De

内核裁剪

RealTouch 评估板 RT-Thread 入门文档

版本号：1.0.0
日期：2012/9/5

修订记录

日期	作者	修订历史
2012/9/5	bloom5	创建文档

实验目的

- 因为 RT-Thread 拥有高度可裁剪性，我们尝试一下可以它到底可以裁到多小

硬件说明

本实验使用 RT-Thread 官方的 Realtouch 开发板作为实验平台。涉及到的硬件主要为

- 无，本实验只涉及编译，当然在板上验证正确性也是可以的

实验原理及程序结构

关于内核裁剪的内容，也可以参考 wiki：http://www.rt-thread.org/dokuwiki/doku.php?id=rt-thread%E8%A3%81%E5%89%AA%E7%A4%BA%E4%BE%8B

实验设计

本实验的主要设计目的是通过修改 rt_config.h 文件，来尝试可以达到的最小裁剪尺寸。请读者注意，本实验本身不具有实际的工程参考价值，只是帮助读者快速了解相关 API 的用法。

源程序说明

本实验对应 xxxxxx

系统依赖

因为是为了裁剪，所以在 rtconfig.h 中不需要特意开启任何项目

主程序说明以及相关编译结果

首先我们看看默认的 rtconfig.h 中内容。

```
/* RT-Thread config file */
#ifndef __RTTHREAD_CFG_H__
#define __RTTHREAD_CFG_H__

//#define RT_USING_NEWLIB
```

