ECEN 5803

Mastering Embedded System Architecture

(Fall-2023)

PROJECT-1 : MODULE 4

AUTO-CONFIGURE WITH STMCUBE

Date

10/12/2023

Guided By

Prof. Timothy Scherr

Submitted By

Ajay Kandagal

Kshitija Dhondage

1. Create start-up code using the STMCUBE configuration software, found here: https://www.st.com/en/development-tools/stm32cubemx.html Be sure to set the system clock to 84 MHz and the ADC sample clock to 100 kHz.

Below are the steps to create a project with above mentioned configuration using STM32CueMX.

1. Create a new project by selecting Nucleo-F401RE board.
2. By default, the clocks were set to 84MHz and hence there was no need to make any changes in the clock tree.

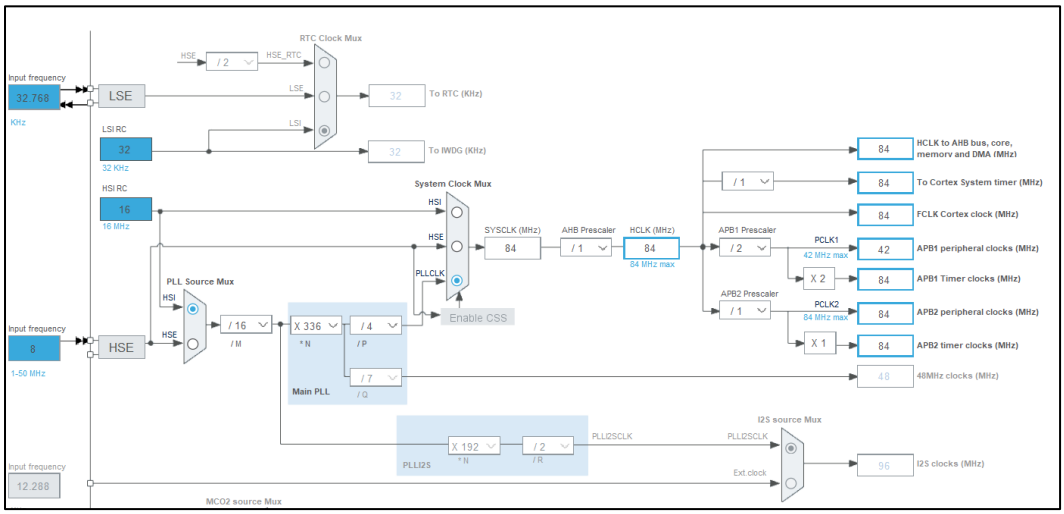

Figure 1: Clock Configuration set at 84MHz using STM32CubeMX GUI

3. To set the ADC sampling rate at 100kHz, select the “Pin & Configuration” under this select “Timers” and then TIM2. The clock source is set to Internal Clock. And to generate timer OV evets at 100kHz to trigger ADC following settings are used.

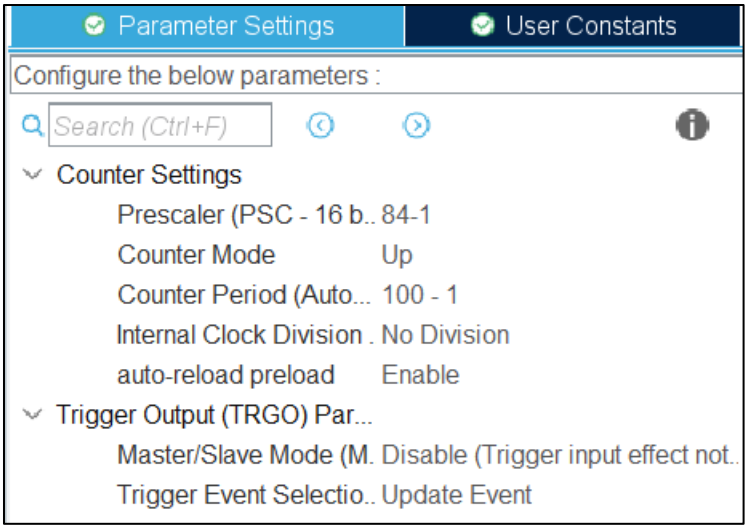

Figure 2: Timer 2 setup to trigger ADC at 100kHz.

