NUC980 Linux BSP 使用手冊

Mar. 21, 2019 Rev. 1.02

內容

1 NUC980 Linux BSP 簡介

這包 BSP 支持了NUC980 系列芯片. 新唐科技的 NUC980 系列芯片是以 ARM926EJS 為核心的系统级單芯片. 包含了 16kB I-Cache, 16kB D-Cache, 16 kB SRAM, 16.5 kB IBR 以及MMU 記憶體管理模塊. 可以由 USB, NAND , SD/eMMC, 及 SPI Flash 開機.

NUC980 系列有豐富的高速周邊, 提供兩組 10/100 Mbps 網口, 一個 USB 2.0 高速 host/device 及一個 USB 2.0 高速 host 控制器, 最多六組 USB 1.1 host lite 接口, 兩個 CMOS感測器介面支持 CCIR601 及 CCIR656, 兩個 SD 接口支持 SD/SDHC/SDIO 卡, 一個 NAND Flash 接口支持 SLC 及 MLC, 一個 I2S 接口支持 I2S 及 PCM 協議. NUC980 系列同時內建了硬件加解密模塊, 支持 RSA, ECC, AES, SHA, HMAC 及亂數產生器 (RNG).

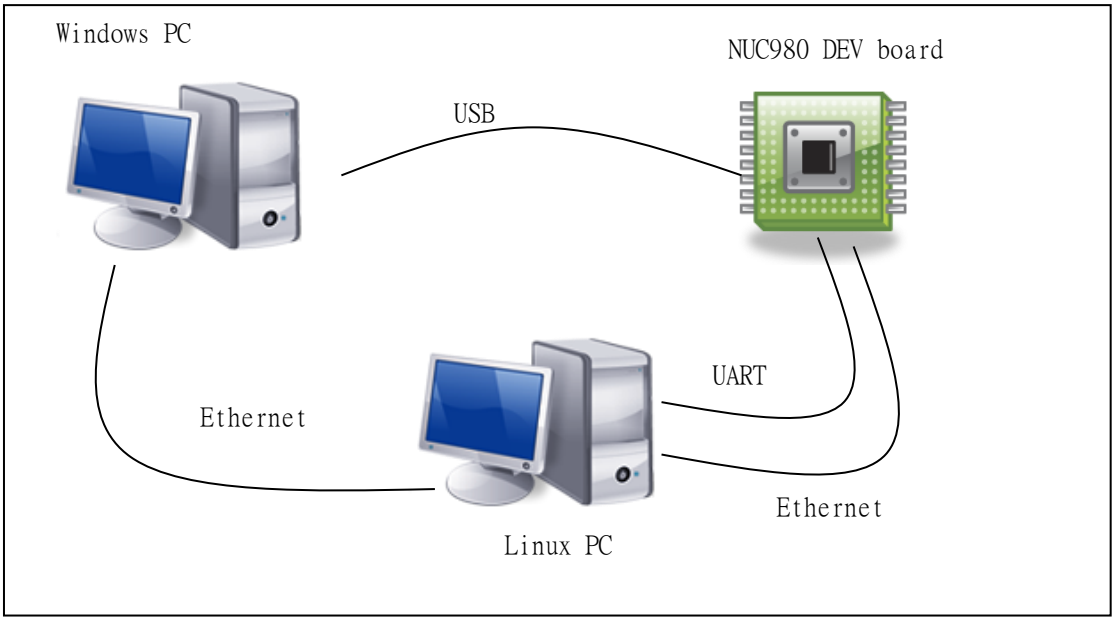
NUC980 系列提供最多高達十組 UART 接口, 兩組 ISO-7816-3 接口, 一組 Quad-SPI接口, 兩組 SPI 接口, 最多四組 I2C 接口, 四組 CAN 2.0B接口, 八個 PWM 輸出, 8 個 12-bit SAR ADC 通道, 六組 32-bit 計時器, WDT (看門狗), WWDT (窗口看門狗), 32.768 kHz 外接晶振及 RTC. The NUC980 系列另外提供了兩個組帶 10 個通道的 PDMA 來增加資料傳輸率.

這包 Linux BSP 包含了以下內容:

- Linux 4.4 內核源碼, 以及 NUC980 使用的驅動程式
- GCC 4.8.4 交叉編譯器, 支持 EABI.
- uClibc-0.9.33 庫文件
- Binutils-2.24 交叉開發工具
- 演示個接口功能的範例程式源碼, 以及一些開源軟件
- U-Boot 2016.11 源碼, 以及 NUC980 使用的驅動程式
- Windows 端燒錄程序 Nu-Writer, 以及所需的驅動
- 說明文檔

1.1 開發環境連線

在Linux 環境下, 基本的系統訊息以及 shell 環境的溝通都是透過串口來達成. 不論是U-Boot 或是Linux 均使用 UART0 來做為訊息溝通的接口. 在 U-Boot 環境下, 也支援了網口 TFTP 的傳輸. 另外新唐也提供了基於 Windows 平台的 USB 介面燒寫工具. 以下是開發環境連線的示意圖. 若是使用虛擬機, 則只需要一台 PC 即可



1.2 開發板設置

NUC980 系列芯片支持不同的開機模式, 可從 SPI, NAND, eMMC/SD 開機, 或是進入 USB ISP 模式. 這些設置是透過 PG[1:0] 的 jumper 控制. 另外, 因為複用腳位的關係, 開發版上會有些 jumper 須依不同系統需求來設置. 請參考開發版的文件來做系統相應的設置.

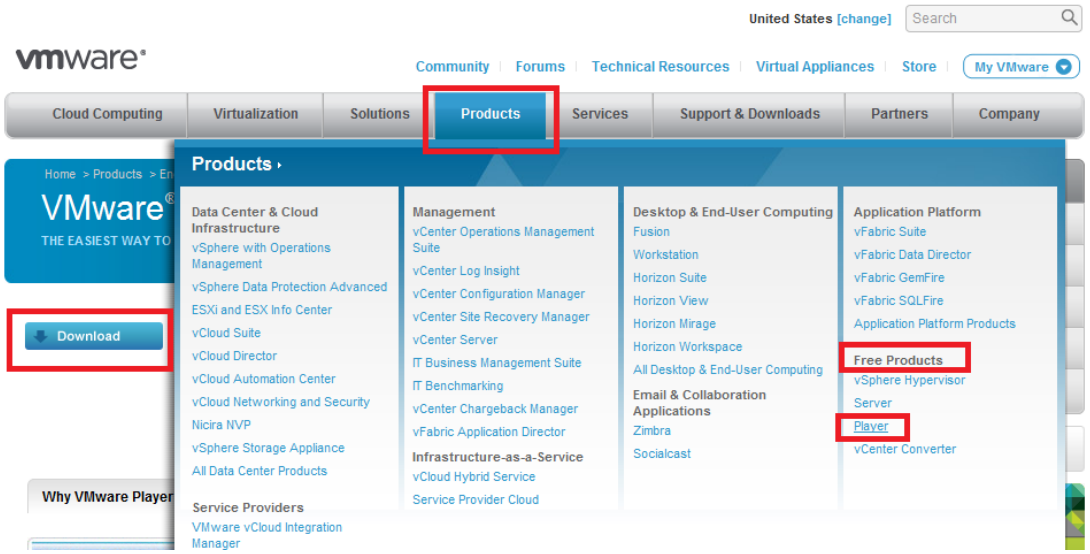
2 BSP 安裝

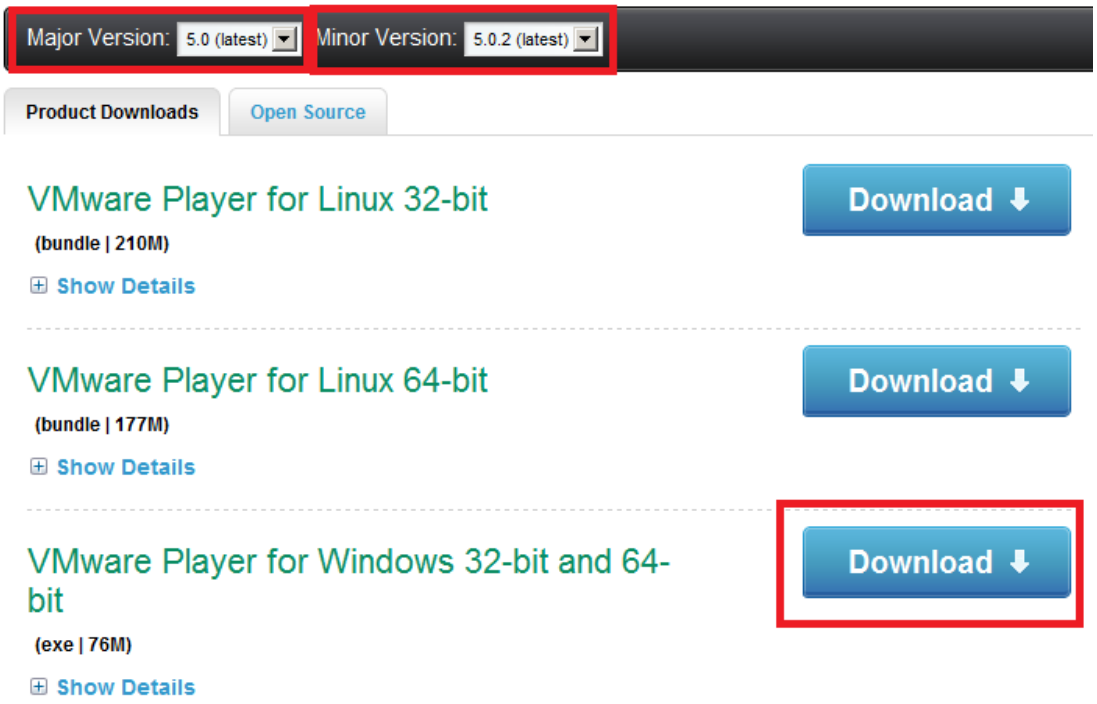
2.1 系統需求

NUC980 Linux BSP 提供了基於 Linux 作業系統的交叉編譯環境. 新唐有在不同的 x86 Linux 環境測試了本 BSP, 包含了 Ubuntu, CentOS, Debian…等. 因 Linux 發行版眾多, 系統設置會有些許差異, 有時使用這需更改系統設置, 使開發環境順利執行.
Linux 開發環境可選擇架設原生環境, 或是選擇架設於 Windows 作業系統中的虛擬機上. 本章節介紹了VMware 虛擬機安裝, CentOS Linux 開發環境安裝, 以及 BSP 的安裝步驟

2.2 VMware 虛擬機下載及安裝

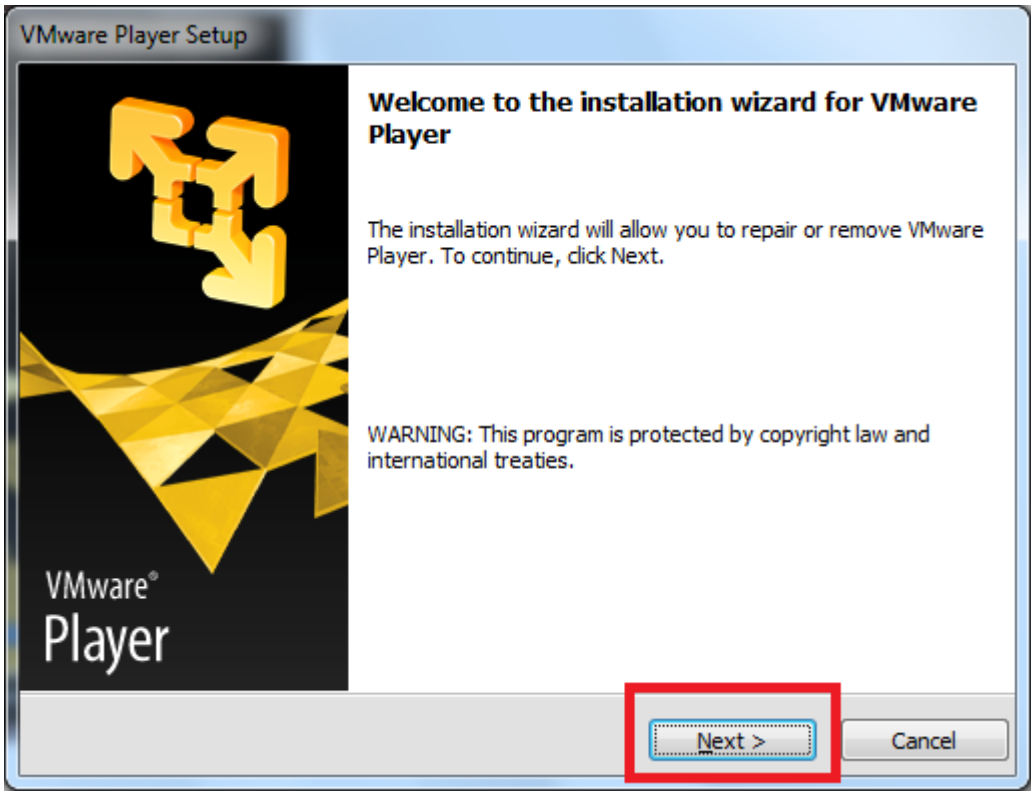
VMware 提供了免費的虛擬機 VMware player 5.0.2 供使用者下載. 從 VMware 官網 http://www.vmware.com/. 的頁面進入 “Products” → “Free Products” → “Player”, 然後按下 “Download” 鍵, 選擇 “5.0 (latest)” 作為 “Major Version” 以及 “5.0.2 (latest)” 作為 “Minor Version” 之後下載 “VMware Player for Windows 32-bit and 64-bit”. 請參考以下的截圖.