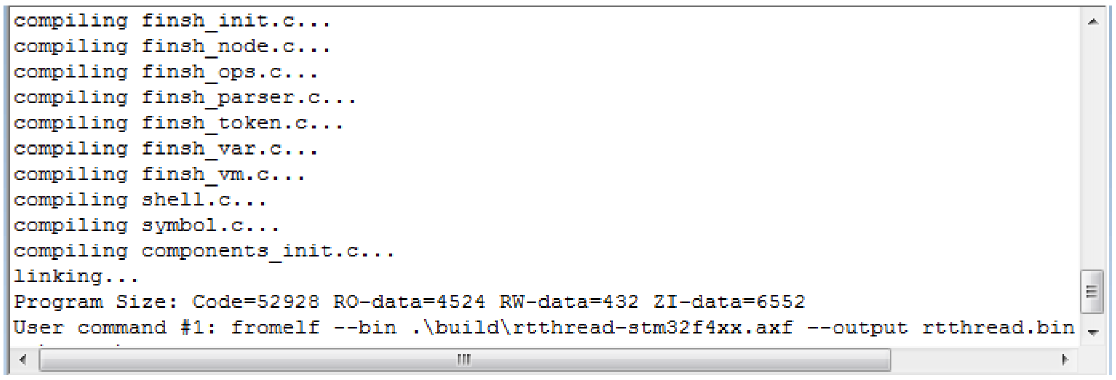
```
/* RT_NAME_MAX*/
#define RT_NAME_MAX          8

/* RT_ALIGN_SIZE*/
#define RT_ALIGN_SIZE     8

/* PRIORITY_MAX */
#define RT_THREAD_PRIORITY_MAX   32

/* Tick per Second */
#define RT_TICK_PER_SECOND   100

/* SECTION: RT_DEBUG */
/* Thread Debug */
#define RT_DEBUG

#define RT_USING_OVERFLOW_CHECK

/* Using Hook */
#define RT_USING_HOOK

#define IDLE_THREAD_STACK_SIZE    1024

/* Using Software Timer */
/* #define RT_USING_TIMER_SOFT */
#define RT_TIMER_THREAD_PRIO            4
#define RT_TIMER_THREAD_STACK_SIZE      512
#define RT_TIMER_TICK_PER_SECOND  10

/* SECTION: IPC */
/* Using Semaphore*/
#define RT_USING_SEMAPHORE

/* Using Mutex */
#define RT_USING_MUTEX

/* Using Event */
#define RT_USING_EVENT

/* Using MailBox */
#define RT_USING_MAILBOX

/* Using Message Queue */
#define RT_USING_MESSAGEQUEUE
```

```
/* SECTION: Memory Management */
/* Using Memory Pool Management*/
#define RT_USING_MEMPOOL

/* Using Dynamic Heap Management */
#define RT_USING_HEAP

/* Using Small MM */
#define RT_USING_SMALL_MEM

/* SECTION: Device System */
/* Using Device System */
#define RT_USING_DEVICE
#define RT_USING_SERIAL

/* SECTION: Console options */
#define RT_USING_CONSOLE
/* the buffer size of console*/
#define RT_CONSOLEBUF_SIZE128
/* console name */
#define RT_CONSOLE_DEVICE_NAME    "uart3"

#define RT_USING_COMPONENTS_INIT

/* SECTION: finsh, a C-Express shell */
#define RT_USING_FINSH
/* Using symbol table */
#define FINSH_USING_SYMTAB
#define FINSH_USING_DESCRIPTION

/* SECTION: device filesystem */
/* #define RT_USING_DFS */
//#define RT_USING_DFS_ELMFAT
#define RT_DFS_ELM_WORD_ACCESS
/* Reentrancy (thread safe) of the FatFs module.  */
#define RT_DFS_ELM_REENTRANT
/* Number of volumes (logical drives) to be used. */
#define RT_DFS_ELM_DRIVES               2
/* #define RT_DFS_ELM_USE_LFN                    1 */
#define RT_DFS_ELM_MAX_LFN              255
/* Maximum sector size to be handled. */
#define RT_DFS_ELM_MAX_SECTOR_SIZE  512
```

```
/* the max number of mounted filesystem */
#define DFS_FILESYSTEMS_MAX                     2
/* the max number of opened files               */
#define DFS_FD_MAX                              4

/* SECTION: lwip, a lighwight TCP/IP protocol stack */
//#define RT_USING_LWIP
/* LwIP uses RT-Thread Memory Management */
#define RT_LWIP_USING_RT_MEM
/* Enable ICMP protocol*/
#define RT_LWIP_ICMP
/* Enable UDP protocol*/
#define RT_LWIP_UDP
/* Enable TCP protocol*/
#define RT_LWIP_TCP
/* Enable DNS */
#define RT_LWIP_DNS

/* the number of simulatenously active TCP connections*/
#define RT_LWIP_TCP_PCB_NUM      5

/* ip address of target*/
#define RT_LWIP_IPADDR0   192
#define RT_LWIP_IPADDR1   168
#define RT_LWIP_IPADDR2   1
#define RT_LWIP_IPADDR3   30

/* gateway address of target*/
#define RT_LWIP_GWADDR0   192
#define RT_LWIP_GWADDR1   168
#define RT_LWIP_GWADDR2   1
#define RT_LWIP_GWADDR3   1

/* mask address of target*/
#define RT_LWIP_MSKADDR0  255
#define RT_LWIP_MSKADDR1  255
#define RT_LWIP_MSKADDR2  255
#define RT_LWIP_MSKADDR3  0

/* tcp thread options */
#define RT_LWIP_TCPTHREAD_PRIORITY              12
#define RT_LWIP_TCPTHREAD_MBOX_SIZE             4
#define RT_LWIP_TCPTHREAD_STACKSIZE             1024
```

