MCU bare metal firmware boot process without boot loader

https://embeddedsecurity.io/sec-stm32i-firmware.html

Using stm32L152RE (ARM Cortex M3) as an example for description.

prerequisite:

The compiler creates objects files for the application code.

Then linker uses the link script file .ld to link these object files aling with the startup file .s.

The .ld file defines the memory size and their starting location:

/\* Entry Point \*/

ENTRY(Reset\_Handler)

/\* Highest address of the user mode stack \*/

\_estack = ORIGIN(RAM) + LENGTH(RAM); /\* end of "RAM" Ram type memory \*/

\_Min\_Heap\_Size = 0x200; /\* required amount of heap \*/

\_Min\_Stack\_Size = 0x400; /\* required amount of stack \*/

/\* Memories definition \*/

MEMORY

{

RAM (xrw) : ORIGIN = 0x20000000, LENGTH = 80K

FLASH (rx) : ORIGIN = 0x8000000, LENGTH = 512K

}

This indicates the highest memory location of RAM is 0x20014000, which is the top of the stack.

M3, M4 use a ful descending stack.

The linker maps this initial sp value to the very first address (0) of the entire 32-bit memory space.

On Reset the MCU loads the Main Stack Pointer value 0x20014000 from address 0.

Each address increment in 32-bit MCU is 4 bytes.

At power on reset (or button reset) the cortex M3, M4 program counter is loaded with the first address 0.

This address contains the initial stack pointer value pointing to the top of stack (0x20014000).

The MCU fetches the top-of-stack value from this address 0.

Then it starts code execution from the boot memory (flash by boot0 pin == low) from the address 0x00000004.

The M3, M4 datasheet has the definition of its vector table from the first memory address (0x00000000) upward.

And datasheet lists the RESET service routine is fixed at this address 0x00000004.

(system using boot loader will have different vector table and is relocated else where).

The first thing the MCU come out of hardware reset is to pop the initial stack pointer value

out of the Vector table, at address 0, then load it into the SP register.

Next the 2nd entry in the vector table which is the Reset Handler routine is called.

At compiled time the linker file specifies the ENTRY point that contains the address value of the RESET vector.

The RESET vector is one entry of the vector table assigned in the memory address map.

At run time start the hardware reset causes the MCU to look at a special memory location RESET vector address.

The instructions in this vector is then loaded and executed.

These instructions specify on how to start up the MCU.

Mostly occurs next is to copy the vector table from flash to RAM so that the table can be accessed

and executed quicker than from the flash. This arrangement strives to reduce the interrupt calls latency.

Next the data and the bss segments are copied to the RAM.

If the linker-file specifies any RAM functions then they are copied to the RAM.

It follows with calling the SystemInit function:

bl SystemInit

Finally the boot process jumps to the main function.

If any unexpected event occurred then it jumps to special handler

**Default\_Handler:**

**Infinite\_Loop:**

b Infinite\_Loop

Examine the detail for M3 on global vs. local variable and with static modifier --

Summary --

local regular variables are stored in the code (.text) segment.

global regular or static variables are stored in .bss if not initialized (including 0),

and in .data if initialized to non 0.

In the L152RE\_SPI.map file:

Memory Configuration

Name Origin Length Attributes

RAM 0x0000000020000000 0x0000000000014000 xrw

FLASH 0x0000000008000000 0x0000000000080000 xr

\*default\* 0x0000000000000000 0xffffffffffffffff

The address value can be found in the L152 programming reference

regular global variable within a file:

case 1.

int my\_var;

compilation output:

text data bss dec hex filename

9524 20 1732 11276 2c0c L152RE\_SPI.elf

in L152RE\_SPI.map file:

Allocating common symbols

Common symbol size file

hspi2 0x58 ./Core/Src/main.o

mem\_cmd 0x1 ./Core/Src/main.o

my\_var 0x4 ./Core/Src/main.o

.bss 0x000000002000000c 0xc4 load address 0x0000000008002548

0x000000002000000c \_sbss = .

0x000000002000000c \_\_bss\_start\_\_ = \_sbss

\*(.bss)

.bss 0x000000002000000c 0x1c c:/st/stm32cubeide\_1.6.1/stm32cubeide/plugins/com.st.stm32cube.ide.mcu.externaltools.gnu-tools-for-stm32.9-2020-q2-update.win32\_2.0.0.202105311346/tools/bin/../lib/gcc/arm-none-eabi/9.3.1/thumb/v7-m/nofp/crtbegin.o

\*(.bss\*)

\*(COMMON)

COMMON 0x0000000020000028 0xa4 ./Core/Src/main.o

0x0000000020000028 hspi2

0x0000000020000080 mem\_cmd

0x0000000020000084 my\_var

case 2.

int my\_var = 1;

compilation output shows the same.

in L152RE\_SPI.map file:

.data.my\_var 0x0000000000000000 0x4 ./Core/Src/main.o

case 3.

int my\_var = 0;

compilation output shows the same.

in L152RE\_SPI.map file:

.bss.my\_var 0x0000000000000000 0x4 ./Core/Src/main.o

static gloable variable:

case 1.

static int my\_var;

// or static int my\_var = 0;

compilation output shows the same.

in L152RE\_SPI.map file:

.bss.my\_var 0x0000000000000000 0x4 ./Core/Src/main.o

my\_var is excluded from the Allocating common symbols group

local static variable within a function:

int main(void)

{

static int my\_var;

}

compilation output shows the same.

in L152RE\_SPI.map file:

.bss.my\_var.5928

my\_var is excluded from the Allocating common symbols group

local regular variable (initialized nor not) within a function:

compilation output:

text data bss dec hex filename

9528 20 1732 11280 2c10 L152RE\_SPI.elf

in L152RE\_SPI.lst file:

int main(void)

{

8000474: b580 push {r7, lr}

8000476: b0ac sub sp, #176 ; 0xb0

8000478: af00 add r7, sp, #0

/\* USER CODE BEGIN 1 \*/

int my\_var = 0x12;

}

my\_var is excluded in both .data and .bss segment in map file.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file startup\_stm32l152xe.s

\* @author MCD Application Team

\* @brief STM32L152XE Devices vector table for GCC toolchain.

\* This module performs:

\* - Set the initial SP

\* - Set the initial PC == Reset\_Handler,

\* - Set the vector table entries with the exceptions ISR address

\* - Configure the clock system

\* - Branches to main in the C library (which eventually

\* calls main()).

\* After Reset the Cortex-M3 processor is in Thread mode,

\* priority is Privileged, and the Stack is set to Main.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