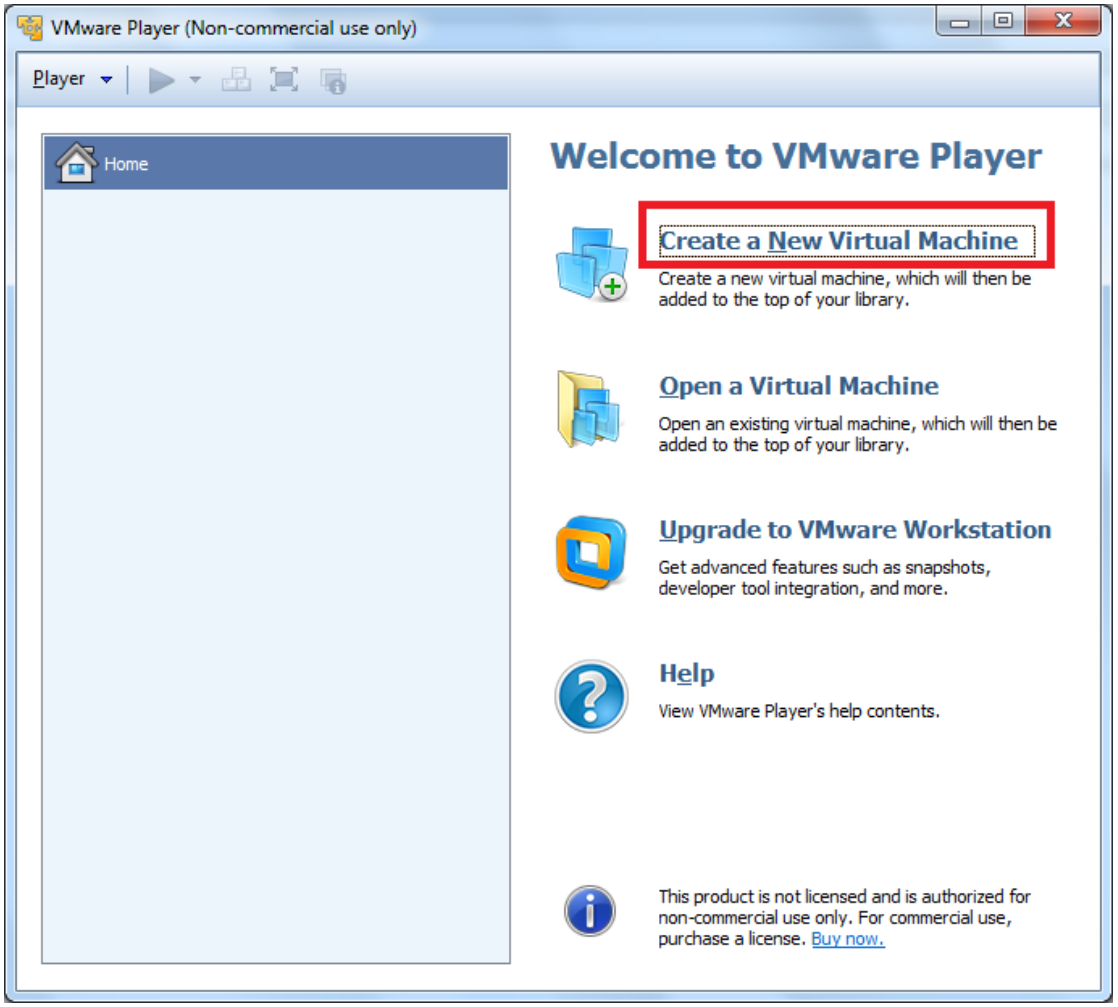


下載完成後, 請先雙擊下載的 VMware檔案.

然後按下 “Next” 繼續完成安裝步驟.

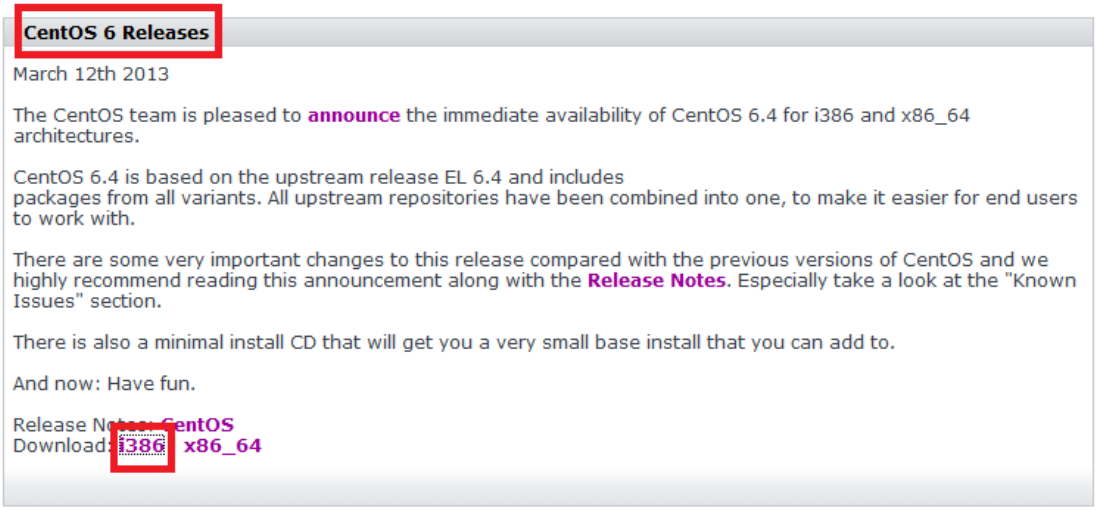


最後, 雙擊安裝完成的檔案來建立虛擬機.



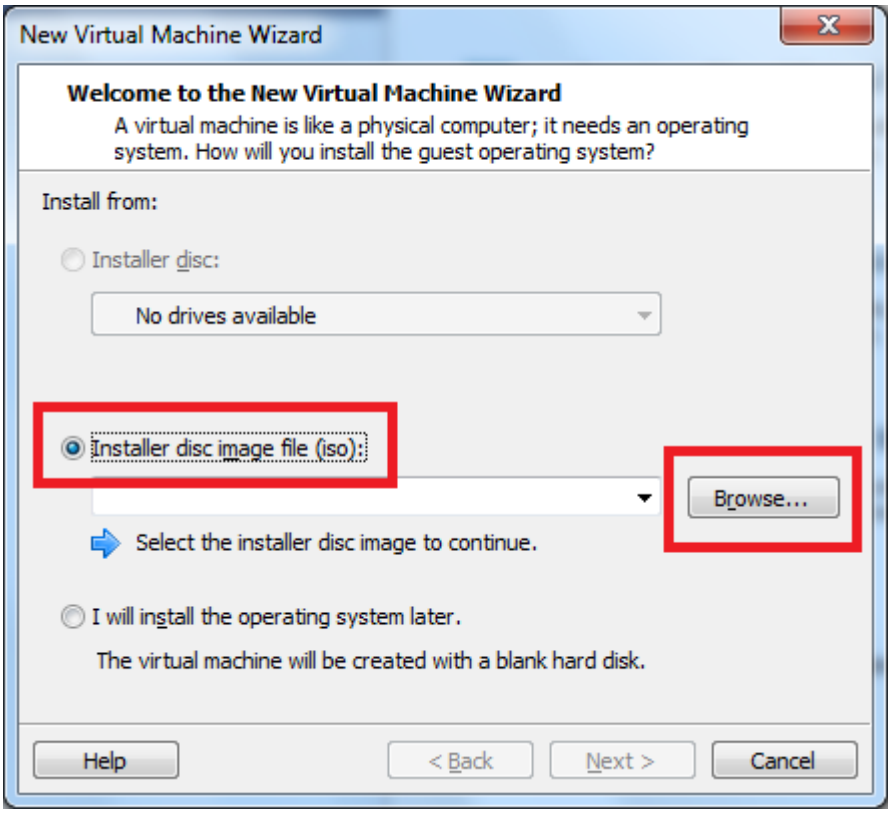
2.3 CentOS Linux 下載及安裝

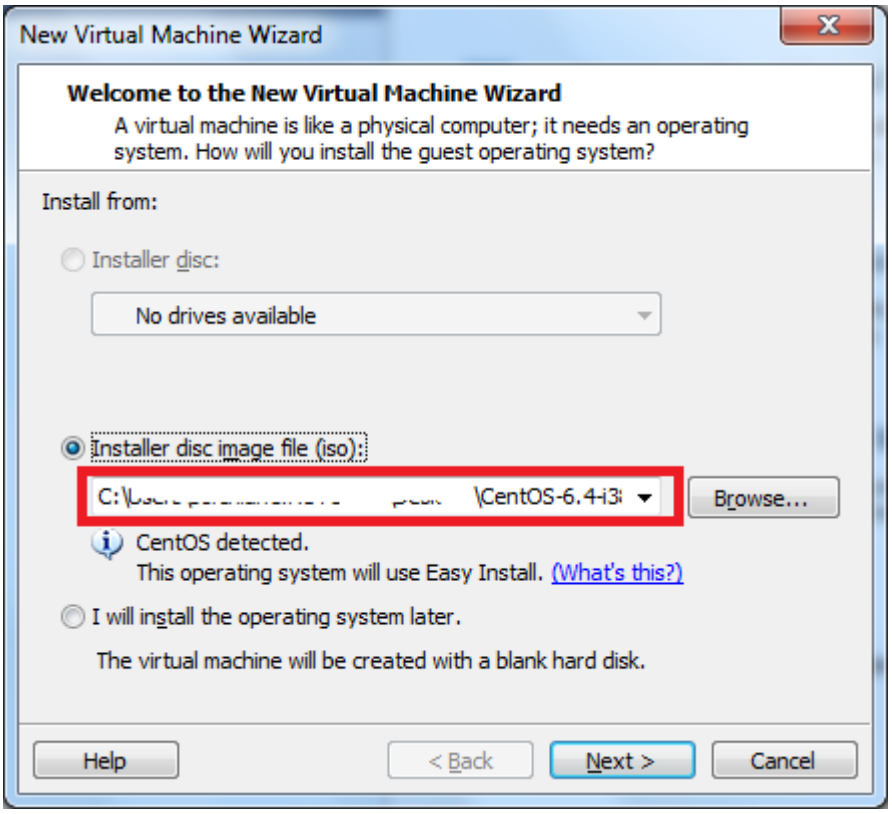
在此介紹了在 VMware 下的 CentOS 6.4 安裝步驟. 若是要安裝原生 PC 機, 步驟也類似. CentOS 6.4 請先連線至以下網址下載: http://www.centos.org/. 下載的網頁由 “CentOS 6 Releases” → “i386” 進入. 選擇 “CentOS-6.4-i386-bin-DVD 1.iso” 下載.



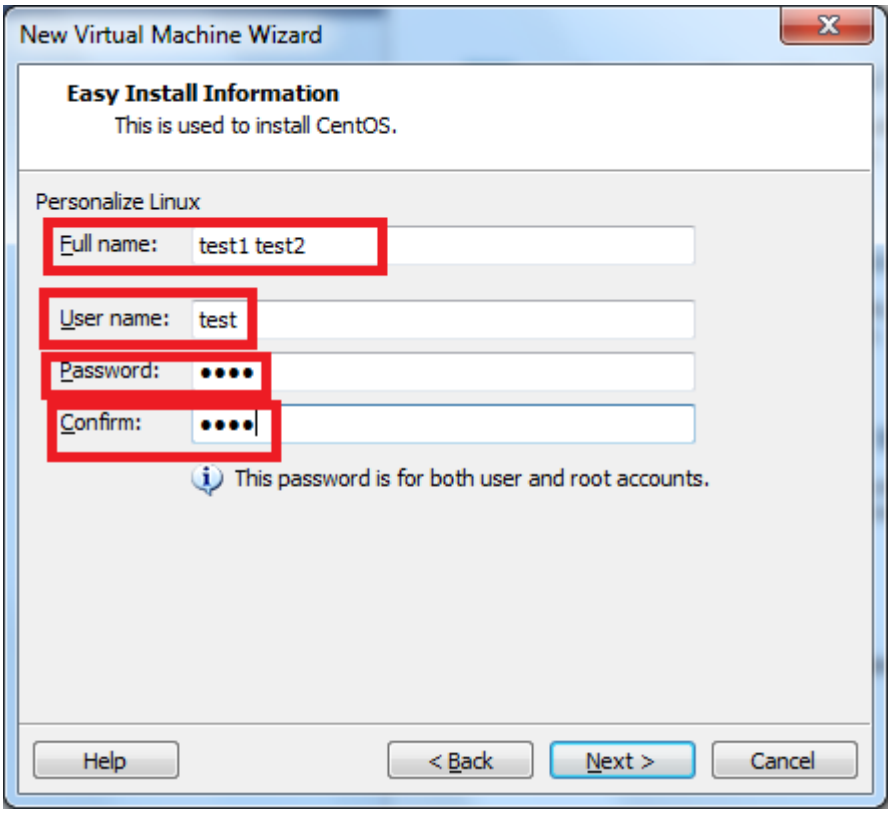
若是 VMware 已安裝完成, 點選 “Create a New Virtual Machine” 繼續安裝Linux 虛擬機, 或返回上一章節繼續完成 VMware 安裝.

首先, 點選 “Installer disc image file (iso):” and “Browse…” 並指定下載的 CentOS 6.4 iso 無安裝來源檔案.

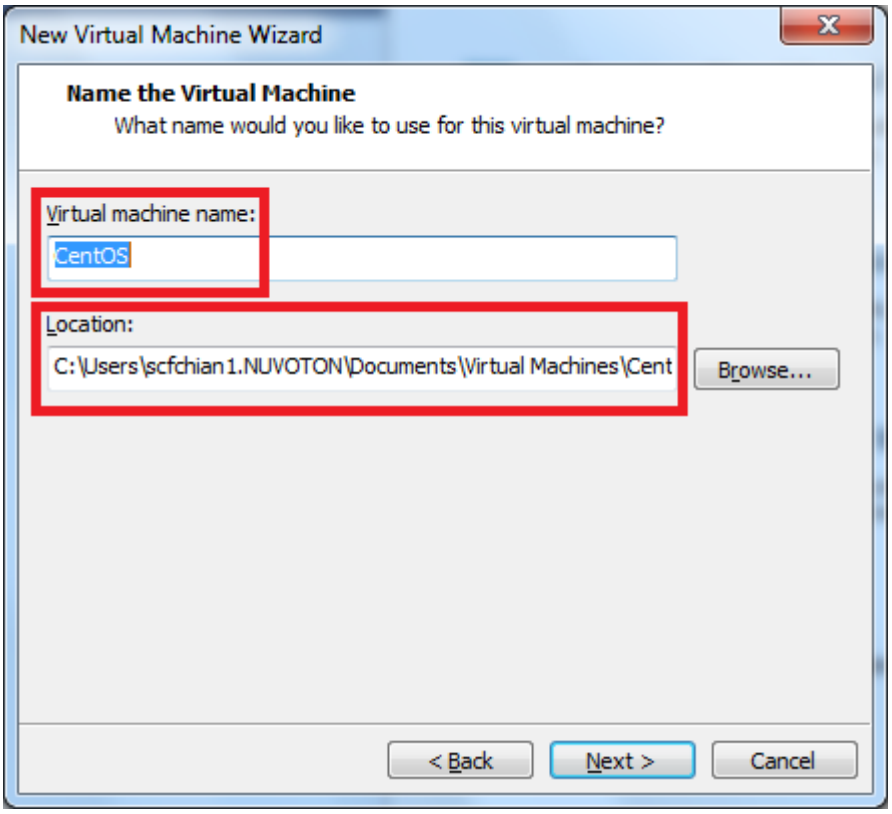

New Virtual Machine Wizard
Welcome to the New Virtual Machine Wizard
A virtual machine is like a physical computer; it needs an operating system. How will you install the guest operating system?
Install from:
Installer disc:
No drives available
Installer disc image file (iso):
Browse...
Select the installer disc image to continue.
I will install the operating system later.
The virtual machine will be created with a blank hard disk.
Help
< Back
Next >
Cancel



