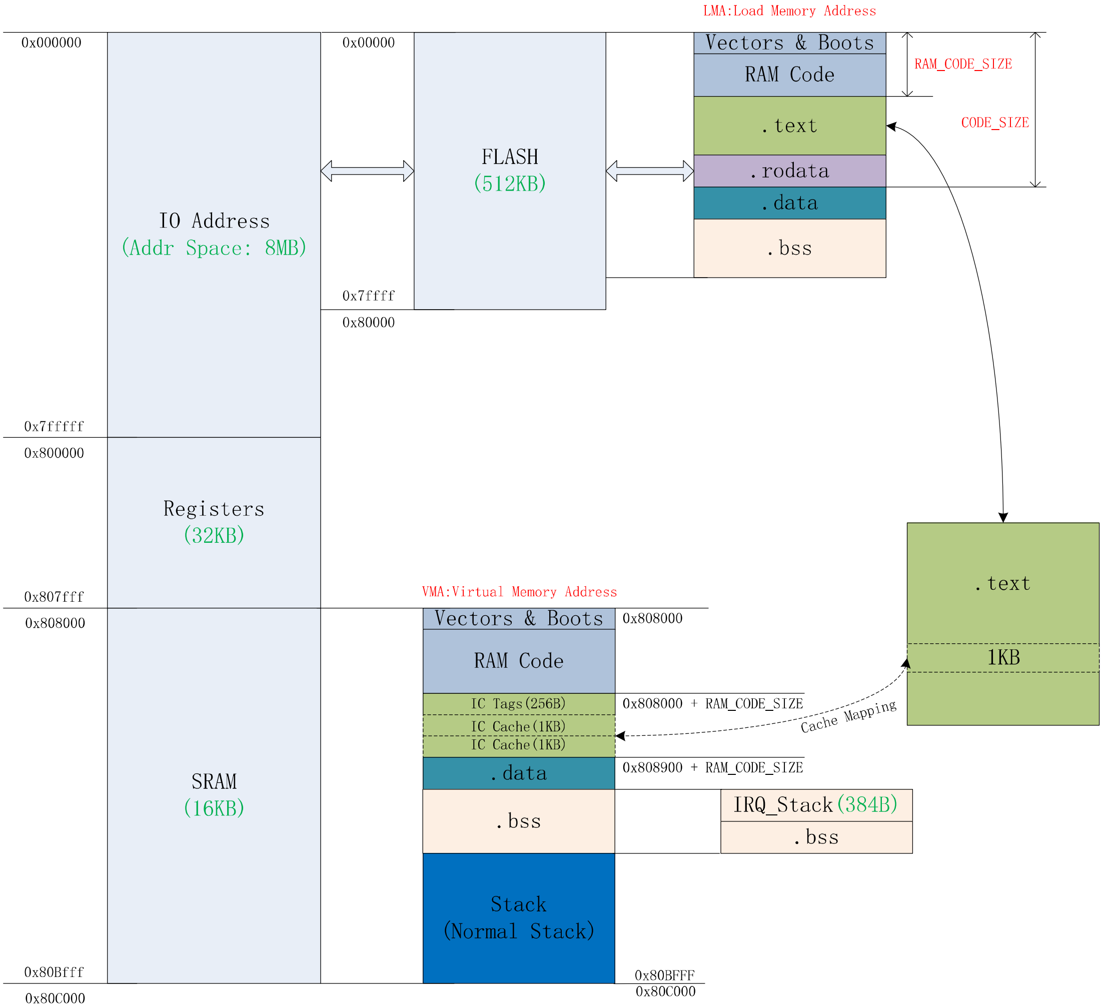
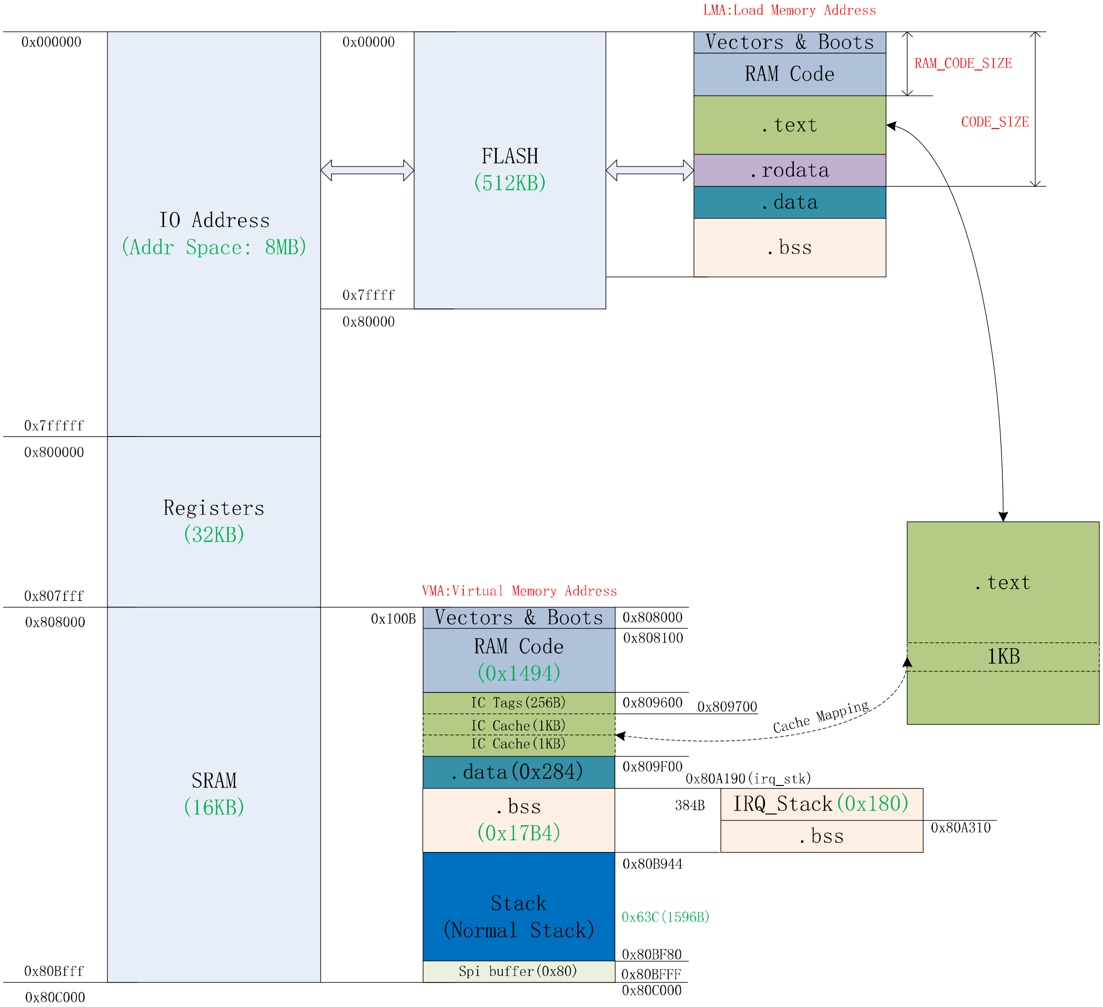
Boot.link

```
/*****************************************************************************************
 * to tell the linker the program begin from __start label in cstartup.s, thus do not   *
 * treat it as a unused symbol                                                           *
 *****************************************************************************************/

ENTRY(__start)

SECTIONS
{
   . = 0x0;
       .vectors :
       {
       *(.vectors)
       *(.vectors.*) /* MUST as follows, when compile with -ffunction-sections -fdata-sections,
                     session name may changed */
       }
       .ram_code :
       {
       *(.ram_code)
       *(.ram_code.*)
       }
    PROVIDE(_ramcode_size_ = . );
    PROVIDE(_ramcode_size_div_16_ = (. + 15 ) / 16);
    PROVIDE(_ramcode_size_div_256_ = (. + 255) / 256);
    PROVIDE(_ramcode_size_div_16_align_256_ = ( (. + 255) / 256) * 16);
       .text :
       {
       *(.text)
       *(.text.*)
       }
       .rodata :
       {
       *(.rodata)
       *(.rodata.*)
       }

   . = (((. + 3) / 4)*4);
       PROVIDE(_dstored_ = .);
       PROVIDE(_code_size_ = .);

    . = 0x808900 + _ramcode_size_div_256_ * 0x100;  /* 0x100 aligned, must greater than or
equal to:0x808000 + ram_code_size +  irq_vector(0x100) + IC_tag(0x100) + IC_cache(0x800) ==
0x808a00 + ram_code_size */
       .data :
        AT ( _dstored_ )  /* .data reprents VMA, _dstored_ represents LMA */
        {
   . = (((. + 3) / 4)*4);
        PROVIDE(_start_data_ = . );
        *(.data);
        *(.data.*);
   . = (((. + 3) / 4)*4);
        PROVIDE(_end_data_ = . );
        }

       .bss :
       {
    . = (((. + 3) / 4)*4);
    PROVIDE(_start_bss_ = .);
       *(.sbss)
       *(.sbss.*)
       *(.bss)
       *(.bss.*)
       }
   PROVIDE(_end_bss_ = .);
   PROVIDE(_bin_size_ = _code_size_ + _end_data_ - _start_data_);
   PROVIDE(_ictag_start_ = 0x808000 + (_ramcode_size_div_256_) * 0x100);
   PROVIDE(_ictag_end_ = 0x808000 + (_ramcode_size_div_256_ + 1) * 0x100);
}
```



```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .vectors      00000100  00000000  00000000  00008000  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .ram_code     00001494  00000100  00000100  00008100  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .text         00005790  000015a0  000015a0  000095a0  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  3 .rodata       00000bf4  00006d30  00006d30  0000ed30  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .data         00000284  00809f00  00007924  00011f00  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  5 .bss          000017b4  0080a190  00007bb4  00012184  2**4
