/\* This is the stack that is used by code running within main()

\* In case of NORTOS,

\* - This means all the code outside of ISR uses this stack

\* In case of FreeRTOS

\* - This means all the code until vTaskStartScheduler() is called in main()

\* uses this stack.

\* - After vTaskStartScheduler() each task created in FreeRTOS has its own stack

\*/

--stack\_size=32768//0x00400

/\*--stack\_size=0x400\*/

/\* This is the heap size for malloc() API in NORTOS and FreeRTOS

\* This is also the heap used by pvPortMalloc in FreeRTOS

\*/

/\* --heap\_size=32768 \*/

--heap\_size=98304//0x20000

-e\_vectors /\* This is the entry of the application, \_vector MUST be plabed starting address 0x0 \*/

/\* This is the size of stack when R5 is in IRQ mode

\* In NORTOS,

\* - Here interrupt nesting is disabled as of now

\* - This is the stack used by ISRs registered as type IRQ

\* In FreeRTOS,

\* - Here interrupt nesting is enabled

\* - This is stack that is used initally when a IRQ is received

\* - But then the mode is switched to SVC mode and SVC stack is used for all user ISR callbacks

\* - Hence in FreeRTOS, IRQ stack size is less and SVC stack size is more

\*/

\_\_IRQ\_STACK\_SIZE = 256;

/\* This is the size of stack when R5 is in IRQ mode

\* - In both NORTOS and FreeRTOS nesting is disabled for FIQ

\*/

\_\_FIQ\_STACK\_SIZE = 256;

\_\_SVC\_STACK\_SIZE = 4096; /\* This is the size of stack when R5 is in SVC mode \*/

\_\_ABORT\_STACK\_SIZE = 256; /\* This is the size of stack when R5 is in ABORT mode \*/

\_\_UNDEFINED\_STACK\_SIZE = 256; /\* This is the size of stack when R5 is in UNDEF mode \*/

SECTIONS

{

 /\* This has the R5F entry point and vector table, this MUST be at 0x0 \*/

 .vectors:{} palign(8) > R5F\_VECS

 /\* This has the R5F boot code until MPU is enabled, this MUST be at a address < 0x80000000

 \* i.e this cannot be placed in DDR

 \*/

 /\* Sections needed for C++ projects \*/

 GROUP {

 .ARM.exidx: {} palign(8) /\* Needed for C++ exception handling \*/

 .init\_array: {} palign(8) /\* Contains function pointers called before main \*/

 .fini\_array: {} palign(8) /\* Contains function pointers called after main \*/

 } > DDR\_1

 GROUP {

 .text.hwi: palign(8)

 .text.cache: palign(8)

 .text.mpu: palign(8)

 .text.boot: palign(8)

 .text:abort: palign(8)

 } > R5F\_TCMA

 /\* This is rest of code. This can be placed in DDR if DDR is available and needed \*/

 GROUP {

 .text: {} palign(8) /\* This is where code resides \*/

 .rodata: {} palign(8) /\* This is where const's go \*/

 } > DDR\_1

 GROUP {

 /\* This is the resource table used by linux to know where the IPC "VRINGs" are located \*/

 .resource\_table: {} palign(4096)

 } > DDR\_0

 /\* This is rest of initialized data. This can be placed in DDR if DDR is available and needed \*/

 GROUP {

 .data: {} palign(8) /\* This is where initialized globals and static go \*/

 } > DDR\_1 //MSRAM

 /\* This is rest of uninitialized data. This can be placed in DDR if DDR is available and needed \*/

 GROUP {

 .stack: {} palign(8) FILL(0x00000000) /\* This is where the main() stack goes \*/

 } >DDR\_1//R5F\_TCMA

 GROUP {

 .threadstack: {} palign(8) FILL(0x00000000) /\* This is where the EIP thread stacks go \*/

 } > DDR\_1 //R5F\_TCMB0 DDR1

 GROUP {

 .bss: {} palign(8) FILL(0x00000000) /\* This is where uninitialized globals go \*/

 RUN\_START(\_\_BSS\_START)

 RUN\_END(\_\_BSS\_END)

 .sysmem: {} palign(8) /\* This is where the malloc heap goes \*/

 RUN\_START(\_\_HEAP\_START)

 RUN\_END(\_\_HEAP\_END)

 .fbtlthreadstack: {} palign(8) FILL(0x00000000) /\* This is where the FBTL thread stacks go \*/