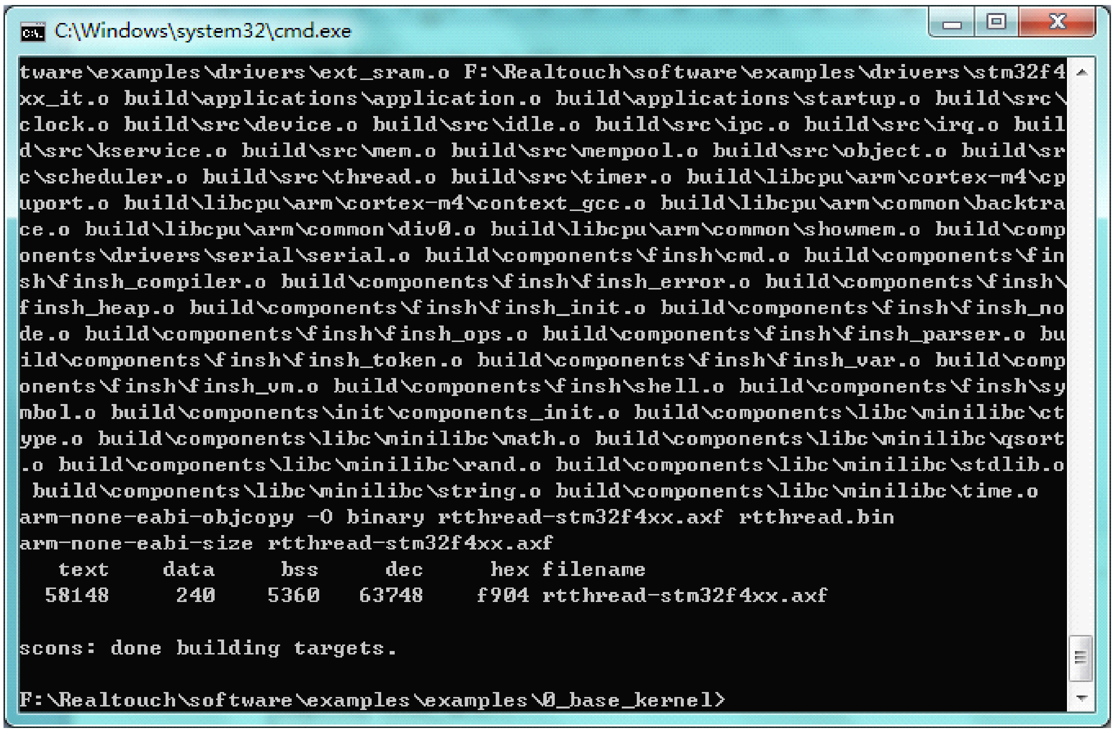
```
/* ethernet if thread options */
#define RT_LWIP_ETHTHREAD_PRIORITY              15
#define RT_LWIP_ETHTHREAD_MBOX_SIZE             4
#define RT_LWIP_ETHTHREAD_STACKSIZE             512

/* TCP sender buffer space */
#define RT_LWIP_TCP_SND_BUF         8192
/* TCP receive window. */
#define RT_LWIP_TCP_WND             8192

#define CHECKSUM_CHECK_TCP              0
#define CHECKSUM_CHECK_IP               0
#define CHECKSUM_CHECK_UDP              0

#define CHECKSUM_GEN_TCP                0
#define CHECKSUM_GEN_IP                 0
#define CHECKSUM_GEN_UDP                0

#endif
```

这是 keil 编译的结果：



```
compiling finsh_init.c...
compiling finsh_node.c...
compiling finsh_ops.c...
compiling finsh_parser.c...
compiling finsh_token.c...
compiling finsh_var.c...
compiling finsh_vm.c...
compiling shell.c...
compiling symbol.c...
compiling components_init.c...
linking...
Program Size: Code=52928 RO-data=4524 RW-data=432 ZI-data=6552
User command #1: fromelf --bin .\build\rtthread-stm32f4xx.axf --output rtthread.bin
```