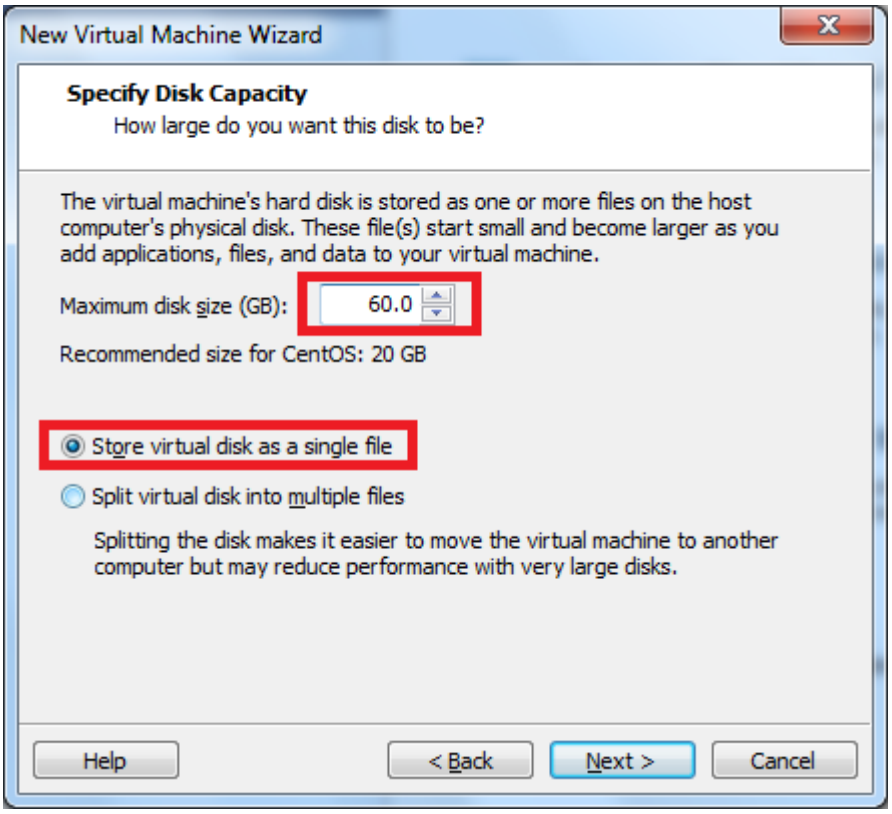
接著輸入 “Full name:”, “User name:”, “Password:”, and “Confirm:” 按下 Confirm 鍵. 請記得在此設置的用戶名稱及密碼, 它們將用來登入安裝的 CentOS 作業系統.



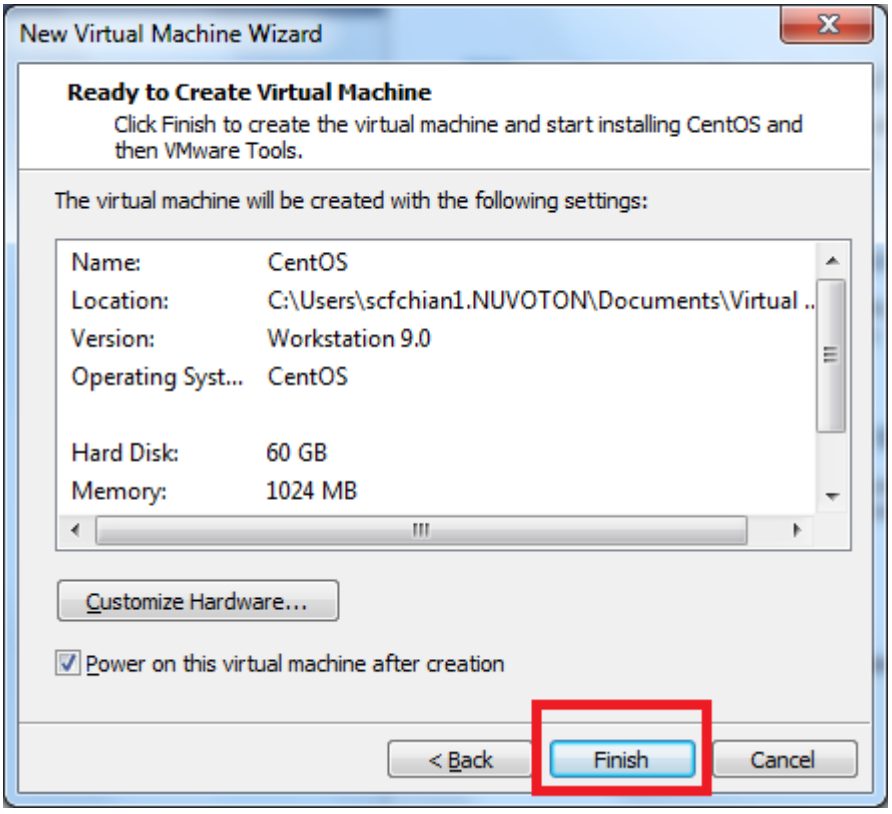
接下來, 若有必要則輸入 “Virtual machine name:” 以及 “Location:”, 否則保留默認值即可.



下一步則是輸入 “Maximum disk size (GB):” 值, 建議至少 60GB, 然後選擇 “Store virtual disk as a single file”.

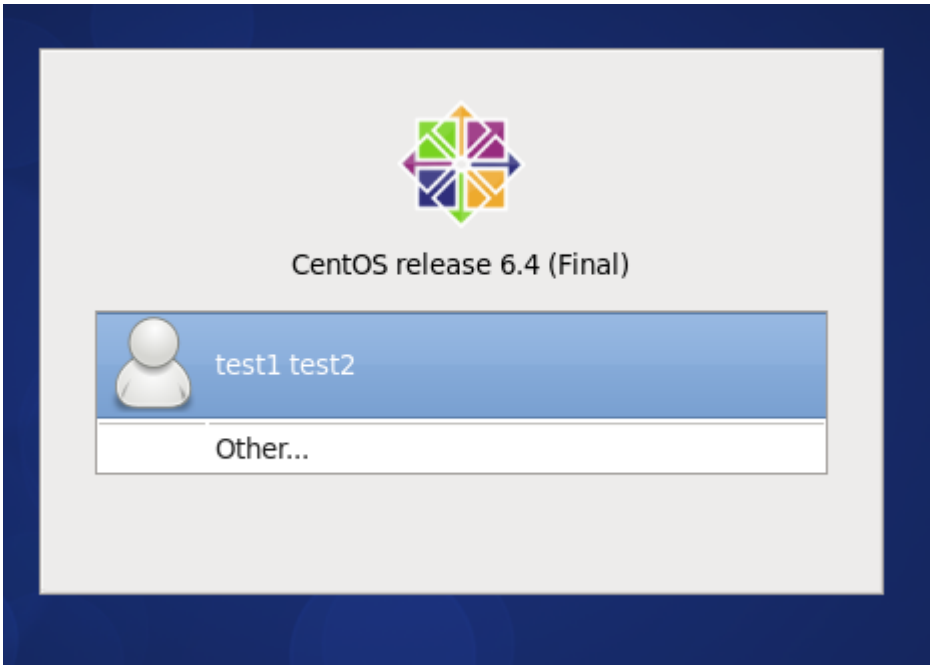

New Virtual Machine Wizard
Specify Disk Capacity
How large do you want this disk to be?
The virtual machine's hard disk is stored as one or more files on the host computer's physical disk. These file(s) start small and become larger as you add applications, files, and data to your virtual machine.
Maximum disk size (GB):
60.0
Recommended size for CentOS: 20 GB
Store virtual disk as a single file
Split virtual disk into multiple files
Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.
Help
< Back
Next >
Cancel

最後一步則是按下 “Finish” 來完成 CentOS 的設定.



VMware 會繼續完成 CentOS 的安裝.

安裝完成後, 會顯示 CentOS 的登入畫面, 此時可用之前設置的用戶名稱及密碼來登入



2.4 補齊元件

多數的 Linux 發行版不會將所有的原件都安裝. 但有些元件在使用 BSP 的開發過程中是必須的. 另外有些元件可讓開發的過程更加順利. 以下列了一些Linux 安裝過程會省略, 但必須及建議安裝的原件.

元件名稱	功用	必須/建議
patch	用來打補丁的工具	必須
libc6-dev	交叉編譯去所需動態鏈結的 32 位函式庫 (i386 版本)	必須
libncurses5-dev	設置內核介面所需使用的動態鏈結庫	必須
git-all	版本控管軟體	必須
Minicom 或cutecom	串口控制台, 可用來顯示以及控制 U-Boot 及內核.	建議

各個Linux 發行版的元件安裝介面不盡相同. Ubuntu 的使用者可以使用 apt-get 命令或是 Synaptic Package Manager 來安裝元件. 而 Fedora的使用者可以使用 rpm 命令或是 Package Manager來安裝元件. 請參考所使用的 Linux 發行版文件來安裝缺少的元件.

2.5 BSP 安裝步驟

Linux BSP 包含了三個目錄. 各目錄的內容列在下表:

目錄名稱	內容
BSP	一個包含了交叉編譯工具, 預編映像檔以及根文件系統的壓縮包.
Documents	BSP 相關文件
Tools	Windows 上的NuWriter燒錄工具以及驅動程式. 以及 SD Writer.

BSP 的源碼可以使用 repo 工具下載. 以下使使用 repo 下載的方式.
確認帳號主目錄下存在 bin/ 目錄, 且路徑有加入path 環境變數中.

```
$ mkdir ~/bin

$ export PATH=~/bin:$PATH
```

下載 repo 工具並設定可執行屬性.

```
$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo

$ chmod a+x ~/bin/repo
```

產生一個空的工作目錄.

```
$ mkdir WORKING_DIR

$ cd WORKING_DIR
```

設定 git 使用的姓名及電子郵件地址.

```
$ git config --global user.name "Your Name"

$ git config --global user.email "you@example.com"
```

使用以下兩個命令其中之一下載 NUC980 BSP 的 manifest 檔案. 第一個是從 Github 下載, 第二個是從 Gitee 下載. 使用者依據所在的的連線速度擇一使用即可

```
$ repo init -u git://github.com/OpenNuvoton/manifest.git -b nuc980-2019.09 -m
github.xml
```

或

```
$ repo init -u https://gitee.com/OpenNuvoton/manifest.git -b nuc980-2019.09 -
m gitee.xml
```

接著用以下命令下載源碼.

```
$ repo sync
```

下載完源碼後, 請將 BSP 目錄中的壓縮包複製到 Linux 開發機器上. 並使用以下的命令解壓

縮 :

```
$ tar zxvf nuc980bsp.tar.gz
```

在此目錄中有安裝腳本 install.sh. 此腳本需要管理者權限才可以執行. 可以選擇使用 “su” 命令切換到管理者來執行:

```
$ su

Password: (Enter password of root)

# ./install.sh
```

或是使用 sudo 來執行安裝腳本 (若是安裝的 Linux 沒有開放 root 權限, 例如 Ubuntu, 則可以使用本方式來安裝 BSP)

```
# sudo ./install.sh
```

以下為安裝過程的控制台輸出, 過程中輸入的目錄需要跟之前設定的 WORKING_DIR 相同.

```
Now installing arm_linux_4.8 tool chain to /usr/local/

Setting tool chain environment

Installing arm_linux_4.8 tool chain successfully

Install rootfs, applications, u-boot and Linux kernel

Please enter absolute path for installing(eg:/home/<user name>) :

BSP will be installed in /<path you input>/nuc980bsp

/home/someone

Extract rootfs and pre-build images

...

...

NUC980 BSP installion complete
```

若是使用的 Linux 開發環境之前已經安裝過新唐提供的交叉編譯工具, 安裝腳本會詢問是否須複寫編譯工具, 否則腳本並不會詢問使用者, 而是直接在 /usr/local/arm_linux_4.8目錄安裝編譯工具. 在第一種已安裝過編譯工具的情況下, 若是要複寫, 可按Y (或是 yes、y、YES), 然後按 Enter 鍵.

安裝完成交叉編譯工具後, 安裝腳本會詢問安裝 Linux 內核, U-Boot, 以及範例程序的***絕對路徑***. 下表列出了會裝在指定目錄中的項目.

目錄名稱	內容

applications	範例程序以及開源軟件，例如 busybox，wireless tool...
buildroot	可用來編譯內核，交叉編譯工具，應用程序... 等的集合.
image/kernel	使用默認設置預先編譯好的內核
image/U-Boot	使用默認設置預先編譯好的支持 NAND 或 SPI flash 的 U-Boot 執行檔及環境變數env.txt檔。其中 U-Boot 的默認執行位址均為 0xE00000。
linux-4.4.y	內核源碼
rootfs	根文件系統
u-boot-2016.11	U-Boot V2016.11 源碼
nuwriter	Linux 下使用的命令行介面 NuWriter

安裝腳本會嘗試將安裝的目錄設置正確權限，並將交叉編譯器的路經加至系統搜尋路徑($PATH)中. 但在有些 Linux 版本中, 可能發生無法正確設置的問題. 此時需麻煩使用者手動設置正確的權限並且將/usr/local/arm_linux_4.8/bin加到 $PATH 中.

請注意, 在安裝完成後, 使用者須先登出再登入, $PATH 的設定才會生效.

2.6 透過 Git 下載

使用者可以自行至源碼倉庫複製源碼. 複製完後，可以至源碼倉庫以git pull 命令同步最新的更新. 以下列出各個源碼倉庫的鏈結. Git 的操作指令不在本文件的說明範圍，但可以至 https://git-scm.com/ 查詢基本的操作方式.

