嵌入式即時作業系統於運動控制領域之運用

Implementation of an embedded real-time operating system on the Motion Control

工業技術研究院 機械所 沈万凱

摘要

現今一般運動控制模組多爲基於PC技術並依附其處理器性能之PC-Based架構控制器，然而在高階工具機不斷朝高速高精密度的規格需求靠攏下，即時性的問題也相對浮現出來，爲滿足高速高精度下即時處理事件發生和運動控制演算法的龐大計算量，因而發展出更高階的CPU-Based架構控制器，將處理器、作業系統、和運動軌跡之運算整合於單一運動控制卡上，藉由獨立CPU的運作模式與強即時性(hard real time)作業系統之配合，將可有效解決控制系統於PC-Base架構下所遭遇的即時性能之問題。

Abstract

The general motion control modules, up to date, are frequently used in the PC-Based system. Due to the problem of the real-time system which demands for high-speed and high precision, the PC-Based system is not able to immediately deal with the event and does not meet the requirements. Therefore, the hard real-time embedded operating system which works on the motion control card is developed to solve the real-time problem occurred on the PC-Based.

關鍵字

智慧型運動控制平台 IMP（Intelligent Motion control Platform）

嵌入式系統(Embedded System)

即時作業系統（Real-Time Operating System，RTOS）
硬即時（Hard Real Time）

前言

控制器若以架構來分，目前工業上常見之控制器約可概分爲三種類型：PLC-Based、PC-Based、以及CPU-Based控制器（stand alone型）[1]。PLC-Based與PC-Based控制器基本上皆需配合運動控制卡運作，目前國內工具機或產業機台

廠商多以應用PC-Based架構之控制器爲主，少數廠商使用PLC控制器。而PC-Based常面臨之即時性問題，從過去至今即有許多國內外廠商與研究單位投入研究和解決，例如採用IPC(Industrial PC)與DSP晶片整合爲雙CPU架構[2]的運作模式，以DSP負責運動控制法則之演算，而將控制器之人機介面，包含讀取輸入資料與顯示輸出資料，以及流程控制等工作交由IPC負責執行；或者在軟體上採用具即時處理性能之作業系統 (RTOS)：諸如VxWorks、RTX (Real-Time Extension)、QNX、 LynxOS 、RtLinux、Winows CE 等軟體平台，其目的無非皆欲強化控制器，使其獲得更高之精確性和可靠度。在雙CPU架構模式下，雖能有效解決即時性之問題，然花費於硬體上所需的資源亦相對較多，在成本同時提高的考量下，對所選用的DSP與PC硬體設施便將形成限制。

而即時性作業系統在PC-Based系統或CPU-Based架構上各有其應用領域，於PC-Based控制器應用上常聽到的RTX，便是一種執行於Windows底下的即時性作業軟體(Real Time Extension)，此外如著名的VxWorks則是可執行於CPU-Based架構上且性能卓越的嵌入式即時作業系統軟體。此類作業系統應用領域會因使用者所需環境而有所不同，工研院機械所機電控制整合部爲了因應計算量日趨龐大複雜的運動控制演算法與工具機產業升級下對高速高精度規格之要求，因而發展了一套屬於CPU-Based控制器架構的智慧型運動控制平台(Intelligent Motion Platform ,IMP)，內建中央處理器(Power PC 405)及一套即時性作業系統(VxWorks)，可有效解決即時性之問題。

嵌入式系統

嵌入式系統(Embedded System)是軟體與硬體的綜合體，可涵蓋機械或其它的附屬裝置的整個綜合體設計之目的，來滿足某種特殊功能，所以可以瞭解嵌入式系統似乎強調著“特定功能”的原則，因此它有可能爲整個硬體設備之一部份或是一台機器，也有可能是軟體型態或是韌體形式；有的以明顯型態存在且易於辨識，而有的嵌入式系統則隱藏於機器內部中，不易從外觀辨識出。

嵌入式系統的多元化讓我們很難辨識出其型態，但是一般的嵌入式系統有幾個共同的特徵，有別於一般的桌上型系統，其特徵如下：

1. 用來執行特定功能：

通常一個嵌入式系統只反覆地執行一個特定程式，不像桌上型系統可以執行各式各樣的程式，如文書處理、電玩程式，當然還有其它新加入的程式。當然嵌

入式系統也有例外，像是有更新軟體的能力，就像我們的手機可以用韌體更新的方式來升級。另一種狀況是有好幾個程式被輪流載入系統內，例如有些飛彈在飛行時執行一個程式，而在鎖定目標時又執行另一個程式。雖然如此，這些狀況依然說明了嵌入式系統只執行特定的功能。