这是 gcc 编译结果：



```
tware\examples\drivers\ext_sram.o F:\Realtouch\software\examples\drivers\stm32f4
xx_it.o build\applications\application.o build\applications\startup.o build\src\
clock.o build\src\device.o build\src\idle.o build\src\ipc.o build\src\irq.o buil
d\src\kservice.o build\src\mem.o build\src\mempool.o build\src\object.o build\sr
c\scheduler.o build\src\thread.o build\src\timer.o build\libcpu\arm\cortex-m4\cp
uport.o build\libcpu\arm\cortex-m4\context_gcc.o build\libcpu\arm\common\backtra
ce.o build\libcpu\arm\common\div0.o build\libcpu\arm\common\showmem.o build\comp
onents\drivers\serial\serial.o build\components\finsh\cmd.o build\components\fin
sh\finsh_compiler.o build\components\finsh\finsh_error.o build\components\finsh\
finsh_heap.o build\components\finsh\finsh_init.o build\components\finsh\finsh_no
de.o build\components\finsh\finsh_ops.o build\components\finsh\finsh_parser.o bu
ild\components\finsh\finsh_token.o build\components\finsh\finsh_var.o build\comp
onents\finsh\finsh_vm.o build\components\finsh\shell.o build\components\finsh\sy
mbol.o build\components\init\components_init.o build\components\libc\minilibc\ct
ype.o build\components\libc\minilibc\math.o build\components\libc\minilibc\qsort
.o build\components\libc\minilibc\rand.o build\components\libc\minilibc\stdlib.o
 build\components\libc\minilibc\string.o build\components\libc\minilibc\time.o
arm-none-eabi-objcopy -O binary rtthread-stm32f4xx.axf rtthread.bin
arm-none-eabi-size rtthread-stm32f4xx.axf
   text    data     bss     dec     hex filename
  58148     240    5360   63748    f904 rtthread-stm32f4xx.axf

scons: done building targets.

F:\Realtouch\software\examples\examples\0_base_kernel>
```

