

晶心科技 技術文章

如何在晶心平台實作 ROM patch Implement ROM patch on Andes platform

發表人：賴歆雅，技術經理，晶心科技股份有限公司

地址：新竹科學工業園區力行一路1號2樓

Tel : 886-3-666-8300 ext. 631

hylai@andestech.com

.....晶...心...科...技...新...聞...聯...絡...人.....

市場部 Janine 徐家玲經理

電話: 03-6668300 ext. 614

行動: 0932-315015

E-mail: Janine@andestech.com

Web: www.andestech.com

如何在晶心平台實作 ROM patch

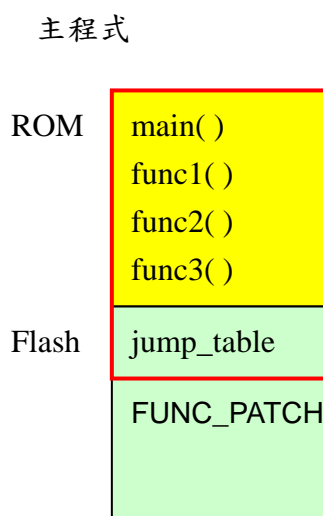
賴歆雅，技術經理，晶心科技股份有限公司

筆者曾協助多家公司工程師，在 AndesCore™上發展 firmware。我們發現，當客戶開發 Non-OS 的程式碼，最常遇到的問題在於開發者不知如何撰寫 linker script。網路上有 GNU ld 的使用文件，但是 linker script 的範例太少，尤其開發者需要撰寫進階的 linker script，常常不知如何下手。

本篇文章我們分享如何實作 ROM patch。使用晶心 CPU 建構的 embedded system，一般具有 CPU、週邊 IP 及 RAM、ROM。部份客戶使用 ROM code 開機，程式碼放在 ROM 內，data section 放在 SRAM 裡。ROM code 的特性是成本低，跟著 IC 光罩一起生產，當 IC 製作完成即不可修改，若有製作上的錯誤或是程式碼邏輯上的錯誤，只能用 ROM patch 的方式修補。也就是將需要修補的程式碼放到小容量的 flash 裡。這就是我們今天要分享的技術。

1. 主程式架構

首先介紹主程式的架構。IC 的 Memory layout 如下圖。



圖表1 主程式的 memory layout 圖

紅色框線的部份，為主程式編譯的範圍。主程式 main 會呼叫到 func1、func2 和 func3 這 3 個 function。

在上圖中，黃色區域是 IC 的 ROM，這部份的程式是 IC 製作出來即不可以改變。綠色部份是 flash。在圖中，flash 分成 2 區，一個是 jump_table，存放 func1~func3 的位址。剩餘的空間 FUNC_PATCH，預留給 patch 使用。

為了要修補 ROM 內的 function，所以規劃出 jump_table 區域，原本都是指向 ROM 的 function。如果 ROM 裡的部份 function 損壞或是需要改寫，就把 jump_table 改為指向 FUNC_PATCH 裡新建的 function。

1.1 原始程式碼

主程式的程式碼如下：(main.c)

技術文章發表, 請儘速發佈

2012 年 6 月 28 日

```
#include <stdio.h>

#include <stdlib.h>

int func1(int);

int func2(int);

int func3(int);

int num1=1;

int num2=2;

int num3=3;

typedef struct strfunptr {

    int (*func_a)(int);

    int (*func_b)(int);

    int (*func_c)(int);

}sfptr;

sfptr jump_table __attribute__((section ("FUNC_TABLE"))) = {func1, func2, func3};

int main(void) {

    printf("func1(30)=%d\n", jump_table.func_a(30));

    printf("func2(30)=%d\n", jump_table.func_b(30));

    printf("func3(30)=%d\n", jump_table.func_c(30));

    return EXIT_SUCCESS;

}

int func1(int x){

    return x*num1;

}
```

技術文章發表, 請儘速發佈

2012 年 6 月 28 日

```
int func2(int x){  
    return x*num2;  
}  
  
int func3(int x){  
    return x*num3;  
}
```

上面的程式碼中，第 16 行的程式碼 `__attribute__((section("FUNC_TABLE")))`，作用是將 `jump_table` 放在特定的 "FUNC_TABLE" section 裡。

1.2 主程式 linker script (僅列需要修改的部份)

```
FUNC_TABLE 0x510000 :  
{  
    *(.FUNC_TABLE)  
}
```

