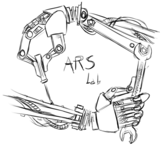
The Startup Sequence of STM32

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
`santoro@dmi.unict.it`



L.S.M. Course

Startup and “main” function

- When an STM32 MCU is powered-on, it **does not** execute immediatelly the **main()** function
- A boot sequence is instead activated with includes the execution of some intialization code
- At the end of the boot sequence, the main() function is finally run

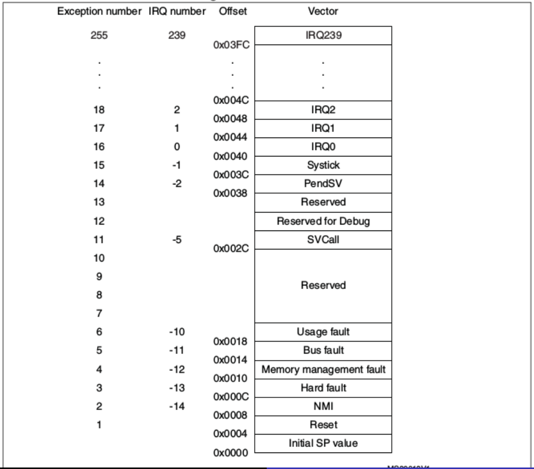
The “startup.s” file

- The real program that executes at power-on is placed in a startup assembly source file called **startup_stm32f401xe.s**
- It contains:
  - A code that prepares the memory to run the user program
  - The definition of **interrupt vectors**
- Indeed, everything starts from the definitions placed in the **interrupt vector table**

The Interrupt Vector Table

- The **Interrupt Vector Table** is a region of the flash memory starting at a fixed address, for the STM32F4 is `0x0800 0000`
- It contains 32-bit word elements, each one specifying the a jump address to handle a specific interrupt

Figure 11. Vector table



Exception number	IRQ number	Offset	Vector
255	239	0x03FC	IRQ239
. . .		. . . 0x004C	. . .
18	2	0x0048	IRQ2
17	1	0x0044	IRQ1
16	0	0x0040	IRQ0
15	-1	0x003C	Systick
14	-2	0x0038	PendSV
13			Reserved
12			Reserved for Debug
11	-5	0x002C	SVCall
10			Reserved
9			
8			
7			
6	-10	0x0018	Usage fault
5	-11	0x0014	Bus fault
4	-12	0x0010	Memory management fault
3	-13	0x000C	Hard fault
2	-14	0x0008	NMI
1		0x0004	Reset
		0x0000	Initial SP value

The Interrupt Vector Table and the Startup File

- The startup file **startup_stm32f401xe.s** includes a section that defines the interrupt vector table:

```
    .section  .isr_vector,"a",%progbits

   .word  _estack
   .word  Reset_Handler
   .word  NMI_Handler
   .word  HardFault_Handler
   .word  MemManage_Handler
   .word  BusFault_Handler
   .word  UsageFault_Handler
   .word  0
   .word  0
   .word  0
   .word  0
   .word  SVC_Handler
   .word  DebugMon_Handler
   .word  0
   .word  PendSV_Handler
   .word  SysTick_Handler
...
```

- The first code executed at startup is thus referred by the label **Reset_Handler**

The Reset Handler

- The code of the **Reset_Handler** includes a part that prepares the memory (it copies into RAM the initial values of the variables) and then calls (in sequence):
    - The **`SystemInit`** function
    - The **`__libc_init_array`** function
    - The **`main`** function (finally!)

```
    .section  .text.Reset_Handler
...
Reset_Handler:
    ....
    bl  SystemInit /* Call the clock system intitialization function.*/

    bl __libc_init_array /* Call static constructors */

    bl  main /* Call the application's entry point.*/
```

The startup functions

- **SystemInit** is a user function that has the role of configuring the **clock** of the processor
- It is placed in the source file `system_init.c` of the stm32_unict_lib

- **__libc_init_array** is a library function that initializes all the structures needed by the **libc**
- It is placed in the source files of the libc

The Startup Sequence of STM32

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
`santoro@dmi.unict.it`

L.S.M. Course