1/8/2022

Embedded C

Lab (2)

Reham Nady Abd Elmotaal

1-Introduction:

Write a baremetal software to toggle led which connected to GPIO port A13 in Stm32f103CX micro-controller chip . I will build everything from scratch including startup ,linker script and source codes ,and compile them using arm cross tool chain.

2-Source codes:

2.1:main.c

To make a GPIO toggling in STM32, you need to work with two peripherals:

1-RCC (reset and clock control):

The RCC is necessary because the GPIO has disabled clock by default.

2- GPIOx (general purpose input/output).

we make union to each register to give availability of access whole register or access bit by bit .

```
typedef union{
  vuint32_t allport;
  struct {…
  }Sbits;
}APB2ENR_t;

typedef union{
  vuint32_t allport;
  struct {…
  }Sbits;
}CRH_t;

typedef union{
  vuint32_t allport;
  struct {
    vuint32_t reserve :13 ;
    vuint32_t p_13     :1 ;
  }Sbits;
}GPIOA_ODR_t;
```

2.2: uart.c:

this function received char by char and then transmit it using UARTDR register.

3-Startup:

startup written in assembly:

In this code we make :

1-vector section: Define Interrupt vectors Section

2- reset section :in this section I just brunch to main function.

```
4   .section .vectors
5
6   .word _stack_top
7   .word _reset
8   .word Vector_handler    /* 2 NMI*/
9   .word Vector_handler    /* 3 Hard Fault*/
10  .word Vector_handler    /* 4 NM Fault*/
11  .word Vector_handler    /*5 Bus Fault*/
12  .word Vector_handler    /*6 usage fault*/
13  .word Vector_handler    /*7 Reserved*/
14  .word Vector_handler    /*8 reserved*/
15  .word Vector_handler    /*9 reserved*/
16  .word Vector_handler    /*10 reserved*/
17  .word Vector_handler
18  .word Vector_handler
19  .word Vector_handler
20  .word Vector_handler
21  .word Vector_handler
22  .word Vector_handler
23  .word Vector_handler
24  .word Vector_handler
25  .word Vector_handler
26
27  .section .text
28  _reset:
29    bl main
30    b .
31  .thumb_func
32  Vector_handler:
33   b _reset
```

4-Linker script:

In this linker script, we define memory boundaries ,in this app we have just one memory . the last section in linker used to divide my code in all file and organize it to burn it on the micro controller.

```
MEMORY
{
  flash (rx) : ORIGIN = 0x08000000, LENGTH = 128k
  sram (rwx) : ORIGIN = 0x20000000, LENGTH = 20k
}
SECTIONS
{
  .text : {
    *(.vectors*)
    *(.text*)
    *(.rodata)

  }> flash
  .data : {

    *(.data*)

  }>sram AT> flash
  .bss : {

    *(.bss*)

  }>sram
  . = . + 0x1000 ;
  _stack_top = . ;

}
```

5-Symbols:

5.1:symbol of main.o:

```
$ arm-none-eabi-nm.exe  main.o
00000000 D APB2ENR
00000000 R const_variable
00000004 D CRH
0000000c D g_variable
00000000 T main
00000008 D R_ODR
```

1- APb2ENR : which in data section.

2- const_variable: which in readonly data section.

3- CRH : which in data section.

4- g_variable : which in data section.

5- main: which in text section

6-R_ODR: : which in data section.

5.3:symbol of startup.o:

```
$ arm-none-eabi-nm.exe  startup.o
00000000 t _reset
         U _stack_top
         U main
00000006 t Vector_handler
```

1- reset: which in text section.

2-stack_top: unresolved symbol and will be resolved during Linking process.

3- main: unresolved symbol and will be resolved during Linking process.

4- vector_handler: which in text section.

5.4:elf image sympols:

```
$ arm-none-eabi-nm.exe  learn.elf
08000108 t _reset
20001010 D _stack_top
20000000 D APB2ENR
08000110 T const_variable
20000004 D CRH
2000000c D g_variable
08000050 T main
20000008 D R_ODR
0800010e t Vector_handler
```

1- _reset: which in text section.

2- stack_top: which in data section.

3-APB2ENR :which in data section.

4- const_variable: which in text section.

5- CRH: which in data section.

6- g_variable : which in data section.

7- main: which in text section.

8-R_ODR: which in data section.

9- vector_handler : which in text section.