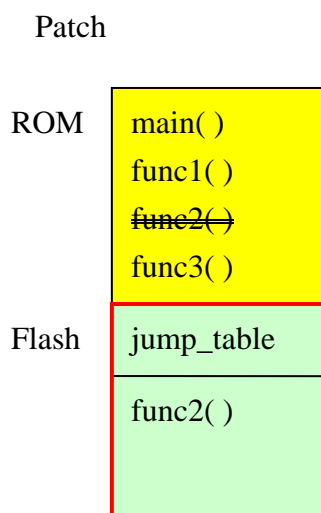
Flash 的位址由 0x510000 起，將 FUNC_TABLE 固定在 flash 的最開頭，語法如上。

1.3 主程式執行結果

```
func1(30)=30  
func2(30)=60  
func3(30)=90
```

2. 經過Patch之後的架構圖

假設 ROM 裡的 func2 損壞，要改用 flash 裡的 func2。需要更改指向 func2 的指標，及 func2 的內容。如下圖：



圖表2 ROM patch 的 memory layout 圖

用紅色框線標起來的地方，表示為 patch 編譯的範圍。其中 jump table 在這裡重新編譯，指向新的位址。

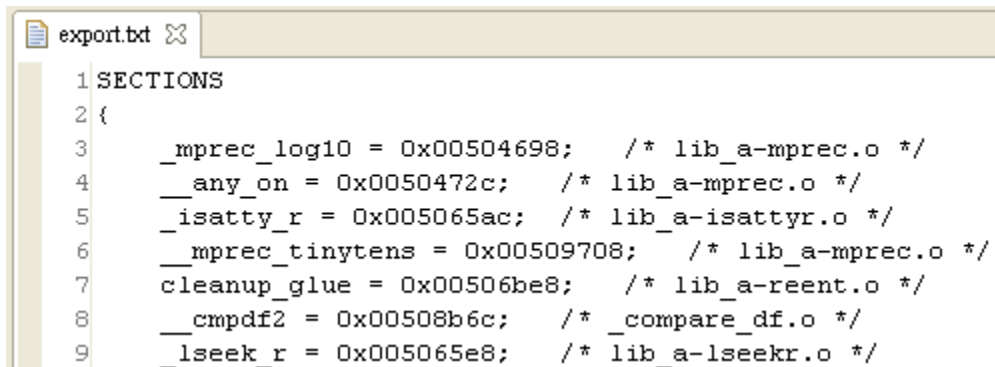
2.1 實作方法

(1) 匯出主程式的 symbol table。

在主程式的 Linker flags 加上 -Wl,--mgen-symbol-ld-script=export.txt，ld 會產生 export.txt 這個檔案，這個檔案包含了一個 SECTION block 以及許多變數的位址。如下圖所示

技術文章發表, 請儘速發佈

2012 年 6 月 28 日



```

1 SECTIONS
2 {
3     _mprec_log10 = 0x00504698; /* lib_a-mprec.o */
4     __any_on = 0x0050472c; /* lib_a-mprec.o */
5     _isatty_r = 0x005065ac; /* lib_a-isatty.o */
6     _mprec_tinytens = 0x00509708; /* lib_a-mprec.o */
7     cleanup_glue = 0x00506be8; /* lib_a-reent.o */
8     __cmpdf2 = 0x00508b6c; /* _compare_df.o */
9     _lseek_r = 0x005065e8; /* lib_a-lseekr.o */

```

圖表 3 主程式的 symbol

Linker script 在 import Main program 的 symbols 時，除了需要修改的 func2 不要 import 之外，其他的 symbols 全部要 import 進來。(將 export.txt 刪去這一行：func2 = 0x005001c4; /* ./main.o */)

(2) patch 在編譯之前，先匯入主程式的 symbol table。(將 export.txt 檔案放在一起編譯)。Patch 的 linker script 要匯入主程式的 symbol，寫法如下面紅色字體。

```

ENTRY(_start)

/* Do we need any of these for elf?
   __DYNAMIC = 0; */

INCLUDE "../export.txt"

SECTIONS
{

```

圖表 4 patch 的 linker script (節錄)

(3) patch 的程式碼裡如下，沒有 main function，也不要加入 startup files。改寫 func2。func2 放在 flash 的 FUNC_PATCH section。並且將 jump_table 裡的 func2，改成指向新的 func2。