 } > DDR\_1 //MSRAM

 /\* This is where the stacks for different R5F modes go \*/

 GROUP {

 .irqstack: {. = . + \_\_IRQ\_STACK\_SIZE;} align(8)

 RUN\_START(\_\_IRQ\_STACK\_START)

 RUN\_END(\_\_IRQ\_STACK\_END)

 .fiqstack: {. = . + \_\_FIQ\_STACK\_SIZE;} align(8)

 RUN\_START(\_\_FIQ\_STACK\_START)

 RUN\_END(\_\_FIQ\_STACK\_END)

 .svcstack: {. = . + \_\_SVC\_STACK\_SIZE;} align(8)

 RUN\_START(\_\_SVC\_STACK\_START)

 RUN\_END(\_\_SVC\_STACK\_END)

 .abortstack: {. = . + \_\_ABORT\_STACK\_SIZE;} align(8)

 RUN\_START(\_\_ABORT\_STACK\_START)

 RUN\_END(\_\_ABORT\_STACK\_END)

 .undefinedstack: {. = . + \_\_UNDEFINED\_STACK\_SIZE;} align(8)

 RUN\_START(\_\_UNDEFINED\_STACK\_START)

 RUN\_END(\_\_UNDEFINED\_STACK\_END)

 } > DDR\_1

 /\* Packet buffer memory used by ICCS \*/

 .bss.icss\_emac\_pktbuf\_mem (NOLOAD): {} palign(8) FILL(0x00000000) > ICSS\_PKT\_BUF\_MEM

 /\* General purpose user shared memory, used in some examples \*/

 .bss.user\_shared\_mem (NOLOAD) : {} > USER\_SHM\_MEM

 /\* this is used when Debug log's to shared memory are enabled, else this is not used \*/

 .bss.log\_shared\_mem (NOLOAD) : {} > LOG\_SHM\_MEM

 /\* this is used only when IPC RPMessage is enabled, else this is not used \*/

 .bss.ipc\_vring\_mem (NOLOAD) : {} > RTOS\_NORTOS\_IPC\_SHM\_MEM

}

/\*

NOTE: Below memory is reserved for DMSC usage

- During Boot till security handoff is complete

 0x701E0000 - 0x701FFFFF (128KB)

- After "Security Handoff" is complete (i.e at run time)

 0x701FC000 - 0x701FFFFF (16KB)

Security handoff is complete when this message is sent to the DMSC,

 TISCI\_MSG\_SEC\_HANDOVER

This should be sent once all cores are loaded and all application

specific firewall calls are setup.

\*/

MEMORY

{

 R5F\_VECS : ORIGIN = 0x00000000 , LENGTH = 0x00000040

 R5F\_TCMA : ORIGIN = 0x00000040 , LENGTH = 0x00007FC0

 R5F\_TCMB0 : ORIGIN = 0x41010000 , LENGTH = 0x00008000

 /\* when using multi-core application's i.e more than one R5F/M4F active, make sure

 \* this memory does not overlap with other R5F's

 \*/

 MSRAM : ORIGIN = 0x70080000 , LENGTH = 0x40000

 /\* This section can be used to put XIP section of the application in flash, make sure this does not overlap with

 \* other CPUs. Also make sure to add a MPU entry for this section and mark it as cached and code executable

 \*/

 FLASH : ORIGIN = 0x60100000 , LENGTH = 0x80000

 /\* shared memories that is used between ICCS and this core. MARK as non-cache or cache+sharable \*/

 ICSS\_PKT\_BUF\_MEM : ORIGIN = 0xA3100000, LENGTH = 0x00010000

 //LINUX\_IPC\_SHM\_MEM : ORIGIN = 0xA0000000 , LENGTH = 0x100000

 DDR\_0 : ORIGIN = 0xA0100000 , LENGTH = 0x100000

 DDR\_1 : ORIGIN = 0xA0200000 , LENGTH = 0xE00000

 //1MB

/\* shared memory segments \*/

 /\* On R5F,

 \* - make sure there is a MPU entry which maps below regions as non-cache

 \*/

 USER\_SHM\_MEM : ORIGIN = 0xA5800000, LENGTH = 0x80

 LOG\_SHM\_MEM : ORIGIN = 0xA5800000 + 0x80, LENGTH = 0x00004000 - 0x80

 RTOS\_NORTOS\_IPC\_SHM\_MEM : ORIGIN = 0xA5000000, LENGTH = 0x00800000

}