                  ALLOC
  6 .TC32.attributes 00000010  00000000  00000000  00012184  2**0
                  CONTENTS, READONLY
  7 .comment      0000001a  00000000  00000000  00012194  2**0
                  CONTENTS, READONLY
SYMBOL TABLE:
00000000 l    d  .vectors  00000000 .vectors
00000100 l    d  .ram_code 00000000 .ram_code
000015a0 l    d  .text 00000000 .text
00006d30 l    d  .rodata   00000000 .rodata
00809f00 l    d  .data 00000000 .data
0080a190 l    d  .bss  00000000 .bss
……
000000b8 l       .vectors  00000000 FLL_D
```

```
00000026 l      .vectors   00000000 FLL_STK
00000030 l      .vectors   00000000 FLL_STK_END
……
000000ac l      .vectors   00000000 DATA_I
0000006e l      .vectors   00000000 COPY_DATA
0000007c l      .vectors   00000000 COPY_DATA_END
00000080 l      .vectors   00000000 END
0080a190 l     O .bss  00000180 irq_stk
000000f8 l      .vectors   00000000 ASMEND
……
0080a538 l     O .bss  00000004 keyScanTick.5315
……
00000374 l     F .ram_code 00000040 flash_send_addr
000003b4 l     F .ram_code 0000002c flash_send_cmd
……
0080a8d0 l     O .bss  00000004 connection_offset.5091
……
0080a310 l     O .bss  00000004 pd.2723
0080a314 l     O .bss  00000002 buffer_mic_rptr.2758
…
```



```
/***************************************************************************************
 * cstart_up.S                                                                         *
 ***************************************************************************************/
    .code    16
.include "version.in"
                      @ Mode, corresponds to bits 0-5 in CPSR
    .equ MODE_BITS,       0x1F@ Bit mask for mode bits in CPSR
    .equ IRQ_MODE,        0x12@ Interrupt Request mode
    .equ SVC_MODE,        0x13@ Supervisor mode

    .equ IRQ_STK_SIZE,    0x180
    .equ __LOAD_RAM,  __LOAD_RAM_SIZE__

@*******************************************************************
@      TC32 EXCEPTION VECTORS
@*******************************************************************
    .section .vectors,"ax"
    .global  __reset
    .global  __irq
    .global  __start
    .global  __LOAD_RAM

__start:                  @ MUST, referenced by boot.link

    .extern irq_handler

    .extern  _ramcode_size_div_16_
