在 TMS320F28XXX DSP 上实现从 flash 拷贝整个程序到 ram 运行的方法

1. 前言

TMS320F28XXX DSP 里，代码从内部 flash 里运行，比从内部 RAM 里运行要慢 30%左右，所以对运行时间苛刻的程序直接在 flash 里运行，往往不能满足要求。故而，需要将代码拷贝到 RAM 以提高运行速度。TI 文档只提供了部分代码从 flash 拷贝到 RAM 中的方法。然而，在一些应用中，需要将整个代码段都拷贝到 RAM 中执行，以提高整体运行速度。本文通过对 TMS320F28XXX 的启动代码研究，来探讨如何在从 FLASH 启动后将整个代码段拷贝到 RAM 中，然后在 RAM 中运行的方法。

2. F28XXX 启动过程

F28XXX 启动代码固化在内部 ROM 中。当 F28XXX 上电或者热复位后，首先由芯片本身将一些寄存器初始化：

PIE disabled(ENPIE=0,VMAP=1,OBJMDE=0,AMODE=0,MOM1MAP=1),

然后 dsp 会对 XMPNMC 管脚采样，根据采样值的高低，来决定启动模式是“微处理器模式”还是“微计算机模式”。当 XMPNMC=0 时，为“微计算机模式”，此时，启动 ROM 存储器被使能而 XINTF Zone 7 被禁止。复位向量从内部启动 ROM 获取，启动 ROM 在复位期间一直被使能。

启动 ROM 里的复位向量（位于 0x3FFFC0）指向 InitBoot 函数(位于 0x3FFC00)。在完成器件初始化(InitBoot)之后，Boot loader 将检查 GPIO 管脚的状态，然后再决定选用的启动模式。启动模式有 4 种：跳转到 flash，跳转到 H0 SARAM,跳转到 OTP 或者调用片上启动程序。

InitBoot Function 所做工作有：1.初始化状态寄存器；2.将堆栈指针设为 0x400；3.读 CSM 密码保护部分；4.调用 SelectBootMode；5.调用 ExitBoot

在完成选择启动模式过程之后，根据选择的启动模式，dsp 会跳到相应的启动入口。也可以自己选择启动入口。这些入口地址都在这之前已经被 dsp 定义好的。

如果从 flash 启动，那么我们的管脚状态应该是

GPIOF4	GPIOF12	GPIOF3	GPIOF2	
(SCITXDA)	(MDXA)	(SPISTEA)	(SPICLK)	
内部上拉	无内部上拉	无内部上拉	无内部上拉	Mode Selected
1	x	x	x	Jump to Flash address 0x3F 7FF6

3. 搬移思路

将搬移代码烧在 flash 上，从 flash 启动之后，搬移代码会被执行。搬移代码做的工作就是将烧在 flash 其他区的代码拷贝到内部 RAM，然后经过初始化环境，PC 指针指向 RAM 里代码部分首地址，开始执行程序。这样，你的程序就在 RAM 中运行起来了。也就是说，我们需要两个程序，一个是搬移程序，启动时运行，用来拷贝代码到 RAM 中；一个就是主程序，

从 flash 里被搬移到 RAM 后在 RAM 中运行。这样 flash 里应该存有两个程序的代码。我们还要做的工作就是，把主程序烧写到 flash 里的某一块，这个块又不会影响 flash 启动时运行搬移程序。完成这个烧写过程的程序，我们称之为 flash 烧写程序。

这样，完成整个搬移过程，一共需要三个程序。两个程序被固化到 flash 里，一个程序在仿真器里运行完成烧写任务即可。

4. 搬移方法

1）首先你要用的主程序必须编译通过，并且通过仿真器在 RAM 里运行无问题。将主程序的 CMD 文件进行改写，保证程序段(.text 段)分配在连续的存储空间。程序从 flash 启动，所有初始化段链接在非易失存储器里，而非初始化段必须链接在易失存储器。我们可以把初始化段都放在一个连续的内部 RAM 空间，而非初始化段放在另一个内部 RAM 空间。如果你的代码不是很大，也可以都放在连续的 RAM 空间。但在实际项目中，通常你会遇到存储空间不够的问题。这时就要考虑将无关紧要的段放在另外的非程序空间了。

.cinit	Flash
.cio	RAM
.const	Flash
.econst	Flash
.pinit	Flash
.switch	Flash
.text	Flash
.bss	RAM
.ebss	RAM
.stack	Lower 64Kw RAM
.sysmem	RAM
.esysmem	RAM
.reset	RAM1

例如：

```
MEMORY
{
PAGE 0 :
    RAMH0          : origin = 0x3F8000, length = 0x002000

PAGE 1 :
    /* SARAM                              */
    RAMM0M1        : origin = 0x000000, length = 0x000800
    RAML0L1        : origin = 0x008000, length = 0x002000
}


SECTIONS
```

```
{
    /* Allocate program areas: */
    .reset                  : > RAMH0       PAGE = 0
    vectors                 : > RAMH0       PAGE = 0
    .cinit                  : > RAMH0       PAGE = 0
    .text                   : > RAMH0       PAGE = 0
    .const                  : > RAMH0       PAGE = 0
    .econst                 : > RAMH0       PAGE = 0
    .switch                         : > RAMH0       PAGE = 0

    /* Allocate data areas: */
    .stack                  : > RAMM0M1     PAGE = 1
    .bss                    : > RAML0L1     PAGE = 1
    .ebss                   : > RAML0L1     PAGE = 1
    .sysmem                 : > RAML0L1     PAGE = 1
}
```