技術文章發表, 請儘速發佈

2012 年 6 月 28 日

```
#include <stdio.h>
#include <stdlib.h>

extern int func1(int);
extern int func3(int);
int func2(int) __attribute__((section ("FUNC_PATCH")));
extern int num2;

typedef struct strfunptr {
    int (*func_a)(int);
    int (*func_b)(int);
    int (*func_c)(int);
}sfptr;

sfptr jump_table __attribute__((section ("FUNC_TABLE"))) = {func1,
func2, func3};

int func2(int x){
    return x*num2*100;
}
```

(4) patch 的 linker script，加入 FUNC_PATH 在 jump_table 之後。

```
FUNC_PATCH 0x510020 :
{
    *(.FUNC_PATCH)
}
```

圖表 6 Patch 的 linker script(節錄)

技術文章發表, 請儘速發佈

2012 年 6 月 28 日

3. 如何除錯

首先，將程式碼存放在 IC 的 ROM 及 flash 裡。(本文為了示範，我們的做法是在 AndeShape™ ADP-XC5 的 FPGA 板上，用 RAM 模擬 ROM 及 flash，分別將主程式和 patch 的 bin 檔 restore 到板子上。)

當 gdb debug 時，載入 patch 的 symbol。以下節錄 gdb 指令。

```
core0(gdb) file mainprog.adx

core0(gdb) add-symbol-file patch.adx 0x500000 -s FUNC_TABLE
0x510000 -s FUNC_PATCH 0x510020

core0(gdb) set $pc=0x500000

core0(gdb) b main

Breakpoint 1 at 0x50010c: file ../main.c, line 20.

core0(gdb) c

Breakpoint 1, main () at ../main.c:20

20          printf("func1(30)=%d\n",jump_table.func_a(30));

core0(gdb) s

func1 (x=30) at ../main.c:28

28          return x*num1;

core0(gdb) n

29      }

core0(gdb) s

main () at ../main.c:21
```

技術文章發表, 請儘速發佈

2012 年 6 月 28 日

```
21          printf("func2(30)=%d\n",jump_table.func_b(30));  
  
core0(gdb) s  
  
func2 (x=30) at ../patchprog.c:24  
  
24          return x*num2*100;  
  
core0(gdb)
```

圖表 7 gdb debug

上面過程中，先載入 main 的 symbol，再載入 patch 的 symbol 及 debug information。"add-symbol-file patch.adx 0x500000 -s FUNC_TABLE 0x510000 -s FUNC_PATCH 0x510020"是將 patch section 的 symbol 及 debug information 也載入 gdb 以 debug。讀者可以在 gdb 裡，打"help add-symbol-file"查閱 add-symbol-file 的用法。

3.1 主程式 patch 後的執行結果

```
func1 (30)=30  
func2 (30)=6000  
func3 (30)=90
```

5. 結語

目前晶心科技使用GNU的toolchain，其功能非常強大。讀者可多動手試試不同的linker script寫法，使得開發firmware更有彈性及效率。

(如果您對此文章有技術方面的疑問，歡迎來信至 hylai@andestech.com 與我們進行討論)。

技術文章發表, 請儘速發佈

2012 年 6 月 28 日

.....關於...晶...心.....



為因應國內外嵌入式系統應用的快速成長，2005 年晶心科技創立於新竹科學園區，致力於開發以 **32 位元處理器** 為核心的系統晶片設計平台 (**Processor-based SoC Platforms**)。晶心科技是國內唯一結合軟硬體平台及系統整合能力且專注於系統晶片核心開發的公司。

隨著電子產業產品日趨多功能化，更多廠商開始對處理器及設計平台要求更佳的整合性、延展性、設計彈性，以及高效能、低成本與低功率。這樣的複雜度已經超越傳統供應廠商所能提供的解決方案。晶心科技考量未來電子系統層面 (ESL) 更廣泛的設計要求，以創新的彈性配置平台 (**Configurable Platforms**)，搭配獨特的軟硬體智財，來滿足未來客戶對產品高品質及快速上市的需求。過去幾年晶心在自有 CPU 平台架構下都處於耕耘市場的階段，但是目前的 CPU 平台包括主流、低、中、高階等級的 **N8、N9、N10、N12** 及 **SN** 系列都已成熟到位，能滿足客戶各種應用面的全方位需求。

關於晶心科技提供之 **32 位元 CPU IP** 及 **ESL 系統開發工具**，歡迎請參閱晶心科技網站：www.andestech.com

.....新...聞...聯...絡...人.....

市場部 Janine 徐家玲經理

電話: 03-6668300 ext. 614

行動: 0932-315015

E-mail: Janine@andestech.com

Web: www.andestech.com