100kHz = 84MHz / (Pre-scaler * Counter Period)

4. Next set the PA0 to ADC1_IN0 from “Pinout View” GUI
5. To configure ADC, select “Analog” and then ADC1. Check IN0 and set the following parameters to set the ADC resolution to 12-bit and trigger source as TIM2.

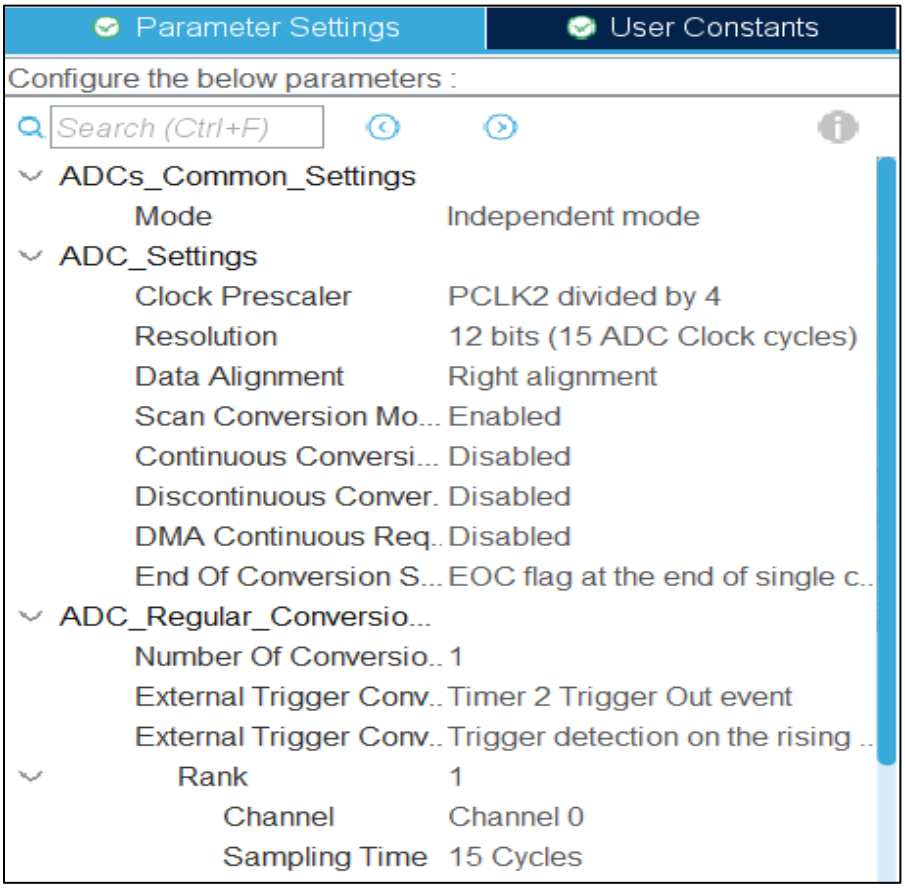

Figure 3: ADC resolution set to 12-bit and external trigger set to TIM2 events.

Also enable the global NVIC for ADC to sample the ADC at 100kHz.

6. In “Project Manager”, project name is assigned, and Tool Chain is set to MDK-ARM. Generate code and then project can be opened in Keil.

Compare this to the startup code you used in the other modules so far. How is it different? Provide a memory map for each.

On pressing the "reset" button in the IDE, the code jumps to address 0x08000234 which is the start address of Reset_Handler. The instruction contains call to the system_init and __main.