这时候的结果包含有一个初始化线程，接下来我们将 application.c 中这个初始化线程移除，另外的话我们要对 rt_config.h 做一次比较大的手术。参照 wiki 上项目，我们几乎见到 RT_USING_字样的宏就将其注释掉，不多说，上“马”：

```
/* RT-Thread config file */
#ifndef __RTTHREAD_CFG_H__
#define __RTTHREAD_CFG_H__

//#define RT_USING_NEWLIB

/* RT_NAME_MAX*/
#define RT_NAME_MAX   4

/* RT_ALIGN_SIZE*/
#define RT_ALIGN_SIZE  4

/* PRIORITY_MAX */
#define RT_THREAD_PRIORITY_MAX   8

/* Tick per Second */
#define RT_TICK_PER_SECOND       100

/* SECTION: RT_DEBUG */
/* Thread Debug */
//#define RT_DEBUG

//#define RT_USING_OVERFLOW_CHECK
```

```
/* Using Hook */
//#define RT_USING_HOOK

#define IDLE_THREAD_STACK_SIZE    256

/* Using Software Timer */
/* #define RT_USING_TIMER_SOFT */
#define RT_TIMER_THREAD_PRIO     4
#define RT_TIMER_THREAD_STACK_SIZE  512
#define RT_TIMER_TICK_PER_SECOND 10

/* SECTION: IPC */
/* Using Semaphore*/
#define RT_USING_SEMAPHORE

/* Using Mutex */
#define RT_USING_MUTEX

/* Using Event */
#define RT_USING_EVENT

/* Using MailBox */
//#define RT_USING_MAILBOX

/* Using Message Queue */
//#define RT_USING_MESSAGEQUEUE

/* SECTION: Memory Management */
/* Using Memory Pool Management*/
//#define RT_USING_MEMPOOL

/* Using Dynamic Heap Management */
//#define RT_USING_HEAP

/* Using Small MM */
//#define RT_USING_SMALL_MEM

/* SECTION: Device System */
/* Using Device System */
//#define RT_USING_DEVICE
//#define RT_USING_SERIAL

/* SECTION: Console options */
```

```
//#define RT_USING_CONSOLE
/* the buffer size of console*/
#define RT_CONSOLEBUF_SIZE   128
/* console name */
#define RT_CONSOLE_DEVICE_NAME   "uart3"

//#define RT_USING_COMPONENTS_INIT

/* SECTION: finsh, a C-Express shell */
//#define RT_USING_FINSH
/* Using symbol table */
#define FINSH_USING_SYMTAB
#define FINSH_USING_DESCRIPTION

/* SECTION: device filesystem */
/* #define RT_USING_DFS */
//#define RT_USING_DFS_ELMFAT
#define RT_DFS_ELM_WORD_ACCESS
/* Reentrancy (thread safe) of the FatFs module.  */
#define RT_DFS_ELM_REENTRANT
/* Number of volumes (logical drives) to be used. */
#define RT_DFS_ELM_DRIVES        2
/* #define RT_DFS_ELM_USE_LFN       1 */
#define RT_DFS_ELM_MAX_LFN          255
/* Maximum sector size to be handled. */
#define RT_DFS_ELM_MAX_SECTOR_SIZE  512

/* the max number of mounted filesystem */
#define DFS_FILESYSTEMS_MAX         2
/* the max number of opened files      */
#define DFS_FD_MAX               4

/* SECTION: lwip, a lighwight TCP/IP protocol stack */
//#define RT_USING_LWIP
/* LwIP uses RT-Thread Memory Management */
#define RT_LWIP_USING_RT_MEM
/* Enable ICMP protocol*/
#define RT_LWIP_ICMP
/* Enable UDP protocol*/
#define RT_LWIP_UDP
/* Enable TCP protocol*/
#define RT_LWIP_TCP
/* Enable DNS */
#define RT_LWIP_DNS
```

```
 /* the number of simulatenously active TCP
connections*/
 #define RT_LWIP_TCP_PCB_NUM  5

 /* ip address of target*/
 #define RT_LWIP_IPADDR0   192
 #define RT_LWIP_IPADDR1   168
 #define RT_LWIP_IPADDR2   1
 #define RT_LWIP_IPADDR3   30

 /* gateway address of target*/
 #define RT_LWIP_GWADDR0   192
 #define RT_LWIP_GWADDR1   168
 #define RT_LWIP_GWADDR2   1
 #define RT_LWIP_GWADDR3   1

 /* mask address of target*/
 #define RT_LWIP_MSKADDR0  255
 #define RT_LWIP_MSKADDR1  255
 #define RT_LWIP_MSKADDR2  255
 #define RT_LWIP_MSKADDR3  0

 /* tcp thread options */
 #define RT_LWIP_TCPTHREAD_PRIORITY     12
 #define RT_LWIP_TCPTHREAD_MBOX_SIZE    4
 #define RT_LWIP_TCPTHREAD_STACKSIZE    1024

 /* ethernet if thread options */
 #define RT_LWIP_ETHTHREAD_PRIORITY     15
 #define RT_LWIP_ETHTHREAD_MBOX_SIZE    4
 #define RT_LWIP_ETHTHREAD_STACKSIZE    512

 /* TCP sender buffer space */
 #define RT_LWIP_TCP_SND_BUF  8192
 /* TCP receive window. */
 #define RT_LWIP_TCP_WND      8192

 #define CHECKSUM_CHECK_TCP           0
 #define CHECKSUM_CHECK_IP            0
 #define CHECKSUM_CHECK_UDP           0

 #define CHECKSUM_GEN_TCP             0
 #define CHECKSUM_GEN_IP              0
```

```
#define CHECKSUM_GEN_UDP                0

#endif
```

改完，如果你就急着编译的话，会很悲剧的出一系列错，因为我们在组织工程的时候仍旧加上了一些非必须的文件，另外在 startup.c，platform.c,board.c 中也有一些地方需要改动。