6-Sections Headers:
6.1: main.o sections headers

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000b8  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000010  00000000  00000000  000000ec  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000fc  2**0
                  ALLOC
  3 .rodata       00000004  00000000  00000000  000000fc  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .debug_info   00000271  00000000  00000000  00000100  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  5 .debug_abbrev 000000fe  00000000  00000000  00000371  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_loc    00000038  00000000  00000000  0000046f  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_aranges 00000020  00000000  00000000  000004a7  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_line   000000a4  00000000  00000000  000004c7  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  9 .debug_str    0000018d  00000000  00000000  0000056b  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .comment      00000012  00000000  00000000  000006f8  2**0
                  CONTENTS, READONLY
 11 .ARM.attributes 00000033  00000000  00000000  0000070a  2**0
                  CONTENTS, READONLY
 12 .debug_frame  0000002c  00000000  00000000  00000740  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

1-text section: size of instruction code =0xb8.

2-data section: size of initialized global array = 0x10 .

3-bss section: size of uninitialized global =0x0.

4-rodata section : size of constant data =0x04.

5-debug sections and other sections.

6.2: startup.o sections headers

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000008  00000000  00000000  00000034  2**1
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  0000003c  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  0000003c  2**0
                  ALLOC
  3 .vectors      00000050  00000000  00000000  0000003c  2**0
                  CONTENTS, RELOC, READONLY
  4 .ARM.attributes 00000021  00000000  00000000  0000008c  2**0
                  CONTENTS, READONLY
  5 .debug_line   0000003b  00000000  00000000  000000ad  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  6 .debug_info   00000091  00000000  00000000  000000e8  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_abbrev 00000014  00000000  00000000  00000179  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_aranges 00000020  00000000  00000000  00000190  2**3
                  CONTENTS, RELOC, READONLY, DEBUGGING
```

1-text section: size of instruction code =0x8.

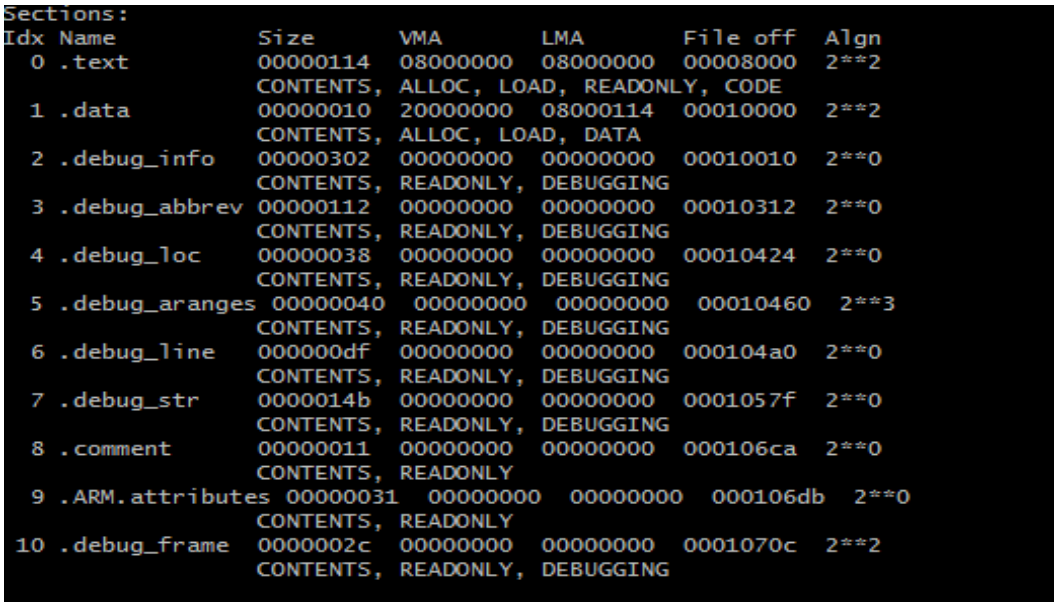
2-data section: size of initialized global array = 0x0 .

3-bss section: size of uninitialized global =0x0.

4-vectors section : size of constant data =0x50.

5-debug sections and other sections.

6.3: elf image sections headers



```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000114  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000010  20000000  08000114  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .debug_info   00000302  00000000  00000000  00010010  2**0
                  CONTENTS, READONLY, DEBUGGING
  3 .debug_abbrev 00000112  00000000  00000000  00010312  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_loc    00000038  00000000  00000000  00010424  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_aranges 00000040  00000000  00000000  00010460  2**3
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_line   000000df  00000000  00000000  000104a0  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_str    0000014b  00000000  00000000  0001057f  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .comment      00000011  00000000  00000000  000106ca  2**0
                  CONTENTS, READONLY
  9 .ARM.attributes 00000031  00000000  00000000  000106db  2**0
                  CONTENTS, READONLY
 10 .debug_frame  0000002c  00000000  00000000  0001070c  2**2
                  CONTENTS, READONLY, DEBUGGING
```

1-text section: size of instruction code =0x114.

2-data section: size of initialized global array = 0x10.

3- debug sections and other section.