STM32RAMid :

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

\*\*

\*\* @file : debug in RAM dedicated

\*\*

\*\* @author : STM32CubeIDE

\*\*

\*\* Abstract : Linker script for STM32F407G-DISC1 Board embedding STM32F407VGTx Device from stm32f4 series

\*\* 1024Kbytes FLASH

\*\* 64Kbytes CCMRAM

\*\* 128Kbytes RAM

\*\*

\*\* Set heap size, stack size and stack location according

\*\* to application requirements.

\*\*

\*\* Set memory bank area and size if external memory is used

\*\*

\*\* Target : STMicroelectronics STM32

\*\*

\*\* Distribution: The file is distributed as is, without any warranty

\*\* of any kind.

\*\*

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

\*\* @attention

\*\*

\*\* Copyright (c) 2022 STMicroelectronics.

\*\* All rights reserved.

\*\*

\*\* This software is licensed under terms that can be found in the LICENSE file

\*\* in the root directory of this software component.

\*\* If no LICENSE file comes with this software, it is provided AS-IS.

\*\*

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

\*/

/\* 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

{

CCMRAM (xrw) : ORIGIN = 0x10000000, LENGTH = 64K

RAM (xrw) : ORIGIN = 0x20000000, LENGTH = 128K

FLASH (rx) : ORIGIN = 0x8000000, LENGTH = 1024K

}

/\* Sections \*/

SECTIONS

{

/\* The startup code into "RAM" Ram type memory \*/

.isr\_vector :

{

. = ALIGN(4);

KEEP((.isr\_vector)) / Startup code \*/

. = ALIGN(4);

} >RAM

/\* The program code and other data into "RAM" Ram type memory \*/

.text :

{

. = ALIGN(4);

(.text) / .text sections (code) \*/

(.text) /\* .text\* sections (code) \*/

(.glue\_7) / glue arm to thumb code \*/

(.glue\_7t) / glue thumb to arm code \*/

\*(.eh\_frame)

(.RamFunc) / .RamFunc sections \*/

(.RamFunc) /\* .RamFunc\* sections \*/

KEEP (\*(.init))

KEEP (\*(.fini))

. = ALIGN(4);

\_etext = .; /\* define a global symbols at end of code \*/

} >RAM

/\* Constant data into "RAM" Ram type memory \*/

.rodata :

{

. = ALIGN(4);

(.rodata) / .rodata sections (constants, strings, etc.) \*/

(.rodata) /\* .rodata\* sections (constants, strings, etc.) \*/

. = ALIGN(4);

} >RAM

.ARM.extab : {

. = ALIGN(4);

(.ARM.extab .gnu.linkonce.armextab.\*)

. = ALIGN(4);

} >RAM

.ARM : {

. = ALIGN(4);

\_\_exidx\_start = .;

(.ARM.exidx)

\_\_exidx\_end = .;

. = ALIGN(4);

} >RAM

.preinit\_array :

{

. = ALIGN(4);

PROVIDE\_HIDDEN (\_\_preinit\_array\_start = .);

KEEP ((.preinit\_array))

PROVIDE\_HIDDEN (\_\_preinit\_array\_end = .);

. = ALIGN(4);

} >RAM

.init\_array :

{

. = ALIGN(4);

PROVIDE\_HIDDEN (\_\_init\_array\_start = .);

KEEP ((SORT(.init\_array.)))

KEEP ((.init\_array))

PROVIDE\_HIDDEN (\_\_init\_array\_end = .);

. = ALIGN(4);

} >RAM

.fini\_array :

{

. = ALIGN(4);

PROVIDE\_HIDDEN (\_\_fini\_array\_start = .);

KEEP ((SORT(.fini\_array.)))

KEEP ((.fini\_array))

PROVIDE\_HIDDEN (\_\_fini\_array\_end = .);

. = ALIGN(4);

} >RAM

/\* Used by the startup to initialize data \*/

\_sidata = LOADADDR(.data);

/\* Initialized data sections into "RAM" Ram type memory \*/

.data :

{

. = ALIGN(4);

\_sdata = .; /\* create a global symbol at data start \*/

(.data) / .data sections \*/

(.data) /\* .data\* sections \*/

. = ALIGN(4);

\_edata = .; /\* define a global symbol at data end \*/

} >RAM

\_siccmram = LOADADDR(.ccmram);

/\* CCM-RAM section

\*

\* IMPORTANT NOTE!

\* If initialized variables will be placed in this section,

\* the startup code needs to be modified to copy the init-values.

\*/

.ccmram :

{

. = ALIGN(4);

\_sccmram = .; /\* create a global symbol at ccmram start \*/

\*(.ccmram)

(.ccmram)

. = ALIGN(4);

\_eccmram = .; /\* create a global symbol at ccmram end \*/

} >CCMRAM AT> RAM

/\* Uninitialized data section into "RAM" Ram type memory \*/

. = ALIGN(4);

.bss :

{

/\* This is used by the startup in order to initialize the .bss section \*/

\_sbss = .; /\* define a global symbol at bss start \*/

bss\_start = \_sbss;

\*(.bss)

(.bss)

\*(COMMON)

. = ALIGN(4);

\_ebss = .; /\* define a global symbol at bss end \*/

bss\_end = \_ebss;

} >RAM

**STM32RAmid :**

/\* User\_heap\_stack section, used to check that there is enough "RAM" Ram type memory left \*/

.\_user\_heap\_stack :

{

. = ALIGN(8);

PROVIDE ( end = . );

PROVIDE ( \_end = . );

. = . + \_Min\_Heap\_Size;

. = . + \_Min\_Stack\_Size;

. = ALIGN(8);

} >RAM

/\* Remove information from the compiler libraries \*/

/DISCARD/ :

{

libc.a ( \* )

libm.a ( \* )

libgcc.a ( \* )

}

.ARM.attributes 0 : { \*(.ARM.attributes) }

}