首先改动的是例程文件夹下 drivers 下的 SConscript，将其中的 usart.c 彻底去掉。

startup.c 中的 rt_show_version()可以去掉，另外关于 heap 可以加上条件编译 ：

```
#ifdef RT_USING_HEAP
    rt_system_heap_init((void*)STM32_SRAM_BEGIN,
(void*)STM32_SRAM_END);
#endif
```

此外，

```
#if RT_USING_DEVICE
   /* init all device */
   rt_device_init_all();
#endif
```

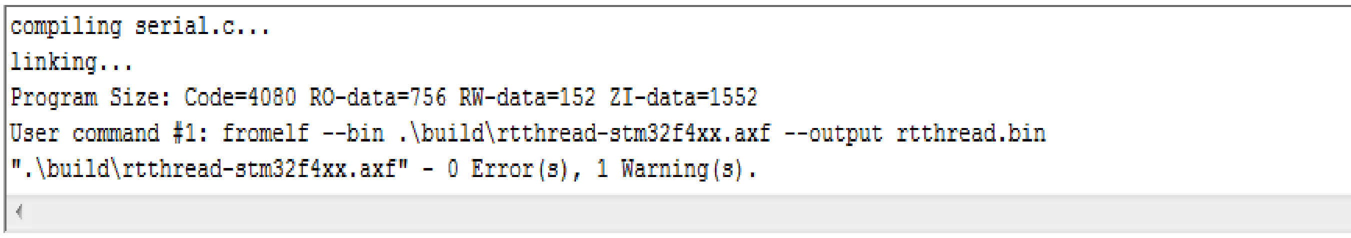
在 platform.c 中，

```
#ifdef RT_USING_DEVICE
    rt_device_init_all();
#endif
```

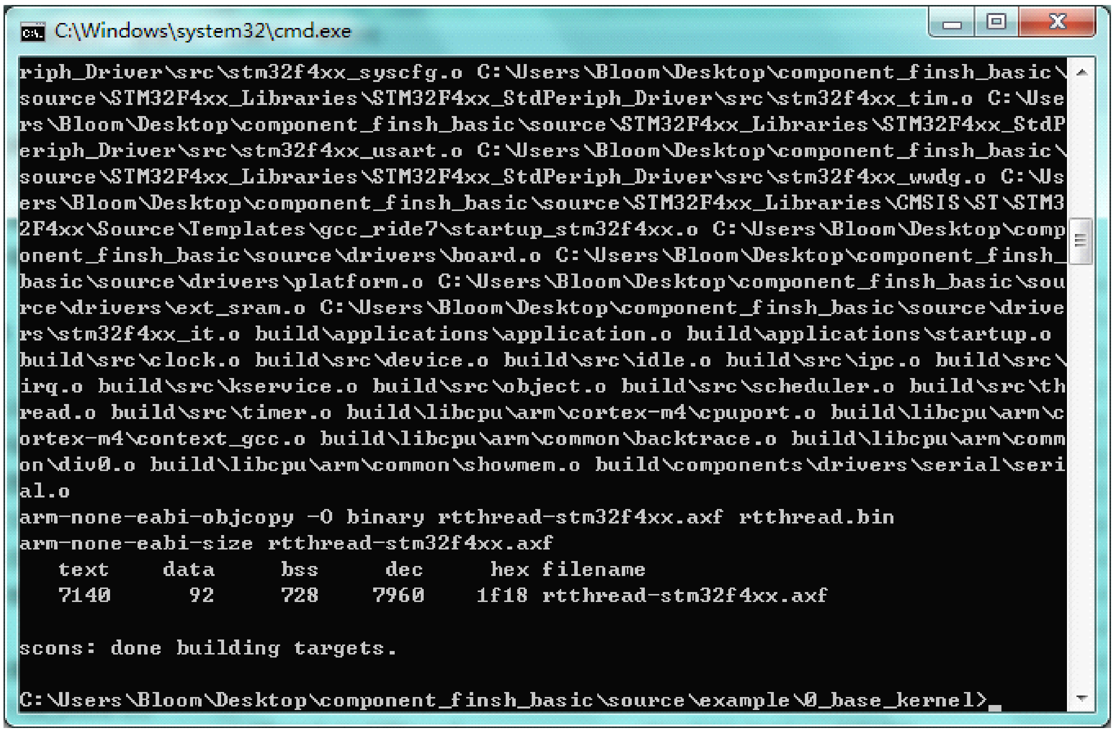
在 board.c 中，rt_hw_board_init()中相关初始化均可以注释掉，因为我们并未用到板上资源。