    .extern  _ramcode_size_div_256_
    .extern  _ramcode_size_div_16_align_256_
    .extern  _ictag_start_
    .extern  _ictag_end_

    .org 0x0
    tj   __reset
    .word    (BUILD_VERSION)   @ version
    .org 0x8
    .word    (0x544c4e4b)      @ ‘T’, ‘L’, ‘N’, ‘K’, “TLNK” represents Telink
    .word    (0x00880000 + _ramcode_size_div_16_align_256_)

    .org 0x10
    tj        __irq
    .org 0x18
    .word    (_bin_size_)      @ Bin file size
@***************************************************************************************
@                             LOW-LEVEL INITIALIZATION
@***************************************************************************************
    .extern  main

    .org 0x20
__reset:
@   tloadr   r0, DAT0 + 36
@   tmov     r1, #1024         @ set sws to GPIO
@   tstorer  r1, [r0, #0]

@   tloadr   r0, DAT0 + 40     @**** enable watchdog at the very first time
@   tloadr   r1, DAT0 + 44
@   tstorer  r1, [r0, #0]

    tloadr   r0, FLL_D
    tloadr   r1, FLL_D+4
    tloadr   r2, FLL_D+8

FLL_STK:   @ Init .data to zero in VMA, rather than LMA, prepare to copy LMA to VMA
    tcmp r1,  r2
    tjge FLL_STK_END
    tstorer r0, [r1, #0]
    tadd    r1, #4
    tj        FLL_STK
FLL_STK_END:

    tloadr   r0, DAT0          @ IRQ Mode
    tmcsr    r0                 @ Change mode to IRQ
    tloadr   r0, DAT0 + 8       @ IRQ Stack, refer to irq_stk in .bss with lcomm attribution
    tmov r13, r0                @ r13 works as SP (Stack Pointer), set Stack for IRQ Mode

    tloadr   r0, DAT0 + 4       @ Normal Mode,
    tmcsr    r0                 @ Change mode to Normal
    tloadr   r0, DAT0 + 12      @ Normal Stack
    tmov r13, r0                @ r13 works as SP (Stack Pointer), set Stack for NOrmal Mode

    tmov r0, #0                 @ Prepare for init .bss section
    tloadr   r1, DAT0 + 16
    tloadr   r2, DAT0 + 20

ZERO:                           @ Init .bss to zero
    tcmp r1, r2
    tjge ZERO_END
    tstorer  r0, [r1, #0]
    tadd    r1, #4
    tj       ZERO
ZERO_END:

    tloadr   r1, DAT0 + 28      @ Prepare to init IC Tags for IC Cache
    tloadr   r2, DAT0 + 32

ZERO_TAG:                       @ Init IC tags to zero
    tcmp r1, r2
    tjge ZERO_TAG_END
    tstorer  r0, [r1, #0]
    tadd    r1, #4
    tj       ZERO_TAG
ZERO_TAG_END:

SETIC:
    tloadr   r1, DAT0 + 24      @ 0x60C is used for IC tags start address register
    tloadr   r0, DAT0 + 36      @ IC tags start address follows RAM_CODE_END