源碼	倉庫鏈結
Applications	https://github.com/OpenNuvoton/NUC980_Linux_Applications.git https://gitlab.com/OpenNuvoton/NuMicro-ARM7-ARM9-Family/NUC980_Linux_Applications.git https://gitee.com/OpenNuvoton/NUC980_Linux_Applications.git
buildroot	https://github.com/OpenNuvoton/NUC980_Buildroot.git https://gitlab.com/OpenNuvoton/NuMicro-ARM7-ARM9-Family/NUC980_Buildroot.git https://gitee.com/OpenNuvoton/NUC980_Buildroot.git
linux-4.4.y	https://github.com/OpenNuvoton/NUC980-linux-4.4.y.git https://gitlab.com/OpenNuvoton/NuMicro-ARM7-ARM9-Family/NUC980-linux-4.4.y.git https://gitee.com/OpenNuvoton/NUC980-linux-4.4.y.git

uboot.v2016.11	https://github.com/OpenNuvoton/NUC970_U-Boot_v2016.11.git https://gitlab.com/OpenNuvoton/NuMicro-ARM7-ARM9-Family/NUC970_U-Boot_v2016.11.git https://gitee.com/OpenNuvoton/NUC970_U-Boot_v2016.11.git
NuWriter	https://github.com/OpenNuvoton/NUC980_NuWriter.git https://gitlab.com/OpenNuvoton/NuMicro-ARM7-ARM9-Family/NUC980_NuWriter.git https://gitee.com/OpenNuvoton/NUC980_NuWriter.git
Linux Command Line NuWriter	https://github.com/OpenNuvoton/NUC980_NuWriter_CMD.git https://gitlab.com/OpenNuvoton/NuMicro-ARM7-ARM9-Family/NUC980_NuWriter_CMD.git https://gitee.com/OpenNuvoton/NUC980_NuWriter_CMD.git
SD Writer	https://github.com/OpenNuvoton/NUC980_SDWriter.git https://gitlab.com/OpenNuvoton/NuMicro-ARM7-ARM9-Family/NUC980_SDWriter.git https://gitee.com/OpenNuvoton/NUC980_SDWriter.git

3 Linux 內核

3.1 內核設置介面

Linux 內核支持各式不同的設置. 可將不需要的功能移除, 只保留需要的功能. 減少內核所消耗的資源.

要進入內核設置頁面, 請在 linux-3.10.x 目錄下輸入 “make menuconfig” 的命令. 即可進入內核設置頁面.

內核設置選單是多層次的. 在本頁面內, 可透過上, 下, 左, 右, 四個方向鍵控制選單的位置. 上, 下鍵選擇要控制的內核功能. 左, 右鍵則是選擇了下排的功能按鈕. 要進入更深一層的選單時, 按下 “Enter” 鍵即可.

下一排的按鈕有五個, 停留在 “Select” 時, 可透過空白鍵始能及失能內核功能. 顯示[ ] 代表此功能失能. [*] 代表始能. [M] 代表此功能邊一層模塊, 可動態加載. 下排按鈕停留在 “Exit” 時, 按空白鍵可帶往上層選單, 若已在最上層選單, 回出現提示是否要儲存並退出. 下排按鈕停留在 “Help” 時, 按下空白鍵可顯示幫助畫面. 下排按鈕停留在 “Save” 以及 “Load” 則是用來儲存當前設置, 或是加載設置.

在設置完成後, 內核的設置檔案名稱為 “.config”, 放置在 linux-3.10.x 目錄下.

3.2 默認設置

新唐為NUC980 系列芯片提供了默認設置. 建議使用這要更改內核設置前, 先加載默認設置, 再進行更改. 使用者可透過 “make nuc980_defconfig”有時因內核設置錯誤造成無法開機時, 也可使用此命令將內核設置還原到可開機的狀態.

.

3.3 Linux 內核設置

本節依據不同的驅動或功能, 介紹所需使能的內核設置.

3.3.1 基本系統設置

- 掛載模塊支持設置
  有些驅動程式只支持動態加載, 例如 USB WiFi 驅動程序, 或是 USB device 的驅動… 等. 請依照下的設定使能掛載模塊支持. 在上電進入 shell 後, 可使用 “insmod <模塊名稱>“ 來加載模塊.

```
[*] Enable loadable module support  --->
```

- 卸載模塊支持設置
  使能掛載模塊支持後, 若是系統需要支持動態卸載, 請依照下的設定使能掛載模塊支持. 要卸載模塊時, 可使用 “rmmod <模塊名稱>“ 來卸載模塊.

```
[*] Enable loadable module support  --->
```

```
  [*]   Module unloading
```

- 開機命令設置 – 以RAM為根文件系統的設定
  開機命令可以用來設置系統. 包含了根文件系統型態, 記憶體大小, console 波特率…等. 以下只是一個簡單的設置, 內核還另外支持了眾多的命令. 可參考 Documentation/kernel-parameters.txt 中的說明.

```
Boot options  --->

  (root=/dev/ram0 console=ttyS0, 115200n8 rdinit=/sbin/init mem=64M) Default
kernel command string

        Kernel command line type (Use bootloader arguments if available)  ---
>
```

- 開機命令設置 – 以SPI Flash JFFS2文件系統為根文件系統的設定
  開機命令可以用來設置系統. 包含了根文件系統型態, 記憶體大小, console 波特率…等. 使用這個設定需要啟動JFFS2文件系統 (請參考 5.3.4 文件系統設置), 並且取消RAM根文件系統設定.

```
General setup  --->

  [ ] Initial RAM filesystem and RAM disk (initramfs/initrd) support
```

  下列範例是需要先將根文件系統使用mkfs.jffs2工具製作成JFFS2格式的鏡像文件再燒入至mtdblock1所在的位置, 當內核啟動會依據下列開機命令將根文件系統帶起.

```
Boot options  --->

  (root=/dev/mtdblock1 rw rootfstype=jffs2 console=ttyS0,115200n8
rdinit=/sbin/init mem=64M) Default kernel command string

        Kernel command line type (Use bootloader arguments if available)  ---
>
```

- 開機命令設置 – 以NAND Flash YAFFS2文件系統為根文件系統的設定
  開機命令可以用來設置系統. 包含了根文件系統型態, 記憶體大小, console 波特率…等. 使用這個設定需要啟動YAFFS2文件系統 (請參考 5.3.4 文件系統設置), 並且取消RAM根文件系統設定.

```
General setup  --->

  [ ] Initial RAM filesystem and RAM disk (initramfs/initrd) support
```

  下列範例是需要先將根文件系統製作成YAFFS2格式的鏡像文件, 再燒入至mtdblock2所在的位置, uboot環境變數也要設定, 當內核啟動會依據下列開機命令將

根文件系統帶起.

```
Boot options  --->

  (noinitrd root=/dev/mtdblock2 rootfstype=yaffs2 rootflags=inband-tags
console=ttyS0, 115200n8 rdinit=/sbin/init mem=64M) Default kernel command
string

        Kernel command line type (Use bootloader arguments if available)  ---
>
```

如果沒有設定uboot環境變數, 要直接由內核變數帶起則設定如下.

```
Boot options  --->

  (noinitrd root=/dev/mtdblock2 rootfstype=yaffs2 rootflags=inband-tags
console=ttyS0, 115200n8 rdinit=/sbin/init mem=64M
mtdparts=nand0:0x200000@0x0(u-boot),0x1400000@0x200000(kernel),-(user)
ignore_loglevel) Default kernel command string

        Kernel command line type (Use bootloader arguments if available)  ---
>
```

- 開機命令設置 – 以NAND Flash UBIFS文件系統為根文件系統的設定
  開機命令可以用來設置系統. 包含了根文件系統型態, 記憶體大小, console 波特率…等. 使用這個設定需要啟動UBIFS文件系統 (請參考 5.3.4 文件系統設置), 並且取消RAM根文件系統設定.

```
General setup  --->

  [ ] Initial RAM filesystem and RAM disk (initramfs/initrd) support
```

下列範例是需要先將根文件系統製作成UBIFS格式的鏡像文件, 再燒入至mtd2所在的位置, 當內核啟動會依據下列開機命令將根文件系統帶起.

```
Boot options  --->

  (noinitrd ubi.mtd=2 root=ubi0:system rw rootfstype=ubifs console=ttyS0,
115200n8 rdinit=/sbin/init mem=64M) Default kernel command string

        Kernel command line type (Use bootloader arguments if available)  ---
>
```

- 開機命令設置 – 以NFS文件系統為根文件系統的設定
  在開發初期，開發者可能需要經常更改或更換rootfs，把rootfs用NFS的方式掛載，這樣可以大幅減少開發的時程。

```
Boot options  --->
```

```
  (noinitrd root=/dev/nfs nfsroot=x.x.x.x:/path_to_nfs_rootfs
ip=y.y.y.y:z.z.z.z:g.g.g.g:m.m.m.m console=ttyS0,115200n8 rdinit=/sbin/init
mem=64M) Default kernel command string
```

其中，x.x.x.x和z.z.z.z均代表NFS伺服器的ip位置，y.y.y.y代表本機的ip位置，g.g.g.g代表gateway的ip位置，m.m.m.m代表netmask的ip位置。
除了設置boot options之後，還需要啟動網路功能(可參考3.3.2)，另外以下功能也需一並啟動。

```
[*] Networking support  --->

  Networking options  --->

         [*]   IP: kernel level autoconfiguration
```

當然還需要啟動NFS的功能。

```
File systems  --->

   [*] Network File Systems  --->

   <*>   NFS client support

   [*]   Root file system on NFS
```

- 開機命令設置 – 支持設備樹
  若是需要使用設備樹功能, 請依照以下的方式設置.

```
[*] Flattened Device Tree support

[*]   Support for the traditional ATAGS boot data passingFile systems  --->
```

3.3.2 網絡功能設置

- TCP/IP 設置
  要使能基本的網絡功能, 請依照以下設置即可.

```
[*] Networking support  --->

  Networking options  --->

        <*> Packet socket

        <*> Unix domain sockets

        [*] TCP/IP networking

        [*]   IP: multicasting
```

- WiFi無線網路設置
  若是要使用無線網路設備, 除了以上的 TCP/IP 設置需使能外, 以下的設置也需一併

使能

```
[*] Networking support  --->
  [*]   Wireless  --->
        <*>  Nuvoton external WiFi driver support
        <*>   cfg80211 - wireless configuration API
        [*]     cfg80211 wireless extensions compatibility
        <*>  Generic IEEE 802.11 Networking Stack (mac80211)


Device Drivers  --->
  [*] Network device support  --->
        [*]   Network core driver support
        [*]   Wireless LAN  --->
```

3.3.3 驅動設置

- Audio 音效接口設置
  I²S 音效接口的設置如下:

```
Device Drivers  --->
  <*> Sound card support  --->
        <*>   Advanced Linux Sound Architecture  --->
              <*>   ALSA for SoC audio support  --->
                    <*>   SoC Audio for NUC980 series
                    <*>     NUC980 I2S support for demo board
                                I2S Mode Selection (Master Mode)  --->
```

I²S 支持主模式或從模式, 使用者可透過選單決定運作的模式.
I²S選定時, NAU8822 codec驅動將被自行被選中, 另外,使用時必須同時啟動I²C功能, 系統才能正確偵測到NAU8822裝置.

如果audio應用程式使用舊式OSS架構編寫, 可使能以下兩個選項, 使ALSA 相容 OSS 架構. 可參考音效範例程式alsa_audio. (源碼位於BSP/applications/demos/alsa_audio 目錄下)

```
Device Drivers  --->
  <*> Sound card support  --->
        <*>   Advanced Linux Sound Architecture  --->
              <*>   OSS Mixer API
              <*>   OSS PCM (digital audio) API
```

- Cryptographic Accelerator設置

  要支持Cryptographic Accelerator, 請先勾選打開 Netowrking support裡的PF_KEY socket.

```
[*]  Networking support  --->
  Networking options --->
        <*> PF_KEY sockets
```

然後再打開 Cryptographic API 裡面的相關設定

```
Cryptographic API  --->
  <*>  Userspace cryptographic algorithm configuration
  [*]  Disable run-time self ests
  <*>  Software async crypto daemon
  <*>  User-space interface for hash algorithm
  <*>  User-space interface for symmetric key cipher algorithms
  [*]  Hardware crypto devices --->
        <*> Support for NUC980 Cryptographic Accelerator
```

NUC980 Cryptographic Accelerator 硬件支持 AES, DES, 及 3-DES crypto 演算法, 及支持 SHA 及 HMAC hash 演算法. 可參考範例程式 crpyto. (源碼位於 BSP/applications/demos/crypto 目錄下)

NUC980 Cryptographic Accelerator 硬件還提供了個硬件亂數產生器 (PRNG; Pseudo Random Number Generator)。要打開亂數產生器，必須要先將上述的 crypto device 驅動選取，然後才會出現可供選擇的硬件亂數產生器驅動，如下:

```
Device Drivers  --->
  Character Devices --->
```

```
        <*> Hardware Random Number Generator Core support

              <*> Nuvoton HW Random Number Generator support
```

- DMA 設置

  NUC980 系列芯片支援DMA功能. 要在內核中使能 DMA, 必須進入”DMA Engine support”選單中, 開啟”NUC980 DMA support”
  如果要在內核中使用DMA, 可以參考 linux-3.10.x/drivers/dma/dmatest.c中的使用方法, 如果需要實際了解dmatest.c的運作流程, 可以開啟”DMA Test Client”. DMA test client將會被自行被帶起.
  如下圖：

```
Device Drivers  --->

  [*]DMA Engin support  --->

        <*>   NUC980 DMA support

        <*>   DMA Test client
```

- 使用者空間記憶體管理功能設置

  如需要在使用者空間獲取一塊記憶體的虛擬和實體位置，就可啟動這功能來達成這樣的目的。
  請啟動如下的配置：

```
Device Drivers  --->

  Character devices  --->

        [*] Support for /dev/nuc980-mem
```

  使用方法可參考2D 範例程式。

- Ethernet 網口設置

  NUC980 系列支持了兩個網口, 可分別開起或同時開啟 要支持網口, 除了網口驅動外, PHY驅動需要一併使能. NUC980 開發版上使用的是 ICPlus 的 PHY, 若是使用了不同的 PHY, 設置需做相應的修改. NUC980 可支援透過 Magic Packet 喚醒, 但須透過 ethtool 開啟此功能. Magic Packet 的格式依循 AMD 的 Magic Packet Technology 白皮書規範. ethtool 工具的使用請參考 4.1 節 ethtool 的說明.

```
Device Drivers  --->

  [*] Network device support  --->
```

```
        <*>Dummy net driver support
        [*]   Ethernet driver support  --->
              <*>     Nuvoton NUC980 Ethernet MAC 0
              <*>     Nuvoton NUC980 Ethernet MAC 1
        -*-   PHY Device support and infrastructure  --->
              <*>   Drivers for ICPlus PHYs
```

- Etimer 設置

  NUC980 內核執行時, 使用基本型時鐘來計時. 另外還提供了四通道的加強型時鐘可輸出 50% 佔空比的輸出, 或是支持捕獲的功能. 四個通道可以獨立控制功能. 以下是相關的設置畫面. 不須使用的功能選擇 “No output” 或是 “No input” 即可. 以下的範例 Etimer 通道 0 及 1 分別使用 PC.6, PC.8 當輸出. 而 通道 2 及通道 3 分別使用 PC.11 及 PC.13 當作捕獲管腳. 請使能”NUC980 Enhance Timer(ETIMER) wake-up support”來支持加強型時鐘喚醒系統的功能.

```
Device Drivers  --->
  Misc devices  --->
        <*> NUC980 Enhance Timer (ETIMER) support
        <*> NUC980 Enhance Timer (ETIMER) wake-up support
              NUC980 ETIMER channel 0 toggle output pin (Output to PC6)  --->
              NUC980 ETIMER channel 0 capture input pin (No input)  --->
              NUC980 ETIMER channel 1 toggle output pin (Output to PC8)  --->
              NUC980 ETIMER channel 1 capture input pin (No input)  --->
              NUC980 ETIMER channel 2 toggle output pin (No output)  --->
              NUC980 ETIMER channel 2 capture input pin (Input from PC11)  --
->
              NUC980 ETIMER channel 3 toggle output pin (No output)  --->
              NUC980 ETIMER channel 3 capture input pin (Input from PC13)  --
->
```

  應用程序可透過 ioctl() 來控制加強型時鐘的功能, 目前支持了喚醒系統, periodic, toggle out, trigger counting mode, 以及 free counting mode 等五個功能. 喚醒系統功能, 可以設定不同的叫醒時間來喚醒系統. 在捕獲模式下 (trigger counting mode), 所捕獲到

的值可經由 read() 讀回. 單位是us, 代表了兩次觸發條件的間隔. free counting mode會測量外部引腳的輸入頻率, 所捕獲到的值可經由 read() 讀回, 單位是Hz, 代表外部引腳的輸入頻率. 所有的功能可以參考 BSP 中所附的範例程序(源碼位於BSP/applications/demos/etimer目錄)來開發所需要的功能.