根据 wiki 文档上所说的，startup_stm32f4xx.s 中的 heap 大小也可以作出调整。根据.map 文件的分析，idle 进程的 stack 大小也可以进一步缩小。

于是乎，我们可以看到：



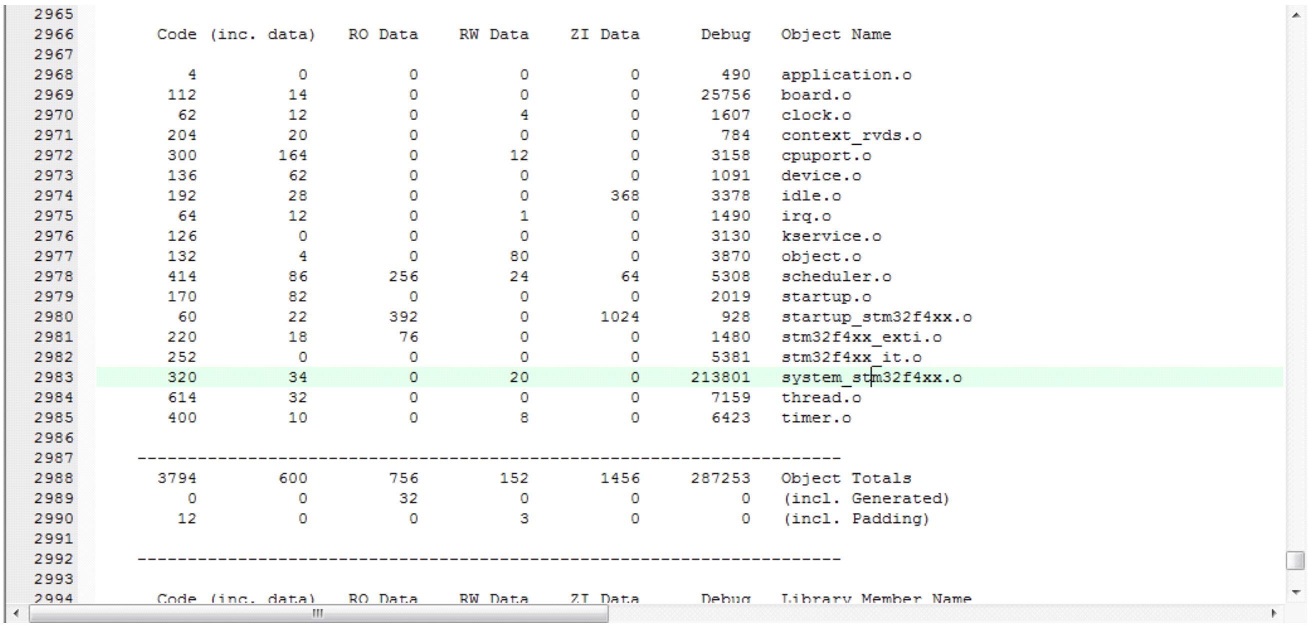
```
compiling serial.c...
linking...
Program Size: Code=4080 RO-data=756 RW-data=152 ZI-data=1552
User command #1: fromelf --bin .\build\rtthread-stm32f4xx.axf --output rtthread.bin
".\build\rtthread-stm32f4xx.axf" - 0 Error(s), 1 Warning(s).
```



```
riph_Driver\src\stm32f4xx_syscfg.o C:\Users\Bloom\Desktop\component_finsh_basic\
source\STM32F4xx_Libraries\STM32F4xx_StdPeriph_Driver\src\stm32f4xx_tim.o C:\Use
rs\Bloom\Desktop\component_finsh_basic\source\STM32F4xx_Libraries\STM32F4xx_StdP
eriph_Driver\src\stm32f4xx_usart.o C:\Users\Bloom\Desktop\component_finsh_basic\
source\STM32F4xx_Libraries\STM32F4xx_StdPeriph_Driver\src\stm32f4xx_wwdg.o C:\Us
ers\Bloom\Desktop\component_finsh_basic\source\STM32F4xx_Libraries\CMSIS\ST\STM3
2F4xx\Source\Templates\gcc_ride7\startup_stm32f4xx.o C:\Users\Bloom\Desktop\comp
onent_finsh_basic\source\drivers\board.o C:\Users\Bloom\Desktop\component_finsh_
basic\source\drivers\platform.o C:\Users\Bloom\Desktop\component_finsh_basic\sou
rce\drivers\ext_sram.o C:\Users\Bloom\Desktop\component_finsh_basic\source\drive
rs\stm32f4xx_it.o build\applications\application.o build\applications\startup.o
build\src\clock.o build\src\device.o build\src\idle.o build\src\ipc.o build\src\
irq.o build\src\kservice.o build\src\object.o build\src\scheduler.o build\src\th
read.o build\src\timer.o build\libcpu\arm\cortex-m4\cpuport.o build\libcpu\arm\c
ortex-m4\context_gcc.o build\libcpu\arm\common\backtrace.o build\libcpu\arm\comm
on\div0.o build\libcpu\arm\common\showmem.o build\components\drivers\serial\seri
al.o
arm-none-eabi-objcopy -O binary rtthread-stm32f4xx.axf rtthread.bin
arm-none-eabi-size rtthread-stm32f4xx.axf
   text    data     bss     dec     hex filename
   7140      92     728    7960    1f18 rtthread-stm32f4xx.axf

scons: done building targets.

C:\Users\Bloom\Desktop\component_finsh_basic\source\example\0_base_kernel>_
```