    tstorerb r0, [r1, #0]       @ Set IC tags start address
    tadd     r0, #1             @ 0x60D is used for IC Cache start address register
    tstorerb r0, [r1, #1]       @ IC tags with align of 256B, so +1 means +256
                                @ So, IC Cache start address is IC tags + 256
    tloadr   r1, DATA_I         @ Prepare to copy .data section from LMA to VMA
    tloadr   r2, DATA_I+4
    tloadr   r3, DATA_I+8
COPY_DATA:                      @ Copy .data section from LMA to VMA
    tcmp     r2, r3
    tjge     COPY_DATA_END
    tloadr   r0, [r1, #0]
    tstorer  r0, [r2, #0]
    tadd     r1, #4
    tadd     r2, #4
    tj       COPY_DATA
COPY_DATA_END:

    tjl  main                   @ Call main
END: tj   END                   @ If main returns, we just while loop here.

    .balign  4
DAT0:
    .word    0x12              @IRQ    @0
    .word    0x13              @SVC    @4
    .word    (irq_stk + IRQ_STK_SIZE)      @IRQ STACK
    .word    (0x80c000 - 128)          @12  stack end :spi buffer 64*2
    .word    (_start_bss_)             @16
    .word    (_end_bss_)               @20
    .word    (0x80060c)                @24
    .word    _ictag_start_             @28      @ IC tag start
    .word    _ictag_end_               @32      @ IC tag end
    .word    _ramcode_size_div_256_    @36
@   .word    (0x808000 + __LOAD_RAM * 0x100)               @28        @ IC tag start
@   .word    (0x808000 + (__LOAD_RAM + 1) * 0x100)          @32       @ IC tag end
@   .word    (0x80000e)                        @36
@   .word    (0x80058c)                 @36       gpio
@    .word    (0x800620)               @40       watchdog
@    .word    (0x802c01)               @44       watchdog
DATA_I:
     .word    _dstored_
     .word    _start_data_
     .word    _end_data_

FLL_D:
     .word    0x00000000 @0xffffffff
     .word    (_start_data_)
     .word    (0x80c000)

     .align 4
__irq:
     tpush    {r14}
     tpush    {r0-r7}
     tmrss    r0

     tmov     r1, r8
     tmov     r2, r9
     tmov     r3, r10
     tmov     r4, r11
     tmov     r5, r12
     tpush    {r0-r5}

     tjl      irq_handler               @Handler for IRQ entrance

     tpop     {r0-r5}
     tmov     r8, r1
     tmov     r9, r2
     tmov     r10,r3
     tmov     r11,r4
     tmov     r12,r5

     tmssr    r0
     tpop     {r0-r7}
     treti    {r15}

ASMEND:

     .section .bss
     .align 4
     .lcomm irq_stk, IRQ_STK_SIZE       @IRQ Stack, Local common location
     .end

#endif
```