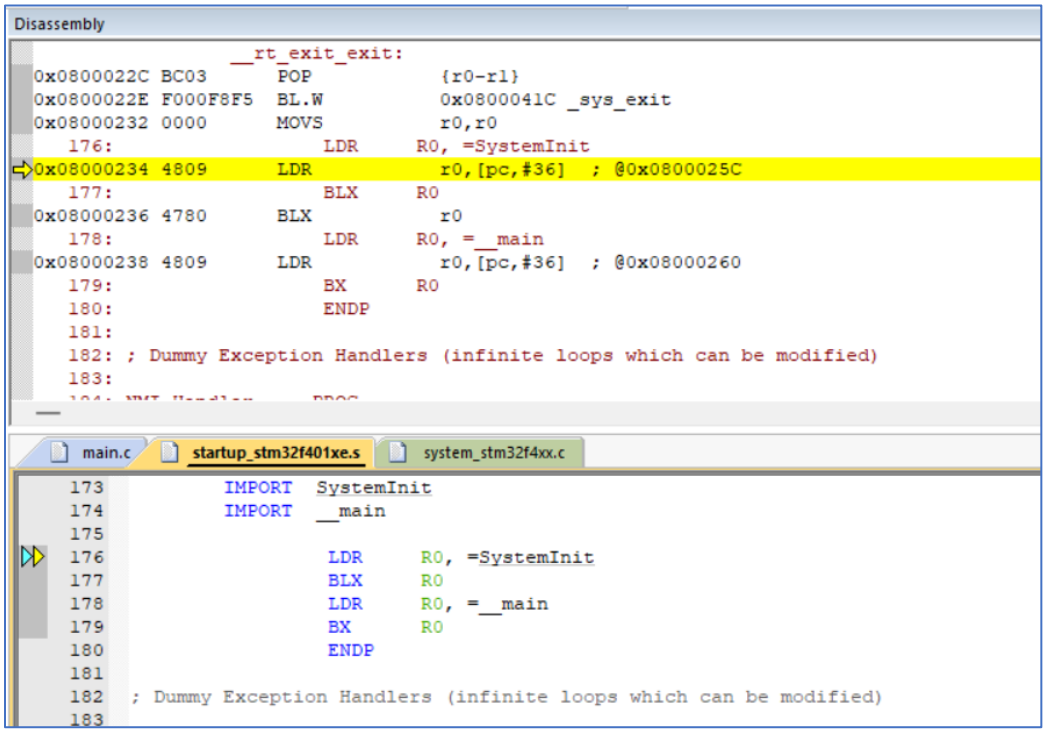

Figure 4: Code generated using STM32CubeMx, code after reset.

The sequence in Module 2 is the same but the Reset_Handler location is different and is located at address 0x08000318.