结果分析

通过裁剪，笔者通过 mdk 现在大概得到的 flash 大小约为 6k，ram 为 1.7k。此外，在裁剪过程中我们可以通过生成的 mdk 观察相关资源占用情况，



```
      Code (inc. data)   RO Data    RW Data    ZI Data      Debug   Object Name

         4          0          0          0          0        490   application.o
       112         14          0          0          0      25756   board.o
        62         12          0          4          0       1607   clock.o
       204         20          0          0          0        784   context_rvds.o
       300        164          0         12          0       3158   cpuport.o
       136         62          0          0          0       1091   device.o
       192         28          0          0        368       3378   idle.o
        64         12          0          1          0       1490   irq.o
       126          0          0          0          0       3130   kservice.o
       132          4          0         80          0       3870   object.o
       414         86        256         24         64       5308   scheduler.o
       170         82          0          0          0       2019   startup.o
        60         22        392          0       1024        928   startup_stm32f4xx.o
       220         18         76          0          0       1480   stm32f4xx_exti.o
       252          0          0          0          0       5381   stm32f4xx_it.o
       320         34          0         20          0     213801   system_stm32f4xx.o
       614         32          0          0          0       7159   thread.o
       400         10          0          8          0       6423   timer.o

    ----------------------------------------------------------------------
      3794        600        756        152       1456     287253   Object Totals
         0          0         32          0          0          0   (incl. Generated)
        12          0          0          3          0          0   (incl. Padding)

    ----------------------------------------------------------------------

      Code (inc. data)   RO Data    RW Data    ZI Data      Debug   Library Member Name
```