- 智能卡 (Smartcard) 設置

  NUC980 擁有兩個智能卡接口, 符合 ISO-7816 以及 EMV 2000 規範. 若是系統中有使用智能卡的需求, 可以參考以下的設定, 使能智能卡的驅動. 這份智能卡驅動支持了 T = 0 以及 T = 1 的智能卡. 兩個接口可以分別設置卡插拔偵測的准位, 以及上電時需要給高電位或是低電位. 這個設定跟外部線路以及使用的卡槽有關. 另外接口 0 可以選擇使用 Port A 或是 Port C 的管腳, 接口 1 可以選擇使用 Port C 或是 Port F 的管腳. 當打開 Perform EMV check 的選項時, 驅動會執行較為嚴格, 符合 EMV 規範的檢查. 有些智能卡可能會被判別為無法操作的卡.

```
Device Drivers  --->

  Misc devices  --->

        <*> NUC980 Smartcard Interface support

        [ ]   Perform EMV check

        [*]   NUC980 SC0 support

                NUC980 SC0 pin selection (Use port A)  --->

                NUC980 SC0 CD pin config (CD high as card insert)  --->

        [ ]     Inverse SC0 power pin level

        [*]   NUC980 SC1 support

                NUC980 SC0 pin selection (Use port C)  --->

                NUC980 SC1 CD pin config (CD high as card insert)  --->

        [ ]     Inverse SC1 power pin level
```

  應用程序可透過 ioctl() 來控制智能卡接口. 以下列出支持的命令以及功用. 實際使用可以參考BSP 中所附的範例程序(源碼位於BSP/applications/demos/sc目錄).
  SC_IOC_GETSTATUS: 判斷目前卡槽的狀態, 例如是否插入還是拔出.
  SC_IOC_ACTIVATE: 激活智能卡. 若是成功, ioctl() 會返回 ATR 的長度.
  SC_IOC_READATR: 讀取智能卡回覆的 ATR (Answer to reset)
  SC_IOC_DEACTIVATE: 關閉智能卡.
  SC_IOC_TRANSACT: 透過 sc_transact 這個結構體對智能卡下命令並讀取回應.
  請注意, 進入省電模式前, 智能卡會被關閉. 系統喚醒後, 若是應用程序要讀寫智能卡,

必須重新激活智能卡.

- GPIO 設置

NUC980系列芯片支援GPIO介面控制, 要讓內核支持 GPIO 的控制, 請使能 “NUC980 GPIO support” 以及 “/sys/class/gpio/…”.

```
Device Drivers  --->

  [*] GPIO Support  --->

        [*]   /sys/class/gpio/... (sysfs interface)

              <*>   NUC980 GPIO support

               <*>     NUC980 external I/O wake-up support
```

GPIO 驅動程序將NUC980系列芯片的 GPIO口, 從GPIOA~GPIOJ 每組IO都保留32個號碼, 所以GPIOA編號0x000~0x01F, GPIOB編號0x020~0x03F, GPIOC編號0x040~0x05F, GPIOD編號0x060~0x07F, GPIOE編號0x080~0x09F, GPIOF編號0x0A0~0x0BF, GPIOG編號 0x0C0~0x0DF, GPIOH 編號 0x0E0~0x0FF, GPIOI 編號 0x100~0x11F, GPIOJ 編號 0x120~0x13F.

用戶程序可透過 sysfs 來控制各 GPIO 口. /sys/class/gpio/…”是通用的GPIO操作接口, 可以透過下列方法來控制GPIO

- /sys/class/gpio/export :來告訴系統需要控制哪個GPIO
- /sys/class/gpio/unexport: 則可以取消哪個GPIO控制
- /sys/class/gpio/gpio0/direction : 針對GPIOA00控制 in 或 out
- /sys/class/gpio/gpio0/value : 針對GPIOA00控制輸出1 或 0

下面範例可將GPIOA0設定成輸出High：

```
$ echo 0 > /sys/class/gpio/export

$ echo out >/sys/class/gpio/gpio0/direction

$ echo 1 >/sys/class/gpio/gpio0/value
```

也可以參考範例程式中的 gpio_demo(源碼位於 BSP/applications/demos/gpio目錄下)

在其他的驅動程序中可以透過下步驟控制GPIO :

- 在驅動程序中加入#inlcude <linux/gpio.h>
- 依據 arch\arm\mach-nuc980\inlcude\mach\gpio.h 決定使用哪個GPIO
- 以NUC980_PC7為例子如下:

  設定輸入模式： gpio_direction_input(NUC980_PC7);
  設定輸出模式和輸出值：gpio_direction_output(NUC980_PC7,1);
  設定輸出high： gpio_set_value(NUC980_PC7, 1);

設定輸出low： gpio_set_value(NUC980_PC7, 0);
取值： gpio_get_value(NUC980_PC7);
確認GPIO是否正在使用: gpio_request(NUC980_PC7, "NUC980_PC7");
取得GPIO 中斷號碼 : gpio_to_irq(NUC980_PC7);

使用GPIO中斷範例 :

```
static irqreturn_t PC7IntHandler(int irq, void *dev_id)
{
printk(KERN_INFO "PC7IntHandler:irq=%d \n",irq);
    return IRQ_HANDLED;
}
int xxx_init(void)
{
int ret,irqno;
ret = gpio_request(NUC980_PC7, "NUC980_PC7");
if (ret) printk("NUC980_PC7 failed ret=%d\n",ret);
irqno=gpio_to_irq(NUC980_PC7);
request_irq(irqno,  PC7IntHandler,
                 IRQF_TRIGGER_RISING | IRQF_TRIGGER_FALLING,
                "NUC980_PC7",
                 NULL);
}
```

需要讓GPIO從power down模式中叫醒 , 可以開啟”NUC980 external I/O wake-up support”功能. 並且在linux-3.10.x/drivers/gpio/gpio-nuc980.c 中可以找到EINT0, EINT1, EINT2, EINT3, EINT4, EINT5, EINT6, EINT7的設定. 以EINT0為例子 如下:

```
/*
 * @brief       External Interrupt 0 Handler
 * @details     This function will be used by EINT0,
 *              when enable IRQ_EXT0_H0 or IRQ_EXT0_F11 in eint0
```

```
 */

/*

static irqreturn_t nuc980_eint0_interrupt(int irq, void *dev_id){

        printk("@0\n");

        return IRQ_HANDLED;

}

*/



/* If enable IRQ_EXT0_H0 or IRQ_EXT0_F11 , linux will enable EINT0

 * User can modify trigger tiypes as below :

 * IRQF_TRIGGER_FALLING / IRQF_TRIGGER_RISING / IRQF_TRIGGER_HIGH /
IRQF_TRIGGER_LOW

 */

struct nuc980_eint_pins eint0[]={

//{IRQ_EXT0_H0, nuc980_eint0_interrupt,IRQF_TRIGGER_FALLING |
IRQF_TRIGGER_RISING,"eint0"},

//{IRQ_EXT0_F11,nuc980_eint0_interrupt,IRQF_TRIGGER_FALLING |
IRQF_TRIGGER_RISING,"eint0"},

{0,0,0,0}

};
```

由上面可以開啟 nuc980_eint0_interrupt 和設定 nuc980_eint_pins eint0 從那一個GPIO pin 和觸發的型態,假設需要使用EINT0中斷從 PH0 , 觸發的型態設定成 上升和下降觸發, 可以參考如下面設定即可.

```
static irqreturn_t nuc980_eint0_interrupt(int irq, void *dev_id){

        printk("@0\n");

        return IRQ_HANDLED;

}

struct nuc980_eint_pins eint0[]={

{IRQ_EXT0_H0, nuc980_eint0_interrupt,IRQF_TRIGGER_FALLING |
IRQF_TRIGGER_RISING,"eint0"},
```

```
{0,0,0,0}
};
```

- GPIO 模似I²C 接口設置
  除了使用硬體I²C, 尚可使用IO模擬的方式來實現I²C控制, 透過修改 arch/arm/mach-nuc980/dev.c 中i2c_gpio_adapter_data結構來選擇使用的腳位. 如設定 .sda_pin = NUC980_PG1, .scl_pin = NUC980_PG0 即是使用PG0為SCL腳, PG1為SDA腳.980980980

```
Device Drivers  --->
<*> I2C support  --->
I2C Hardware Bus support  --->
<* > GPIO-based bitbanging I2C
```

- I²C 接口設置
  I²C 接口的設置如下:

```
Device Drivers  --->
  <*> I2C support  --->
        I2C Hardware Bus support  --->
              <*> NUC980 I2C Driver for Port 0
              NUC980 I2C0 pin selection (SDA:PA_0  SCL:PA_1)  --->
              <*> NUC980 I2C Driver for Port 1
              NUC980 I2C1 pin selection (SDA:PA_13  SCL:PA_14)  --->
              <*> NUC980 I2C Driver for Port 2
              NUC980 I2C2 pin selection (SDA:PB_7  SCL:PB_5)  --->
              <*> NUC980 I2C Driver for Port 3
              NUC980 I2C3 pin selection (SDA:PB_1  SCL:PB_3)  --->
```

I²C port 1 有多組的管腳可選擇使用, 如 Port-A, Port-B, ….

選擇硬體I²C, 系統將使用NUC980內建I²C硬體模組, 進行I²C控制.

內建I²C port 0 client 是 NAU8822, 使用者可自行修改/新增或修改至port 1, 修改檔案位

置為：arch/arm/mach-nuc980/dev.c 中的nuc980_i2c_clients0 結構.

使用者也可依照以下的選項來設置I²C的slave eeprom mode.

```
Device Drivers  --->
  <*> I2C support  --->
        [*] I2C slave support
        <*> I2C eeprom slave driver
```

- MTD NAND flash 設置

  NAND flash 是使用的驅動是掛在 MTD 子系統之下. 可按照以下設置使能.

```
Device Drivers  --->
  Generic Driver Options  --->
        <*> Nuvoton NUC980 FMI function selection
              Select FMI device to support (Support MTD NAND Flash)  --->
  -*- Memory Technology Device (MTD) support  --->
        <*>   Command line partition table parsing
        <*>   Caching block device access to MTD devices
        -*- NAND Device Support  --->
              -*-   Nuvoton NUC980 MTD NAND
```

  驅動中的基本設置如果需要由U-boot環境變數傳入就必須將”Command line partition table parsing”選上, 否則會使用驅動程式裡的默認配置, 主要會將 MTD 分為三塊空間. 上電進入 shell 後, 分別是/dev/mtdblock0, /dev/mtdblock1, 以及 /dev/mtdblock2. 第一塊是放置 U-Boot 的空間, 第二塊放置內核文件, 第三塊則是用來掛載 YAFFS2 或 UBIFS 文件系統的空間. 若是配置有需要更改, 例如增加或減少分區, 改變分區大小. 請直接編輯 uboot/include/nuc980_evb.h 或 drivers/mtd/nand/nuc980_nand.c.980

- PWM 設置

  要支持 PWM, 請依以下的設定使能 PWM, 使用的管腳可能須依系統改變. 不須使用的通道選擇 “No output” 即可.

```
Device Drivers  --->
  [*] Pulse-Width Modulation (PWM) Support  --->
        <*>   NUC980 PWM support
```

```
            NUC980 PWM channel 0 output pin (Output from PB2)  --->
            NUC980 PWM channel 1 output pin (Output from PB3)  --->
            NUC980 PWM channel 2 output pin (No output)  --->
            NUC980 PWM channel 3 output pin (No output)  --->
```

要透過sysfs控制 PWM, 首先在系統開機後,進入 /sys/class/pwm/, 裡面可看到四個 PWM (pwmchip0~3), 每一組代表著一個 PWM通道. 要使用前, 先進入要控制的通道目錄執行 echo 0 > export, 此時會多生成一個 pwm0 目錄. 接著就可以開始控制這個通道了. 在新生成的目錄中有幾個文件可用來控制PWM, 它們的意義列在下表.

文件名稱	目的
period	控制週期, 單位是 ns. 目前驅動支持的最小單位是 us, 以打出 20us 之週期波為例, 使用 echo 20000 > period 命令
duty_cycle	設置佔空比. 單位是 ns. 目前驅動支持的最小單位是 us, 以打出 15us 之佔空比為例, 使用 echo 15000 > period 命令
polarity	設置輸出相位. 可支持 echo normal > polarity 正向輸出, 或是 echo normal > polarity 反向輸出
enable	使能及失能控制. 要使能, 使用 echo 1 > enable 命令, 要使能則使用 echo 0 > enable 命令

以下就是一個開啟 PWM0, 並輸出週期 300us, 佔空比 33% 的例子.

```
$ cd sys/class/pwm
$ ls
pwmchip0  pwmchip1  pwmchip2  pwmchip3
$ cd pwmchip0
$980 ls
device     export     npwm       power      subsystem  uevent     unexport
980$ echo 0 > export
980$ ls
device     npwm       pwm0       uevent
export     power      subsystem  unexport
980$ cd pwm0/
980$ ls
```

```
duty_cycle  enable      period      polarity    power       uevent
980$ echo 1 > enable
980$ echo 300000 > period
980$ echo 100000 > duty_cycle980
```

- Realtek RTL8188 802.11 WiFi 支持

  要支持RTL8188 USB WiFi 功能的話, 除了內核本身無線網絡功能要使能, USB host 驅動要使能外, 以及掛載模塊支持要使能外, 還需依照WiFi無線網路設置開啟相關設定:

  RTL8188 的源碼並沒有包含在 BSP 中. 若是需要源碼, 請跟模組廠索取, 新唐科技無法提供. 這個驅動的使用說明如下.