2）其次，将该工程编译成功后，加载到内部 ram，仿真器自动完成必要的初始化环境之后，pc 指针应该指向_c_init00。

3）自制一个 flash 烧写程序，或者从网上下载其他网友的 flash 烧写程序，将目标地址放在除 flashJ 块以外的块中。烧写的长度不能小于被烧写的主程序长度。该烧写程序在 RAM 中运行。,其代码段不能和被烧写的主程序用的代码段存储区域相同，否则会破坏主程序在 ram 中的代码。

烧写之后，可以用 CCS 的 Save data 功能，来查看 flash 中的数值是否和 ram 里主程序空间数值一致。

4） 用 TI 的烧写插件烧写搬移程序。注意，该搬移程序要能在 dsp 启动后执行。并且，烧写的时候，不能将上一步烧到 flash 上的主程序代码擦除。搬移程序具体见下节。

5. 搬移程序具体实现方法

Boot.asm 文件内容：

```
    .def _InitBoot
    .ref _EntryAddr_H
    .ref _EntryAddr_L

    .sect ".InitBoot"
;
; _InitBoot
;
; 1) Initalizes the stack pointer
; 2) Sets the device for C28x operating mode
; 3) Calls the main boot functions
```

```
; 4) Calls an exit routine
;
_InitBoot:
; Initalize the stack pointer.
   MOV SP, #0 ; Initalize the stack pointer
; Initalize the device for running in C28x mode.
   C28OBJ ; Select C28x object mode
   C28ADDR ; Select C27x/C28x addressing
   C28MAP ; Set blocks M0/M1 for C28x mode
   CLRC PAGE0 ; Always use stack addressing mode
   MOVW DP,#0 ; Initialize DP to point to the low 64 K
   CLRC OVM
; Set PM shift of 0
   SPM 0
; Read the password locations – this will unlock the
; CSM only if the passwords are erased. Otherwise it
; will not have an effect.
   MOVL XAR1,#0x3F7FF8;
   MOVL XAR0,*XAR1++
   MOVL XAR0,*XAR1++
   MOVL XAR0,*XAR1++
   MOVL XAR0,*XAR1
; Cleanup and exit. At this point the EntryAddr
; is located in the ACC register
   BF _ExitBoot,UNC
;
; _ExitBoot
;
;
;This module cleans up after the boot loader
;
; 1) Make sure the stack is deallocated.
; SP = 0x400 after exiting the boot
; loader
; 2) Push 0 onto the stack so RPC will be
; 0 after using LRETR to jump to the
; entry point
; 2) Load RPC with the entry point
; 3) Clear all XARn registers
; 4) Clear ACC, P and XT registers
; 5) LRETR – this will also clear the RPC
; register since 0 was on the stack
;
```

```
_ExitBoot:
;
; Insure that the stack is deallocated
;
   MOV SP,#0
;
; Clear the bottom of the stack. This will endup
; in RPC when we are finished
;
   MOV *SP++,#0
   MOV *SP++,#0

   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
;
; Load RPC with the entry point as determined
; by the boot mode. This address will be returned
; in the ACC register.
;向堆栈中压入 0x3f8000,该地址即为主程序在 ram 中运行的首地址。
   MOV *SP++, #0x8000
   MOV *SP++, #0x3F

   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
   NOP

   POP RPC

   NOP
   NOP
```

```
   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
;
; Put registers back in their reset state.
;
; Clear all the XARn, ACC, XT, and P and DP
; registers
;
; NOTE: Leave the device in C28x operating mode
; (OBJMODE = 1, AMODE = 0)
;
   ZAPA
   MOVL XT,ACC
   MOVZ AR0,AL
   MOVZ AR1,AL
   MOVZ AR2,AL
   MOVZ AR3,AL
   MOVZ AR4,AL
   MOVZ AR5,AL
   MOVZ AR6,AL
   MOVZ AR7,AL
   MOVW DP, #0
;
; Restore ST0 and ST1. Note OBJMODE is
; the only bit not restored to its reset state.
; OBJMODE is left set for C28x object operating
; mode.
;
; ST0 = 0x0000 ST1 = 0x0A0B
; 15:10 OVC = 0 15:13 ARP = 0
; 9: 7 PM = 0 12 XF = 0
; 6 V = 0 11 M0M1MAP = 1
; 5 N = 0 10 reserved
; 4 Z = 0 9 OBJMODE = 1
; 3 C = 0 8 AMODE = 0
; 2 TC = 0 7 IDLESTAT = 0
; 1 OVM = 0 6 EALLOW = 0
; 0 SXM = 0 5 LOOP = 0
```

```
; 4 SPA = 0
; 3 VMAP = 1
; 2 PAGE0 = 0
; 1 DBGM = 1
; 0 INTM = 1
;
   MOV *SP++,#0
   MOV *SP++,#0x0A0B

   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
   NOP

   POP ST1
   POP ST0

   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
   NOP
;
; Jump to the EntryAddr as defined by the
; boot mode selected and continue execution
;
   LRETR
;
       .end
```

主程序调用

```
  unsigned long *srcAddr = (unsigned long *)0x3D8000;
unsigned long *desAddr = (unsigned long *)0x3F8000;
            InitSysCtrl();
```

```
    // Disable and clear all CPU interrupts:
    DINT;
    IER = 0x0000;
    IFR = 0x0000;

    // Initialize Pie Control Registers To Default State:
    InitPieCtrl();

    InitPieVectTable();

    //以下即将 flash 上的代码拷贝到 ram 中，根据自己需要，更改源地址和目标地址
for(i = 0; i < 0x2000; i++)
{
     *(desAddr + i) = *(srcAddr + i);
}

InitBoot();
```