```
/*
******************************************************************************
**
**  File        : LinkerScript.ld
**
**  Author      : STM32CubeIDE
**
**  Abstract    : Linker script for STM32H7 series
**                1024Kbytes FLASH and 560Kbytes 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) 2024 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_D1) + LENGTH(RAM_D1);    /* end of RAM */
/* Generate a link error if heap and stack don't fit into RAM */
_Min_Heap_Size = 0x200;      /* required amount of heap  */
_Min_Stack_Size = 0x400; /* required amount of stack */

/* Specify the memory areas */
MEMORY
{
  ITCMRAM (xrw)    : ORIGIN = 0x00000000,   LENGTH = 64K
  DTCMRAM (xrw)    : ORIGIN = 0x20000000,   LENGTH = 128K
  FLASH    (rx)    : ORIGIN = 0x08000000,   LENGTH = 1024K
  RAM_D1  (xrw)    : ORIGIN = 0x24000000,   LENGTH = 320K
  RAM_D2  (xrw)    : ORIGIN = 0x30000000,   LENGTH = 32K
  RAM_D3  (xrw)    : ORIGIN = 0x38000000,   LENGTH = 16K
}

/* Define output sections */
SECTIONS
{
  /* The startup code goes first into FLASH */
  .isr_vector :
  {
```

```
60     . = ALIGN(4);
61     KEEP(*(.isr_vector)) /* Startup code */
62     . = ALIGN(4);
63   } >FLASH
64
65   /* The program code and other data goes into FLASH */
66   .text :
67   {
68     . = ALIGN(4);
69     *(.text)           /* .text sections (code) */
70     *(.text*)          /* .text* sections (code) */
71     *(.glue_7)         /* glue arm to thumb code */
72     *(.glue_7t)        /* glue thumb to arm code */
73     *(.eh_frame)
74
75     KEEP (*(.init))
76     KEEP (*(.fini))
77
78     . = ALIGN(4);
79     _etext = .;        /* define a global symbols at end of code */
80   } >FLASH
81
82   /* Constant data goes into FLASH */
83   .rodata :
84   {
85     . = ALIGN(4);
86     *(.rodata)         /* .rodata sections (constants, strings, etc.) */
87     *(.rodata*)        /* .rodata* sections (constants, strings, etc.) */
88     . = ALIGN(4);
89   } >FLASH
90
91   .ARM.extab (READONLY) : /* The READONLY keyword is only supported in GCC11 and
     later, remove it if using GCC10 or earlier. */
92   {
93     *(.ARM.extab* .gnu.linkonce.armextab.*)
94   } >FLASH
95   .ARM (READONLY) : /* The READONLY keyword is only supported in GCC11 and later,
     remove it if using GCC10 or earlier. */
96   {
97     __exidx_start = .;
98     *(.ARM.exidx*)
99     __exidx_end = .;
100  } >FLASH
101
102  .preinit_array (READONLY) : /* The READONLY keyword is only supported in GCC11 and
     later, remove it if using GCC10 or earlier. */
103  {
104    PROVIDE_HIDDEN (__preinit_array_start = .);
105    KEEP (*(.preinit_array*))
106    PROVIDE_HIDDEN (__preinit_array_end = .);
107  } >FLASH
108
109  .init_array (READONLY) : /* The READONLY keyword is only supported in GCC11 and
     later, remove it if using GCC10 or earlier. */
110  {
111    PROVIDE_HIDDEN (__init_array_start = .);
112    KEEP (*(SORT(.init_array.*)))
113    KEEP (*(.init_array*))
114    PROVIDE_HIDDEN (__init_array_end = .);
```

```
115   } >FLASH
116
117   .fini_array (READONLY) : /* The READONLY keyword is only supported in GCC11 and
    later, remove it if using GCC10 or earlier. */
118   {
119     PROVIDE_HIDDEN (__fini_array_start = .);
120     KEEP (*(SORT(.fini_array.*)))
121     KEEP (*(.fini_array*))
122     PROVIDE_HIDDEN (__fini_array_end = .);
123   } >FLASH
124
125   /* used by the startup to initialize data */
126   _sidata = LOADADDR(.data);
127
128   /* Initialized data sections goes into RAM, load LMA copy after code */
129   .data :
130   {
131     . = ALIGN(4);
132     _sdata = .;        /* create a global symbol at data start */
133     *(.data)           /* .data sections */
134     *(.data*)          /* .data* sections */
135     *(.RamFunc)        /* .RamFunc sections */
136     *(.RamFunc*)       /* .RamFunc* sections */
137
138     . = ALIGN(4);
139     _edata = .;        /* define a global symbol at data end */
140   } >RAM_D1 AT> FLASH
141
142   /* Uninitialized data section */
143   . = ALIGN(4);
144   .bss :
145   {
146     /* This is used by the startup in order to initialize the .bss section */
147     _sbss = .;         /* define a global symbol at bss start */
148     __bss_start__ = _sbss;
149     *(.bss)
150     *(.bss*)
151     *(COMMON)
152
153     . = ALIGN(4);
154     _ebss = .;         /* define a global symbol at bss end */
155     __bss_end__ = _ebss;
156   } >RAM_D1
157
158   /* User_heap_stack section, used to check that there is enough RAM left */
159   ._user_heap_stack :
160   {
161     . = ALIGN(8);
162     PROVIDE ( end = . );
163     PROVIDE ( _end = . );
164     . = . + _Min_Heap_Size;
165     . = . + _Min_Stack_Size;
166     . = ALIGN(8);
167   } >RAM_D1
168
169   /* Remove information from the standard libraries */
170   /DISCARD/ :
171   {
172     libc.a ( * )
```

```
173     libm.a ( * )
174     libgcc.a ( * )
175   }
176
177   .ARM.attributes 0 : { *(.ARM.attributes) }
178 }
179
```