  1. 上電進入shell後, 使用 insmod 命令加載模塊.

```
$ insmod rtl8188eu.ko
```

  2. 啟動無線網口

```
$ ifconfig lo up
$ ifconfig wlan0 up
```

  3. 使用 wpa_supplicant 連接無線基地台

```
$ ./wpa_supplicant -Dwext -i wlan0 -c <config file> -B
```

  其中需指定基地台的連線設定檔, 以下是基地台設置成 WPA AES 連線時的範例設置:

```
network={
   ssid="TESTTEST"
   proto=WPA
   key_mgmt=WPA-PSK
   pairwise=CCMP
   psk="ZZZZZZZZ"
}
```

- Realtek RTL8192 802.11 WiFi 支持

  要支持RTL8192 SDIO WiFi 功能的話, 除了內核本身無線網絡功能要使能, SDIO host 驅動要使能外, 以及掛載模塊支持要使能外, 還需開啟以下的功能:

```
[*] Networking support --->

-*- Wireless --->

  <*>  Nuvoton external WiFi driver support
```

RTL8192 的源碼並沒有包含在 BSP 中. 若是需要源碼, 請跟模組廠索取, 新唐科技無法提供.

這個驅動的使用說明如下.

4. 上電進入shell後, 使用 insmod 命令加載模塊.

```
$ insmod 8192es.ko
```

5. 啟動無線網口

```
$ ifconfig wlan0 up
```

6. 使用 wpa_supplicant 連接無線基地台

```
$ ./wpa_supplicant -Dwext -i wlan0 -c <config file> -B
```

其中需指定基地台的連線設定檔, 以下是基地台設置成 WPA AES 連線時的範例設置:

```
network={

  ssid="TESTTEST"

  proto=WPA

  key_mgmt=WPA-PSK

  pairwise=CCMP

  psk="ZZZZZZZZ"

}
```

- RS232, RS485, IrDA 串口設置

  NUC980 系列帶有11個串口, 可以分別獨立設置. 請依以下的說明來使能串口功能. 每個串口可以單獨的開關. 除 UART0, UART3, UART5 外, 有多組管腳可選擇, 需要一併設置. 其中, UART0 這組串口是保留給 console 使用的. 不須特別使能.

  除UART0, UART3, UART5, UART7, UART9外,使用者可以開關喚醒功能.

```
Device Drivers  --->

  Character devices  --->

        Serial drivers  --->

              [*]  NUC980 serial support
```

```
[*]    NUC980 UART1 support
[*]      Enable UART1 CTS wake-up function
         NUC980 UART1 pin selection (Tx:PE2, Rx:PE3)  --->
[*]    NUC980 UART2 support
[*]      Enable UART2 CTS wake-up function
         NUC980 UART2 pin selection (Tx:PF11, Rx:PF12)  --->
[*]    NUC980 UART3 support
[*]    NUC980 UART4 support
[*]      Enable UART4 CTS wake-up function
         NUC980 UART4 pin selection (Tx:PC10, Rx:PC11)  --->
[*]    NUC980 UART5 support
[*]    NUC980 UART6 support
[*]      Enable UART6 CTS wake-up function
         NUC980 UART6 pin selection (Tx:PB2, Rx:PB3)  --->
[*]    NUC980 UART7 support
         NUC980 UART7 pin selection (Tx:PG4, Rx:PG5)  --->
[*]    NUC980 UART8 support
[*]      Enable UART8 CTS wake-up function
         NUC980 UART8 pin selection (Tx:PE10, Rx:PE11)  --->
[*]    NUC980 UART9 suppor
         NUC980 UART9 pin selection (Tx:PH2, Rx:PH3)  --->
[*]    NUC980 UART10 support
[*]      Enable UART10 CTS wake-up function
         NUC980 UART10 pin selection (Tx:PB10, Rx:PB11)  --->
[*]     Console on NUC980 serial port
```

若是串口需要作為紅外控制使用, 則除了要使用的串口需要使能外, 紅外模塊的驅動也需依以下的範例始能.

```
[*] Networking support  --->
        <*>   IrDA (infrared) subsystem support  --->
              Infrared-port device drivers  --->
                    <*> NUC980 SIR on UART
```

- SD 卡設置
  以下是SD卡接口使能的設置. NUC980支持一個 SD 卡接口.

```
Device Drivers  --->
  <*>MMC/SD/SDIO card support  --->
        <*>   MMC block device driver
        <*>     Use bounce buffer for simple hosts
        <*>   Nuvoton NUC980 SD Card support
```

  上電進入 shell 後, 若是偵測到有卡插入, 會在 /dev 下出現一個 mmcblk0 裝置. 若卡上有分區, 會依分區方式另外出現 mmcblk0, mmcblk1… 等設備.

- SPI 接口設置
  NUC980 系列蕊片支持了三個 SPI 接口, 可以單獨使能或是同時使能. 以下是同時使能兩個接口的說明:

```
Device Drivers  --->
  [*] SPI support  --->
        <*>  Nuvoton NUC980 Series QSPI Port 0
              QSPI0 pin selection by transfer mode (Normal mode)  --->
              QSPI0 TX/RX by PDMA or not (Use PDMA)  --->
        < >     QSPI0 enable pin for the second chip select
        <*>   Nuvoton NUC980 Series SPI Port 0
              SPI0 IO port selection (Port D)  --->
              SPI0 TX/RX by PDMA or not (Use PDMA)  --->
        < >     SPI0 enable pin for the second chip select
```

  QSPI0可藉由選擇normal或quad模式來決定使用SPI的腳位數量，Normal為4個腳或者

Quad為 6個腳。也可以選擇是否開啟 PDMA。 另外有第二組片 選腳可供選擇。

SPI0 和 SPI1 則只能選擇 Normal mode. 可以選擇是否開啟 PDMA。使用者也可以選擇開啟第二組片 選腳。

如要使用SPI flash, 需要開啟MTD功能, 使能如下選項：

```
Device Drivers  --->
  <*> Memory Technology Device (MTD) support  --->
        <*>   Caching block device access to MTD devices
        Self-contained MTD device drivers  --->
              <*> Support most SPI Flash chips (AT26DF, M25P, W25X, ...)
```

另外, JFFS2檔案系統的功能也要一併選上, 如此才能正確使用SPI Flash裝置.
JFFS2 文件系統的設置方式請參考文件系統設置的章節.
Linux 內核需正確的判斷 SPI flash 才能對其讀寫. 要讓內核識別新的 SPI flash型號, 請修改/新增 drivers/mtd/devices/m25p80.c中的 m25p_ids結構

```
static const struct spi_device_id m25p_ids[] = {
/* Atmel -- some are (confusingly) marketed as "DataFlash" */
{ "at25fs010",  INFO(0x1f6601, 0, 32 * 1024,   4, SECT_4K) },
{ "at25fs040",  INFO(0x1f6604, 0, 64 * 1024,   8, SECT_4K) },
...
{ "en25qh16", INFO(0x1c7015, 0, 64 * 1024, 32, 0) },
...
{ "cat25128", CAT25_INFO(2048, 8, 64, 2) },
  { },
};
```

以及 arch/arm/mach-nuc980/dev.c 中的nuc980_spi_flash_data結構.
其中 .type的字串需與 m25p_ids結構中的其中一個字串相同, 不然無法比對到正確裝置.

```
static struct flash_platform_data nuc980_spi_flash_data = {
```

```
        .name = "m25p80",
        .parts =  nuc980_spi_flash_partitions,
        .nr_parts = ARRAY_SIZE(nuc980_spi_flash_partitions),
        .type = "en25qh16",
};
```

如欲修改 SPI Flash partition 數，則可修改 arch/arm/mach-nuc980/dev.c 中的 nuc980_spi_flash_partitions結構.

```
static struct mtd_partition nuc980_spi_flash_partitions[] = {
        {
                .name = "SPI flash",
                .size = 0x0200000,
                .offset = 0,
        },
};
```

- SPI slave 設置

  NUC980 系列蕊片支持了三個 SPI 接口, 這三個接口皆可當成 slave. 以下範例是使能 QSPI0 接口當 SPI slave:

```
Device Drivers  --->
  -> Misc devices
        <*> NUC980 QSPI0 slave mode support
              QSPI0 pin selection by transfer mode (Normal mode)  --->
        < > NUC980 SPI0 slave mode support
        < > NUC980 SPI1 slave mode support
```

三個 SPI slave 範例程式放在 drivers/misc 目錄下，檔名分別為 nuc980-qspi0-slave.c, nuc980-spi0-slave.c, 以及 nuc980-spi1-slave.c

下面是 QSPI0 slave 範例程式，當收到 master 送過來的 “0x9f” 命令，準備資料並回覆給 master

```
static int QSPI0_Slave_Thread_TXRX(struct nuc980_spi *hw)
{
        unsigned char rx;
        unsigned long flags;
        int i;

        while(1) {

                wait_event_interruptible(slave_done, (slave_done_state !=
0));
                rx = __raw_readl(hw->regs + REG_RX);
                //printk("Receive [0x%x] \n", rx);

                switch (rx) {
                        case 0x9f:
                                for (i = 0; i < QSPI0_SlaveDataLen; i++)
                                        QSPI0_SlaveData[i] = i;

                                nuc980_enable_txth_int(hw);
                                break;
                        default:
                                break;
                }

                InTransmitted = 0;
                slave_done_state = 0;
                nuc980_enable_rxth_int(hw);
        }
```

```
        return 0;
}
```

用戶可以根據自己的應用來修改這個 slave 範例程式。

- SPI NAND 設置

  NUC980 系列蕊片支持 SPI NAND. 以下是使能 SPI NAND 的說明:

  首先使能 SPI

```
Device Drivers  --->
  [*] SPI support  --->
        <*>  Nuvoton NUC980 Series QSPI Port 0
              QSPI0 pin selection (Normal mode)  --->
        <*>  QSPI0 enable pin for the second chip select
              Pin selection (Use SS1 (PD0))  --->
        <*>  Nuvoton NUC980 Series SPI Port 0
              SPI1 IO port selection (Port D)  --->
        < >   SPI1 enable pin for the second chip select
```

然後開啟MTD功能, 使能如下選項：

```
Device Drivers  --->
  <*> Memory Technology Device (MTD) support  --->
        <*>   Caching block device access to MTD devices
        Self-contained MTD device drivers  --->
              <*> Support most SPI Flash chips (AT26DF, M25P, W25X, ...)
        <*>   NAND Device Support --->
        <*>   SPI-NOR Device Support --->
```

並使能 SPI NAND 驅動：

```
Device Drivers  --->
  <*> Staging drivers  --->
```

```
        <*>   SPINAND Device Support
```

Linux 內核需正確的判斷 SPI NAND flash 才能對其讀寫. 要讓內核識別新的 SPI NAND flash型號, 請修改/新增 drivers/mtd/nand/nand_ids.c中的 nand_flash_ids 結構

```
struct nand_flash_dev nand_flash_ids[] = {

        /*

         * Some incompatible NAND chips share device ID's and so must be

         * listed by full ID. We list them first so that we can identify

         * the most specific match.

         */

        {"W25N01GV 1G 3.3V",

                { .id = {0xef, 0xaa} }, SZ_2K, 128, SZ_128K, 0, 2, 64,
NAND_ECC_INFO(1, SZ_512) },

        {"TC58NVG0S3E 1G 3.3V 8-bit",

                { .id = {0x98, 0xd1, 0x90, 0x15, 0x76, 0x14, 0x01, 0x00} },

                  SZ_2K, SZ_128, SZ_128K, 0, 8, 64, NAND_ECC_INFO(1, SZ_512),

                  2 },

...

{NULL}

};
```

以及 arch/arm/mach-nuc980/dev.c 中的nuc980_spi_flash_data結構.
其中 .name 的字串需與 SPI NAND 驅動中的名字(.name) 一致，否則內核無法啟動驅動.
例如: 美光 SPI NAND 的驅動名字叫做 mt29f

```
static struct spi_driver spinand_driver = {

        .driver = {

                .name           = "mt29f",

                .of_match_table = spinand_dt,

        },
```

```
        .probe          = spinand_probe,
        .remove         = spinand_remove,
};
```

arch/arm/mach-nuc980/dev.c 中的nuc980_qspi0_flash_data結構中的 .name 也必須取名為 mt29f.

```
static struct flash_platform_data nuc980_qspi0_flash_data = {
   .name = "mt29f",
   ……
};
```

同時並修改nuc980_qspi0_board_info 中的 .modalias 欄位 .

```
static struct spi_board_info nuc980_qspi0_board_info[] __initdata = {
#ifdef CONFIG_MTD_M25P80
        {
                .modalias = "mt29f",
                …….
},
#endif
};;
```

- USB Host 設置

  要支持 USB Host, 請先勾選打開 USB Host 端支持。 NUC980 USB Host包含EHCI(USB 2.0)及 OHCI(USB1.1)兩個USB Host控制器，必須同時打開。 以下所列之項目必須全部勾選：

```
Device Drivers  --->
  [*] USB support  --->
        <*>   Support for Host-side USB
        <*>     EHCI HCD (USB 2.0) support
        <*>     NUC980 EHCI (USB 2.0) support
```

```
        NUC980 USB Host port power pin select (No USBH_PPWRx and USBH_OVRCUR)
--->

        [*]     NUC980 turn off Usb Hots VBUS power while power down

        <*>     OHCI HCD support

        [*]     NUC980 OHCI (USB 1.1) support
```

其中在 “NUC980 EHCI (USB 2.0) support”底下，依據所使用的芯片型號, 選擇USB Host port power pin. 選單如下:

```
[ ] PE.14 and PE.15 for USBH_PPWR0/1, PH.1 for USBH_OVRCUR

[ ] PF.10, PH.1 for USBH_OVRCUR

[ ] No USBH_PPWRx, PH.1 for USBH_OVRCUR

[X] No USBH_PPWRx and USBH_OVRCUR
```

如果 USB Host 電路設計上使用了 Power Switch IC, 那麼可選擇USB Host port0 及 port1 分別由PE.14及 PE.15控制; 或是選擇統一由 PF.10控制。當選擇使用 USBH_PPWRx 的情況下，USBH_OVRCUR就會被用來當作過電流保護指示腳來使用。如果電路設計沒有採用 Power Switch IC，而是直接供電給 USB port，那麼 USBH_PPWRx 就可以挪作 GPIO 使用，可選擇第三個或第四個選項。如果過電流保護指示腳也不需要，那麼可以選擇第四個選項。

NUC980 USB Host 驅動，提供讓使用者選擇在 power down (休眠) 模式下，選擇是否關閉 VBUS 供電。

```
 [*]     NUC980 turn off Usb Hots VBUS power while power down
```

