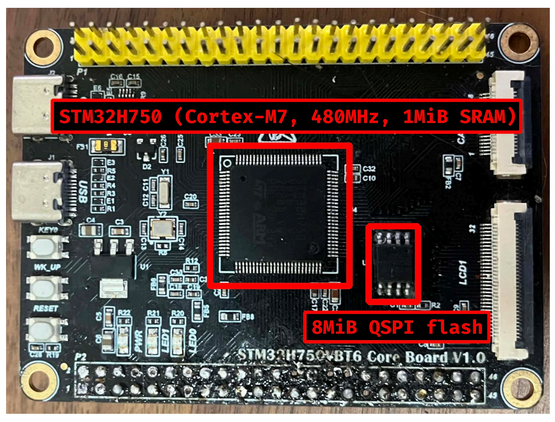
針對 1MB RAM 微控制器之 Linux 7.x 核心最佳化與移植

Optimize Linux Kernel To Fit Microcontroller with 1 MB RAM

作者：陳麒升　指導教授：陳奇業

1 簡介

在這個專題中,我在僅有 1 MiB RAM 的 STM32H750 上運行 Linux 7.0



2 動機

- 微控制器有 determinism 與低延遲性質
- 高性能 MCU（如 STM32H750），已滿足 Linux 核心的最低需求
- 在同時需要即時處理能力與運行複雜軟體程式的應用場景中，Linux noMMU 是個值得考慮的方案

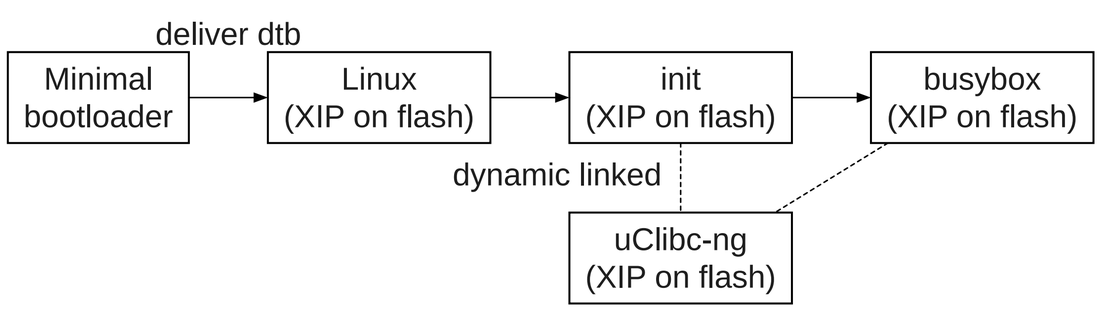
Linux 核心有以下好處：

- 豐富的軟體生態、子系統、驅動程式
- 支援 FDPIC 執行檔格式，且有 uClibc-ng 這樣的 C Library 支援，大多使用者程式只要切換工具鏈就能移植上去
- FDPIC 支援動態連結與 Application XIP
- CRA（網路韌性法案）規定軟體要長期更新，RTOS 的程式靜態連結在核心中，更新軟體等同於更新韌體，相較下，在 Linux 核心中只需要更新使用者程式或函式庫，減少重新整合與驗證的成本
- 內建系統分析工具

（以上參考自「Linux 應用於微控制器的考量」- jserv）

我發現，這樣的軟體模型即使各項元件都可以用了，卻缺乏完整運行案例（至少以開源而論），對於新版 Linux 核心更是如此。於是，本專題以 STM32H750 這樣驅動程式完善卻缺乏系統整合案例的硬體為例，探討如何從頭開始建立極度精簡的 Linux 核心，減少記憶體佔用，並整合 uClibc-ng 與工具鏈，建構相容的使用者程式。這樣的方法論可以運用在相似的嵌入式平台，並且此軟體模型在加上額外 DRAM 後即可因應更複雜的應用場景。

3 啟動流程



4 實作與成果

- **QEMU SoC model（軟體模擬）**

  為了更有效率的 GDB 除錯與做 runtime memory profile，我實現了 STM32H750 的 QEMU SoC model

- **SPARSEMEM_EXTREME**

  STM32H750 的 1MiB SRAM 不是連續的（128KiB / 512KiB / 288KiB / 64KiB），使用 SPARSEMEM_EXTREME 讓 Linux 核心可以充分利用這 1MiB SRAM

- **XIP (eXecuted In Place)**

  XIP 就是運行時期不把唯讀的資料載入到 RAM 中，對 kernel 和使用者程式都可以做

- **從 tinyconfig 開始（minimal Linux config）**

  tinyconfig + STM32 必要的 config 後，data + bss = 152KiB

  - 去除為 page table 預留的空間（32KiB）才能完整使用 128KiB
  - 關掉沒用到的 Linux config（44KiB）
  - 利用 System.map 發現多餘的結構體（3KiB）

  data + bss 降到 105KiB，成功啟動，但還沒有 rootfs，所以 panic

  ```
  Kernel panic - not syncing: No working init found
  ```

- **檔案系統（romfs）**

  比起 initramfs，romfs 不會被載入到 RAM 中，並且 romfs 可以支援 Application XIP。設定方式是在 device tree 中指定一段 flash 中的區域被解析成 romfs。

  ```
  VFS: Mounted root (romfs filesystem) readonly on device 31:0.
  ```

- **使用者程式**
  - 運行無 libc、FLAT binary 使用者程式
  - 運行 uClibc-ng、FLAT binary 使用者程式（XIP）
  - 運行 uClibc-ng、FDPIC 使用者程式（XIP）
- **運行 busybox**

  ```
  BusyBox v1.37.0 (2026-04-17 04:21:07 CST) hush - the humble shell
  Enter 'help' for a list of built-in commands.

  ~ # uname -a
  Linux (none) 7.0.0 #19 Fri Apr 17 04:51:22 CST 2026 armv7ml GNU/Linux
  ~ #
  ```

- **利用 debugfs 進行 memory profile**

  此時啟動後剩下 116 KiB 的 free RAM，下一步是減少動態分配的記憶體。可以在 QEMU 中加入假的記憶體，開 debugfs 來追蹤 slab 的用量。以此，發現 stm32_pinctrl_gpio 相關 driver 分配了異常多的空間，於是透過修剪 device tree，**free RAM** 到了 **228 KiB**

5 延伸討論與未來展望

在本專題的延伸討論中，我與黃敬群老師在 STM32F429 上進一步驗證相似的軟體模型在增加額外記憶體後,可以啟用更多 Linux 核心軟體堆疊與週邊（LCD display、touchscreen），在數 MiB RAM 情況下運行較複雜的使用者程式（觸控式人機介面）,並且近期在 **Embedded Linux Conference North America** 中發表我們的成果。

以下是一些未來可能可以做的事：

- 對 Linux 核心的即時處理能力做評估
- 評估精簡核心對降低功耗與啟動時間的成效
- 拿廉價的 SoC 來做產品

查看專案原始碼 / 按顆星星