_BUFFER \
      _Pragma("location=\".dma_buffer\"")
#else
  #define DMA_BUFFER \
      __attribute__((section(".dma_buffer")))
#endif

/* USER CODE END PM */
```

In this example we will place a Lookuptable in the AXI Section, it has NS values. The Sine-Table is longer, but in this example it is kept short. 0X07FF is 50% of the DAC's value if DAC_ALIGN_12B_R is used.

```
DMA_BUFFER uint16_t Wave_LUT[NS] = { 0x07FF,  0x086A, … 0x0794};
```

In the Main or another function the timer TIM6 needs to be started, it is automatically reloading(see the configuration done) , after that the DAC need to be started in DMA-Mode. Pleas note that these function return values and should be checked and an error-handling performed, depending you like to use that

```
HAL_TIM_Base_Start(&htim6);
HAL_DAC_Start_DMA(&hdac1, DAC_CHANNEL_1, (uint32_t*)Wave_LUT, NS, DAC_ALIGN_12B_R);
```

Check the Build

To see that the memory is set up correctly, checking the Build-Analyzer can help. To do that, select the elf-file(binary) and open the Build-Analyzer.

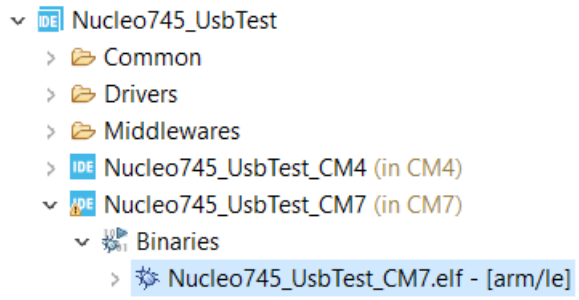

Figure 15: Mark the elf to analyze it.

In the Build-Analyzer you can check the correct implementation of the AXI memory and storing the default-values in the Flash.

Build Analyzer | Static Stack Analyzer | Debug | Search

Nucleo745_UsbTest_CM7.elf - /Nucleo745_UsbTest_CM7/Debug - Feb 24, 2021 9:20:25 AM

Memory Regions | Memory Details

Selection: 480 B

Search

Name	Run address (VMA)	Load address (LMA)	Size
ITCMRAM	0x00000000		64 KB
FLASH	0x08000000		1024 KB
.isr_vector	0x08000000	0x08000000	664 B
.text	0x080002a0	0x080002a0	102,34 KB
.rodata	0x08019c00	0x08019c00	3,7 KB
.ARM	0x0801aad0	0x0801aad0	8 B
.init_array	0x0801aad8	0x0801aad8	4 B
.fini_array	0x0801aadc	0x0801aadc	4 B
.data	0x20000000	0x0801aae0	916 B
.RxDecripSection	0x20000394	0x0801ae74	96 B
.TxDecripSection	0x200003f4	0x0801aed4	96 B
.dma_buffer	0x24000000	0x0801af34	480 B
RAM	0x20000000		128 KB
.data	0x20000000	0x0801aae0	916 B
.RxDecripSection	0x20000394	0x0801ae74	96 B
.TxDecripSection	0x200003f4	0x0801aed4	96 B
.bss	0x20000454		23,2 KB
._user_heap_stack	0x20006124		8 KB
RAM_AXI	0x24000000		512 KB
.dma_buffer	0x24000000	0x0801af34	480 B
RAM_D2	0x30000000		128 KB

Figure 16: Build Analyzer