如果使用者不勾選這項設定，那麼在系統休眠的時候，NUC980 USB Host 僅會關閉 USB PHY 電源，而不關閉 VBUS 電源。由於 VBUS 仍然保持供電，所以 USB 裝置盡管處於 suspend (暫停) 狀態，它仍然可以獲得 NUC980 USB Host 的供電，以維持在不斷電狀態。
如果使用者勾選了這項設定，那麼在系統休眠的時候，NUC980 USB Host會同時關閉 USB PHY 及 VBUS 電源。對於完全由 USB bus 供電的 USB device 來說，這就像當於斷電了。當系統喚醒時，NUC980 USB Host 重新對 USB PHY 及 VBUS 供電，這相當於所有的 USB 裝置斷開之後的重新連接，因此，所有連接的 USB 裝置都需要重新枚舉及初始化。

- USB Host 與 USB Mass storage 裝置設置

除了NUC980 的 USB host 驅動需要設置, 另外還需選擇所要支持的裝置類別, 例如 Mass Storage. 若是要選擇 Mass Storage, 則需先開啟 SCSI 設備支持, 才會出現 Mass Storage 的選項

```
Device Drivers  --->
  SCSI device support  --->
        <*>   SCSI device support
              <*> legacy / proc/scsi/ support
              <*> SCSI disk support
              <*> SCSI media changer support
              [*] Asynchronous SCSI scanning
              [*] SCSI low-level drivers
  [*] USB support  --->
        <*>   USB Mass Storage Support
```

- USB Host 與 USB Video Class 裝置設置
  如果要支援 USB Video Class (UVC) 視訊裝置，必須打開下列的設置選項：

```
Device Drivers  --->
  Multimedia support  --->
        <*>  Cameras/video grabbers support
        <*>  Media Controller API
        <*>  V4L2 sub-device userspace API
          <*>  V4L2 int device
        <*>  Media USB Apapters  --->
              <*>  USB Video Class (UVC)
              [*]    UVC input events device support
        <*>  V4L platform device
  Graphics support  --->
        <*>  Support for frame buffer devices  --->
              [*]  NUC980 LCD framebuffer support
```

- USB Host 與 HID 裝置設置
  如欲使用USB HID (如USB mouse, keyboard …)等裝置，除了需要USB host的功能外，另外需開啟HID及input的功能，請參考以下的設置。

```
Device Drivers --->
  HID support  --->
        HID bus support  --->
              <*>   User-space I/O driver support for HID subsystem
              <*>   Generic HID driver
        USB HID support  --->
              <*> USB HID transport layer
  Input device support  --->
        <*>   Mouse interface
        [*]     Provide legacy /dev/psaux device
        <*>   Event interface
        [*]   Keyboards  --->
              <*>   AT keyboard
        [*]   Mice  --->
              <*>   PS/2 mouse
```

- USB Host 與 USB modem
  如欲使用USB modem等裝置，除了需要USB host的功能外，另外需開啟以下的設置。

```
Device Drivers →
   [*] USB Support →
         [*] USB Serial Converter support →
               [*] USB driver for GSM and CDMA modems
```

- USB Device設置 – Mass Storage Device

```
Device Drivers --->
  [*] USB support --->
```

```
        <*> USB Gadget Support --->
              USB Peripheral Controller --->
                    <*> NUC980 USB Device Controller
              <M> USB Gadget Driver
              <M> Mass Storage Gadget
```

編譯完內核後，會產生三個 module 文件， fs/configfs/configfs.ko, drivers/usb/gadget/libcomposite.ko, drivers/usb/gadget/g_mass_storage.ko, 需要將此三個文件複製到rootfs或是系統可以存取到的地方。
USB Mass Storage Gaget的使用方法如下(以mmcblk0p1 裝置為例)：

```
$ insmod configfs.ko
$ insmod libcomposite.ko
$ insmod g_mass_storage.ko file=/dev/mmcblk0p1 stall=0 removable=1
```

- USB Device設置 – Serial Device

```
Device Drivers --->
  [*] USB support --->
        <*> USB Gadget Support --->
              USB Peripheral Controller --->
                    <*> NUC980 USB Device Controller
              <M> USB Gadget Driver
              <M> Serial Gadget (with CDC ACM and CDC OBEX support)
```

編譯完內核後，會產生七個 module 文件， fs/configfs/configfs.ko, drivers/usb/gadget/libcomposite.ko, drivers/usb/gadget/g_serial.ko, drivers/usb/gadget/u_serial.ko, drivers/usb/gadget/usb_f_serial.ko, drivers/usb/gadget/usb_f_acm.ko, drivers/usb/gadget/usb_f_obex.ko, 需要將此七個文件複製到rootfs或是系統可以存取到的地方。
USB Serial Gaget的使用方法如下：

```
$ insmod configfs.ko
$ insmod libcomposite.ko
$ insmod u_serial.ko
```

```
$ insmod usb_f_acm.ko
$ insmod usb_f_serial.ko
$ insmod usb_f_obex.ko
$ insmod g_serial.ko
g_serial gadget: Gadget Serial v2.4
g_serial gadget: g_serial ready
```

- VCAP 影像擷取接口設置
  NUC980系列芯片支援Video-Capture介面，在內核中使能此功能，首先必須開啟“cameras/video grabbers support”，然後進入 “Encoders, decoders, sensors and other helper chips”選項中, 開啟 “NUC980 Video-in support”並且選擇攝影機的型號, 目前支援OV7725, OV5640, NT99050 以及 NT99141.

```
Device Drivers  --->
  <*> I2C support  --->
       I2C Hardware Bus support  --->
            <*> GPIO-based bitbanging I2C
  [*] Multimedia support  --->
       [*]   Camera/video grabbers support
       [*]   Media Controller API
       [*]   V4L2 sub-device userspace API
       [*]   V4L platform devices  --->
             Encoders, decoders, sensors and other helper chips  --->
                  <*> Nuvoton NUC980 Video-In Support
                   (3)   Max frame buffer
                   (24000000) Video frequency
                   Nuvoton NUC980 Image Sensor Selection (NT99141)  --->
```

Video-Capture介面必須使用I²C介面來控制攝影機, 所以在開啟Video-Capture介面之前必須設定先開啟I²C介面, I²C的部分請參考I²C章節.

目前Video-Capture驅動支援V4l2 API, 範例部分請參考範例程式中的cap2lcm. (BSP/applications/demos/cap 目錄) , 從cap2lcm 的源碼中可以看見API如下:
xioctl(fd, VIDIOC_S_FMT, &fmt) : 設定影像的長寬以及格式
xioctl(fd, VIDIOC_DQBUF, &buf) : 接收影像
xioctl(fd, VIDIOC_QBUF, &buf) : 將接收完成的影像釋放出來
xioctl(fd, VIDIOC_STREAMON, &type) : 啟動CAP
xioctl(fd, VIDIOC_STREAMOFF, &type) : 停止CAP

- Watchdog Timer 看門狗設置

  要支持看門狗, 請參考以下的設置. 目前默認的超時為 2.03 秒. 用戶程序可透過 ioctl() 下 WDIOC_SETTIMEOUT 命令更改超時時間. 驅動中支持三個不同週期, 輸入小於2的話, 是0.53 秒. 介於 2~8 之間的話是 2.03 秒, 而超過 8 的話是設置成 8 秒. 可參考wdt範例程式. (BSP/applications/demos/wdt目錄)
  如果要支持看門狗喚醒系統的功能，請使能 “NUC980 WDT wake-up support”

```
Device Drivers  --->

  [*] Watchdog Timer Support  --->

        <*>   Nuvoton NUC980 Watchdog Timer

        <*>     NUC980 WDT wake-up support
```

- Window Watchdog Timer 窗口看門狗設置

  要支持窗口看門狗, 請參考以下的設置.

```
Device Drivers  --->

  [*] Watchdog Timer Support  --->

        <*>   Nuvoton NUC980 Window Watchdog Timer
```

NUC980 窗口看門狗與看門狗主要有三個差異, 第一, 設置始能之後, 無法更改設置, 無法停止. 第二, 只能在特定的時間內讓窗口看門狗復位, 而不是如同看門狗只要還沒超時, 隨時可復位. 所以應用程序在使用時, 一定要先使用 ioctl() 的 WDIOC_GETTIMELEFT 命令取得可復位的時間. 只有當返回時間為 0, 才可使用 WDIOC_KEEPALIVE 命令窗口看門狗復位, 否則在不正確的時間下達 WDIOC_KEEPALIVE, 會馬上造成系統復位. 第三, 窗口看門狗的設計是 CPU 在掉電以及閒置模式時, 是不數的. 當系統不忙碌時, 而Linux 內核會在每次系統時鐘中斷之間, 讓 CPU 進入閒置模式. 所以窗口看門狗可復位的時間會依據系統的忙碌程度而有所改變.可參考wwdt範例程式. (BSP/applications/demos/wwdt目錄)

- RTC 設置
  使用者使用RTC功能時,還可以設置RTC的喚醒功能

```
Device Drivers --->
  [*] Real Time Clock --->
        <*> NUC980 RTC driver
        [*]   Enable RTC wake-up function
```

- CAN 設置
  NUC980 系列帶有2個CAN(Controller Area Network), 可以分別獨立設置. 請依以下的說明來使能CAN功能. 每個CAN可以單獨的開關. CAN0有多組管腳可選擇, 需要一併設置.
  使用者也可以設置CAN的喚醒功能

```
-*- Networking support  --->
  <*>   CAN bus subsystem support  --->
        --- CAN bus subsystem support
        <*>   CAN Gateway/Router (with netlink configuration)
              CAN Device Drivers  --->
                    <*> Platform CAN drivers with Netlink support
                    [*] CAN bit-timing calculation
                    <*> NUC980 CAN0/CAN1 devices  --->
                          --- NUC980 CAN0/CAN1 devices
                          [*]   NUC980 CAN0 support
                          [*]    Enable CAN0 wake-up function
                                 NUC980 CAN0 pin selection (Tx:PB11, Rx:PB10)
--->
                                      (X) Tx:PB11, Rx:PB10
                                      ( ) Tx:PH3, Rx:PH2
                                      ( ) Tx:PI4, Rx:PI32
                          [*]   NUC980 CAN1 support
```

```
                            [*]    Enable CAN1 wake-up function
                                   NUC980 CAN1 pin selection (Tx:PH15, Rx:PH14)
--->
                                        (X) Tx:PH15, Rx:PH14
```

可搭配CAN範例程式(BSP/applications/demos/CAN)來使用。

- IIO ADC 設置

  使用者可以透過 IIO的架構來使用普通模式的ADC，可參考以下的設置來開啟IIO功能。

```
Device Drivers  --->
  <*> Industrial I/O support  --->
  [*]   Enable buffer support within IIO
  [*]   Enable triggered sampling support
  Analog to digital converters  --->
    <*> Nuvoton NUC980 Normal ADC driver
        Reference voltage selection (Internal AVDD, 3.3V)  --->
```

其中，參考電壓有兩種可以選擇，分別是”Internal AVDD 3.3V” 及”VREF input” 。

應用層使用方式如下：

```
$ cat /sys/bus/iio/devices/iio:device0/in_voltageX_raw
```

其中，X為頻道(X=0~8)。

- SCUART 智能卡串口模式設置

  NUC980 系列帶有2個智能卡接口. 這兩個接口除了智能卡功能外, 也能拿來當成基本的串口使用. 當系統自帶的串口不敷使用時, 可再擴充出兩個串口. 在智能卡串口模式下, SC_CLK 會當成傳送端使用, 而SC_DAT 則是當成接收端使用..

```
Device Drivers --->
  Character devices --->
        Serial drivers --->
                [*] NUC980 Smartcard UART mode support
                [*] NUC980 SCUART0 support
```

```
                    NUC980 SCUART0 pin selection (Tx:PA5, Rx:PA4) --->
               [*] NUC980 SCUART1 support
                    NUC980 SCUART1 pin selection (Tx:PC7, Rx:PC8) --->
```

在智能卡串口模式下, 設備文件會是 /dev/ttySCU0 以及 /dev/ttySCU1. 基本的操作與源生串口相同. 但是限制比原生的串口多, 例如傳送以及接收各只有四級的 FIFO, 不支援流量控制, 也無法支援 RS485, IrDA 等傳輸模式. 當系統串口足夠使用時, 應優先考慮使用原生的串口.

- Loop back裝置設置

  Loop back裝置可讓系統讀寫一個文件就像讀寫block裝置一樣，這個文件可以是任何系統可以讀寫的文件系統，然後加載至指定的loop back裝置上即可使用。使能設置方式如下：

```
Device Drivers --->
  Block devices --->
        <*>   Loopback device support
```

使用方式如下：

1. 產生文件檔

```
$ dd if=/dev/zero bs=1M count=1 of=fat.img
```

2. 格式化文件檔 (以FAT文件系統為例)

```
$ busybox mkfs.vfat fat.img
```

3. 加載文件檔

```
$ mount -o loop fat.img /mnt/loop
```

3.3.4 文件系統設置

- FAT 文件系統設置

  FAT 是 SD 卡以及 U 盤上常見的文件格式. 可以照以下的設置使能.

```
File systems  --->
  DOS/FAT/NT Filesystems  --->
        <*> MSDOS fs support
        <*> VFAT (Windows-95) fs support
        (437) Default codepage for FAT
        (iso8859-1) Default iocharset for FAT
```

以 SD 卡的第一分區為例的話, 加載文件系統的命令是:

```
$ mount -t vfat /dev/mmcblk0p1 /mnt
```

- JFFS2 文件系統設置
  JFFS2是 NAND flash 上使用的文件系統之一. 可依照以下的設置使能.

```
File systems  --->
  [*] Miscellaneous filesystems  --->
        <*>   Journalling Flash File System v2 (JFFS2) support
        [*]   JFFS2 write-buffering support
```

- ROMFS 文件系統設置
  ROMFS 文件系統是根文件系統使用的格式之一, 可以照以下的設置使能.

```
File systems  --->
  [*] Miscellaneous filesystems  --->
        <*>   ROM file system support
              RomFS backing stores (Block device-backed ROM file system
support)  --->
```

- YAFFS2 文件系統設置
  YAFFS2 是 NAND flash 上使用的文件系統之一， 需先使能 MTD 的 “Caching block device access to MTD devices Device drivers” 方可勾選。 可以照以下的設置使能：

```
File systems  --->
  [*] Miscellaneous filesystems  --->
        <*>   yaffs2 file system support
        <*>   Autoselect yaffs2 format
        <*>   Enable yaffs2 xattr support
```

加載 YSFFS2 文件系統的命令是：

```
$ mount -t yaffs2 -o”inband-tags” /dev/mtdblock2 /flash
```

- exFAT 文件系統設置
  exFat為微軟開發出來新一代的文件系統，exFAT可補足FAT在單一文件大小及裝置總容量上的限制。可以依照以下的設置使能：

```
File systems  --->

  DOS/FAT/NT Filesystems  --->

        <*> exFAT fs support
```