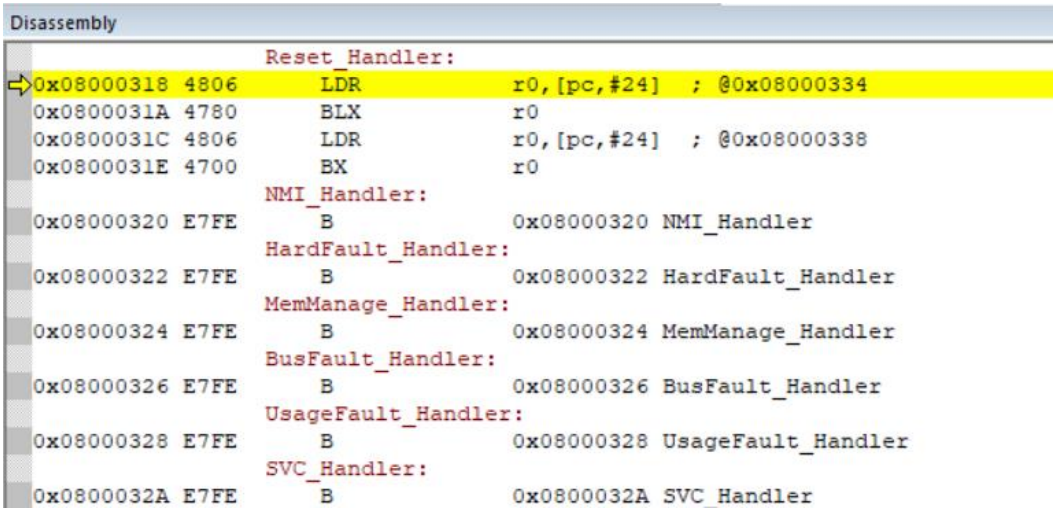

Figure 5: Audio Code from Module 2 Audio Example, code after reset

```
.constdata                              0x08001a0a   Section       16  system_stm32f4xx.o(.constdata)
.constdata                              0x08001a1a   Section        8  system_stm32f4xx.o(.constdata)
.data                                   0x20000000   Section        4  main.o(.data)
.data                                   0x20000004   Section       12  stm32f4xx_hal.o(.data)
.data                                   0x20000010   Section        4  system_stm32f4xx.o(.data)
.bss                                    0x20000014   Section      212  main.o(.bss)
.bss                                    0x200000e8   Section       96  libspace.o(.bss)
HEAP                                    0x20000148   Section      512  startup_stm32f401xe.o(HEAP)
Heap_Mem                                0x20000148   Data         512  startup_stm32f401xe.o(HEAP)
STACK                                   0x20000348   Section     1024  startup_stm32f401xe.o(STACK)
Stack_Mem                               0x20000348   Data        1024  startup_stm32f401xe.o(STACK)
__initial_sp                            0x20000748   Data           0  startup_stm32f401xe.o(STACK)
```

Figure 6: Code generated using STM32CubeMx, memory initialization.

```
__heap_base                             0x20000428   Data           0  startup_stm32f401xe.o(HEAP)
Image$$RW_IRAM1$$ZI$$Limit              0x20000828   Number         0  anon$$obj.o ABSOLUTE
__heap_limit                            0x20000828   Data           0  startup_stm32f401xe.o(HEAP)
__initial_sp                            0x20018000   Number         0  startup_stm32f401xe.o ABSOLUTE
```

Figure 7: Code from Module2 Audio example

In the STM32CubeMx generated code, the heap base is at 0x20000148 and in the audio code of Module 2, it is at 0x20000428. In CubeMx code, the heap size is mentioned as 512 bytes, and if we do the calculation for the module 2 code by subtracting the __heap_base with __heap_limit we get 0x400 which is 1024 bytes.

```
==============================================================================

    Code (inc. data)   RO Data    RW Data    ZI Data      Debug

    6262        420        462         20       1844     517986   Grand Totals
    6262        420        462         20       1844     517986   ELF Image Totals
    6262        420        462         20          0          0   ROM Totals

==============================================================================

    Total RO  Size (Code + RO Data)                 6724 (   6.57kB)
    Total RW  Size (RW Data + ZI Data)              1864 (   1.82kB)
    Total ROM Size (Code + RO Data + RW Data)       6744 (   6.59kB)

==============================================================================
```

Figure 8: Map file of code generated using STM32CubeMx.

```
================================================================================

    Code (inc. data)   RO Data    RW Data    ZI Data      Debug

    15620       1260       1900         72       1612     163982   Grand Totals
    15620       1260       1900         72       1612     163982   ELF Image Totals
    15620       1260       1900         72          0          0   ROM Totals

================================================================================

    Total RO  Size (Code + RO Data)                17520 (  17.11kB)
    Total RW  Size (RW Data + ZI Data)              1684 (   1.64kB)
    Total ROM Size (Code + RO Data + RW Data)      17592 (  17.18kB)

================================================================================
```

Figure 9: Map file of code from Module2 Audio example

Comparing the map files generated for the two codes, it is evident that the Module 2 code uses more code memory (around 15.6 kB), whereas the code generated by CubeMx takes only 6.2 kB. This might be because the Audio code example uses more Hal modules for PWM, serial and Analog because of which the library takes additional space. The library in Module 2 example takes up 10.3kB and in CubeMx, it is 880 bytes. However, the RAM usage in the CubeMx code is higher even with very few user variables used.