以 SD 卡的第一分區為例的話，加載 exFAT 文件系統的命令是：

```
$ mount -t exfat /dev/mmcblk0p1 /mnt
```

- FUSE 文件系統設置及支援NTFS文件系統

  FUSE (Filesystem in Userspace)，是指完全在用戶態實現的檔案系統。使用者可以透過FUSE系統實現各種自訂的文件系統，較常見使用FUSE來實現用戶文件系統的有NTFS-3G、SSHFS等等。以下將介紹如何透過FUSE實現微軟的NTFS系統(NTFS-3G)。

  可以依照以下的設置使能FUSE文件系統：

  NTFS-3G 是一個由Tuxera公司開發並維護的開源項目，目的是為 Linux 提供 NTFS 分區的的驅動程式。能夠安全快速的對 Windows NT （包括 Windows 2000、Windows XP、Windows Server 2003 和 Windows Vista）的檔案系統進行讀寫。目前NTFS-3G的最新版本是 Tuxera 公司於 2014年2月23日發佈的 ntfs-3g_ntfsprogs-2014.2.15。

  下載位置為：http://www.tuxera.com/community/ntfs-3g-download/。

  下載完後依照ntfs-3g上的使用手冊進行編譯，完成後使用下面命令即可使用。

```
$ ./ntfs-3g /dev/mmcblk0p1 /mnt/mmc
```

- UBIFS文件系統設置

  UBIFS 是用於固態硬碟儲存裝置上，並與LogFS相互競爭，作為JFFS2的後繼檔案系統之一。UBIFS 在設計與效能上均較YAFFS2、JFFS2更適合 MLC NAND FLASH。例如：UBIFS 支援 write-back, 其寫入的資料會被 cache, 直到有必要寫入時才寫到 flash, 大大地降低分散小區塊數量並提高 I/O 效率。

```
Device Drivers  --->

   -*- Memory Technology Device (MTD) support  --->

   <*>   Enable UBI - Unsorted block images  --->

File systems  --->

  [*] Miscellaneous filesystems  --->

        <*>   UBIFS file system support

        [*]     Advanced compression options
```

```
        [*]       LZO compression support
```

3.3.5 加速使用 JFFS2 文件系統的 SPI 開機

當文件系統是 JFFS2 且存放在 SPI flash，加速開機第一步驟是打開 SPI Quad 模式。

- 內核設置打開 SPI Quad 模式

```
Device Drivers  --->

[*] SPI support  --->

<*>   Nuvoton NUC980 Series SPI Port 0
SPI0 pin selection by transfer mode (Quad mode)  --->
```

- 使用 mkfs.jffs2 工具時，將參數 page size 設定為 0x1000

```
mkfs.jffs2  -s 0x1000 -e 0x10000 -p 0x800000 -d  rootfs_jffs2/ -o  jffs2.img
```

有些 SPI flash 的 sector 大小是 4K，這個屬性會定義在 drivers/mtd/spi-nor/spi-nor.c 當中的 spi_nor_ids []
例如 華邦 W25Q128 就是 4K sectoer.

```
{ "w25q128", INFO(0xef4018, 0, 64 * 1024, 256, SECT_4K) },
```

但是 mkfs.jffs2 工具最小的 sector 大小是 8K. 因此, 必須將 spi_nor_ids [] 中的 “SECT_4K” 屬性拿掉。修改如下。

```
{ "w25q128", INFO(0xef4018, 0, 64 * 1024, 256, 0) },
```

- 使用 sumtool 工具，同時內核設置打開 JFFS2_SUMMARY。

```
$ sumtool -i jffs2.img -o jffs2_sumtool.img -e 0x10000
```

- 內核設置打開 JFFS2_SUMMARY

```
-> File systems

     -> Miscellaneous filesystems (MISC_FILESYSTEMS)

     -> Journalling Flash File System v2 (JFFS2) support (JFFS2_FS)

      [*]     JFFS2 summary support
```

3.3.6 使用FIQ

一般的中斷在某些時候有可能會被系統鎖住，進而影響此中斷的即時性。此時，就可使用此章節內所提的FIQ，來確保中斷的即時性。

- 內核設置

```
System Type  --->
  [*] Nuvoton NUC980 FIQ support
```

- 使用方式範例

初始化流程如下(以TIMER0為例)：
在驅動中，將init_FIQ(0); 先放入驅動初始化函式中。

```
static int __init xxx_init(void) {
    ...
    init_FIQ(0);
    ...
}
```

再加入下列代碼及插入呼叫use_fiq()於適當位置。(應取代一般IRQ操作代碼)

```
/*FIQ handler for the timer*/
void nuc980_timer0_interrupt(void) {
    // ... add some code here
    __raw_writel(0x1, REG_TMR_ISR(TIMER0)); /* clear timer0 interrupt flag */

}


static uint8_t fiqStack[1024];
extern unsigned char fiq_handler, fiq_handler_end;
static struct fiq_handler timer0_fiq = {
    .name = "timer0_fiq_handler"
};


void use_fiq(void) {
     int ret;
     struct pt_regs regs;
```

```
    ret = claim_fiq(&timer0_fiq);
    if (ret)
        return;
    set_fiq_handler(&fiq_handler, &fiq_handler_end - &fiq_handler);
    // set some registers use in FIQ handler
    regs.ARM_r8 = (long)nuc980_timer0_interrupt;
    regs.ARM_r10 = (long)REG_AIC_FIQNUM;
    regs.ARM_sp = (long)fiqStack + sizeof(fiqStack) - 4;
    set_fiq_regs(&regs);
    /* Enable the FIQ */
    __raw_writel(__raw_readl(REG_AIC_SRCCTL4) & ~0x00000007,
                 REG_AIC_SRCCTL4);
    enable_fiq(IRQ_TIMER0);
}
```

其中，regs.ARM_r8必須為fiq handler的位址及regs.ARM_r10必須為REG_AIC_FIQNUM的位置。

3.3.7 電源管理

Linux 內核有支持電源管理的功能, 讓系統進入掉電模式, 降低電力的消耗, 等到接收到喚醒信號, 再喚醒系統. 要支持電源管理功能, 請使能以下內核配置選項, 再重新編譯內核.

```
Power management options  --->
  [*] Suspend to RAM and standby
```

使用支持電源管理的內核時, 可以使用以下的命令讓系統進入掉電模式. 在這個模式中, 不必要的時鐘會被關閉, DDR 也會進入自刷新模式(self-refresh mode). 只有被使能地喚醒源能將系統自掉電模式中喚醒.

```
$ echo mem > /sys/power/state
```

請注意, 要降低系統功耗至最低, 進入掉電模式前, GPIO 的上拉/下拉也需要設置. 這部分與硬件設計相關. 要依所接的裝置決定. 請在 arch/arm/mach-nuc980/pm.c 中, nuc980_suspend_enter() 這隻函數的前頭加上相關的設定.

3.4 Linux 內核編譯

內核設置完成後, 在 linux-3.10.x 目錄下執行 “make” 命令, 即可編譯內核. 若是編譯過程沒有錯誤, 則產生的內核影像檔, 以及使用 zip 壓縮的內核影像檔會放置在源碼上一層的 image 目錄中. 若是系統中有安裝 mkimage 工具的話, 也可以使用 “make uImage”來生呈給 U-Boot 使用的映象檔. 若是系統中有安裝 dtc 工具的話, 也可以使用 “make dtbs”來生成設備樹使用的 dtb 檔.

```
$ make
......
  Kernel: arch/arm/boot/Image is ready
cp arch/arm/boot/Image   ../image/980image
updating: ../image/980image (deflated 31%)
  GZIP    arch/arm/boot/compressed/piggy.gzip
  CC      arch/arm/boot/compressed/misc.o
  AS      arch/arm/boot/compressed/piggy.gzip.o
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
$ ls ../image/
980image
```

4 Linux 使用者程序

4.1 範例程序

NUC980 BSP 提供了一些範例程式在 applications 中. 下表列出了這些範例程序, 以及其內容

目錄	描述
alsa-utils-1.0.23	ALSA 命令列工具.* 交叉編譯命令如下: $ ./configure -host=arm-linux -disable-nls --disable-xmlto PKG_CONFIG_LIBDIR=/usr/local/arm_linux_4.8/usr/lib -disable-alsamixer $ make 播放混音設定範例如下: ● Fiel $./amixer set PCM 85% ● $./amixer set Headphone 90% 錄音混音設定範例如下: 來源為Mic時： ● $./amixer set “Mic Bias” on ● $./amixer set “Input PGA” 100% ● $./axmier set ADC 90% 來源為Line In 時： ● $./axmier set "Right Input Mixer R2" on ● $./axmier set "Left Input Mixer L2" on ● $./axmier set "L2/R2 Boost" 100% ● $./axmier set ADC 90%

	播放的命令: ● $./aplay <檔案名稱> 如要播放 BSP 內之音效範例檔，可使用下列命令： $ cd usr $./aplay －c 2 －f S16_LE alsa/8k2ch.pcm 錄音的命令: ● $./arecord －d 10 －f S16_LE －c2 －r8000 －t wav －D plughw:0,0 <檔案路徑及名稱> 同時錄放的命令： ● $./arecord -f S16_LE -r 8000 -c 2 -D plughw:0,0	./aplay
benchmark/netperf-2.6.0	網絡效能測試工具. 交叉編譯命令如下: $./configure --host=arm-linux $ make
busybox-1.22.1/	Busybox 源碼. 交叉編譯命令如下: 1. $ make menuconfig 2. Select applets to be build 3. $ make
lighttpd-1.4.39	lighttpd 源碼. 交叉編譯命令如下: 如果使用 Toolchain gcc 4.8: $ ./configure --host arm-linux --build pentium-pc-linux --without-zlib --without-bzip2 --without-pcre --target arm-linux $ make

	$ sudo make install
	
	如果使用 Toolchain gcc 4.3:
	$ ./configure --host arm-linux --build pentium-pc-linux --without-zlib --without-bzip2 --without-pcre --disable-ipv6 --target arm-linux
	$ make
	$ sudo make install
	
	lighttpd 的 lib 和 sbin 目錄會安裝在 /usr/local 底下，使用時必須將他們拷貝到 root file system.
	config_html_sample/ 目錄下有 lighttpd configuration file (lighttpd.conf) 以及 homepage html (index.html) 範例. 請將 lighttpd.conf 放在 sbin/ 目錄下，並在 sbin/ 目錄下新增子目錄 www，將 index.html 放在 www/ 目錄下.
	另外，請新增以下兩個檔案，分別是 lighttpd 的錯誤以及存取紀錄.
	/var/log/lighttpd/error.log
	/var/log/lighttpd/access.log
	
	開啟 lighttpd 步驟如下:
	● $ ifconfig eth0 192.168.0.100
	● $ cd /usr/local/sbin
	● $ ./arm-linux-lighttpd start -f lighttpd.conf

demos/alsa_audio	音效範例程序 *
demos/cap	影像擷取範例程序 *

demos/can	CAN bus 範例程序 *
demos/crypto	加解密範例程序 *
demos/etimer	加強型時鐘範例程序 *
demos/gpio	GPIO 範例程序 *
demos/irda	紅外範例程序*
demos/lcm/	LCD 範例程序*
demos/sc	智能卡範例程序 *
demos/thread	Thread sample applications. *
demos/rtc	RTC 範例程序 *
demos/uart	UART 範例程序 *
demos/wdt	看門狗範例程序 *
demos/wwdt	窗口看門狗範例程序 *
demos/dma	DMA 範例程序 *
i2c-tools	i2c-tools工具 $ make
yaffs2utils	yaffs2命令工具 $ make

lzo-2.09	壓縮/解壓縮工具.
	交叉編譯命令如下:
	$ cd lizo-2.09
	$ ./configure --host=arm-linux --prefix=$PWD/../install
	$ make
	$ make install

libuuid-1.0.3	產生獨立的序號工具. 交叉編譯命令如下: $ cd libuuid-1.0.3 $ ./configure --host=arm-linux --prefix=$PWD/../install

	$ make $ make install

mtd-utils	mtd-utils源碼. 交叉編譯命令如下:
	需要使用到lzo-2.09.tar.gz套件和libuuid-1.0.3.tar.gz套件
	$ cd mtd-utils
	$ export CROSS=arm-linux-
	$ export WITHOUT_XATTR=1
	$ export DESTDR=$PWD/../install
	$ export LZOCPPFLAGS=-I/home/install/include
	$ export LZOLDFLAGS=-L/home/install/lib
	$ make
	$ make install

ethtool-4.6	ethtool 是一個標準的應用程序可以控制有線網卡的硬件及驅動. 例如控制是否系統要透過網口Magic Packet 喚醒. 要交叉編譯請使用以下命令: ./configure CC=arm-linux-gcc CFLAGS=-mach=armv5te --host=arm-linux;make 讓網口支援喚醒功能, 請使用 ./ethtool –s eth0 wol g 命令. 要關閉喚醒功能請使用 ./ethtool –s eth0 wol d 命令.

*. 若是內核沒有正確的設置驅動程序, 這些範例程序的執行結果會不正確.

4.2 交叉編譯

有時在一些專案中需要移植軟件到ARM 平台. 許多開源軟件都已支持交叉編譯, 此時只要依據這些軟件的說明文檔實行交叉編譯的設定即可.

但有時遇到不支援編譯的軟件時, 就需要手動來更改 Makefile. 一般來說更改過後的 Makefile 與原始版本會相近, 只需要做小幅度的更動即可支援交叉編譯. 以下列了需要修改的部分

- 編譯工具的前贅字需要設置. 例如原本的 Makefile 設定 gcc 為編譯工具, 則在新的 Makefile 需要改成交叉編譯使用的 arm-linux-gcc. 其他例如 as, ld 等工具也須分別更改為 arm-linux-as 及 arm-linux-ld
- 庫文件以及頭文件所在的路經必須作相對應修改. 交叉編譯不使用 x86 系統下的 glibc. 而是使用工具鏈中所提供佔系統資源較少的 uClibc.

以下是一個簡單的交叉編譯 Makefile 可供參考.

```
.SUFFIXES : .x .o .c .s


ROOT = /usr/local/arm_linux_4.8/usr
LIB =$(ROOT)/lib
INC :=$(ROOT)/include


CC=arm-linux-gcc -O2 -I$(INC)
WEC_LDFLAGS=-L$(LIB)
STRIP=arm-linux-strip


TARGET = hello


SRCS := hello.c


LIBS=  -lc -lgcc -lc
all:
      $(CC) $(WEC_LDFLAGS) $(SRCS) -o $(TARGET) $(LIBS)
      $(STRIP) $(TARGET)


clean:
      rm -f *.o
      rm -f $(TARGET)
      rm -f *.gdb
```

5 版本歷史

版本號	日期	描述
1.00	July 01, 2018	初版發布
1.01	Jan. 24, 2019	內容更新
1.02	Mar. 21, 2019	增加 USB modem 設定說明

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.