2. 即時性能的要求：

很多嵌入式系統必須要對周遭環境變化做出感應，並且要即時得到結果，而後做出適當的反應輸出。

3. 系統的限制：

一般系統在設計尺度上都會有一些限制，然在嵌入式系統上這些限制會變的更嚴格。而常見的設計指標如下：

(1) 運算處理能力(Processing Power)：

完成工作所需要的運算處理能力之值。嵌入式系統只需滿足其應用需求，所以並不強調需要多快的速度、多大的暫存器寬度。所以一般常見的系統還是以8bit或16bit爲主，而32bit正在佔據主流地位。

(2) 記憶體(Memory)大小：

代表了存放執行碼與處理資料所需的記憶體。一般用於嵌入式系統的記憶體爲ROM、RAM 或快閃記憶體(Flash Memory)。由於記憶體容量有限，因此所撰寫的程式碼需要越精簡越好。

(3) 研發成本(Development Cost)：

硬體與軟體設計過程中所需要的成本。

(4) 電源管理(Power Management)：

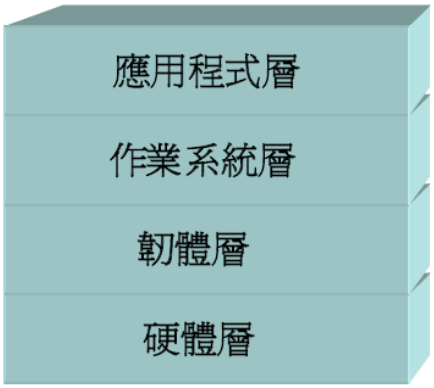
電源管理爲很重要的指標，因爲電源會直接影響系統的執行時間和系統所能消耗的功率。通常會利用軟體設計待機模式，使系統耗電量降至最低，硬體則以低耗能電子元件防止電能損耗。

(5) 可靠度(Reliability)：

因爲不同的應用對於其要求也不盡相同，有些應用容許錯誤發生，而有的則不允許出錯。

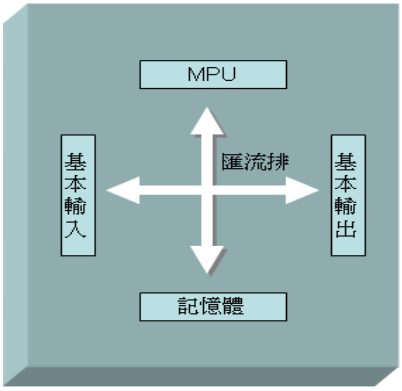
嵌入式系統構成概要

嵌入式系統架構區分成分成四層，分別爲硬體層、韌體層、作業系統層與應用程式層，如圖一



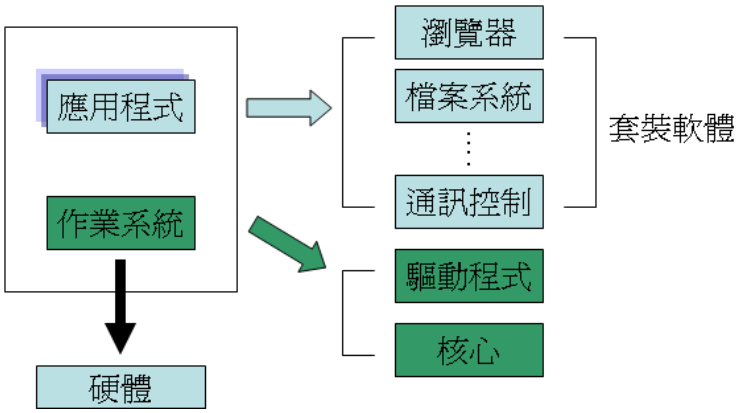
圖一 嵌入式系統架構圖

一般典型嵌入式系統的硬體架構有微處理器單元(Micro Processing Unit, MPU)、記憶體、輸入裝置、輸出裝置等硬體資源而程式與資料等則稱爲軟體資源
，而硬體結構的需求是要依照各自應用而進行變化，但基本的主要硬體構成如圖二所示。



圖二 嵌入式系統之基本硬體構成圖

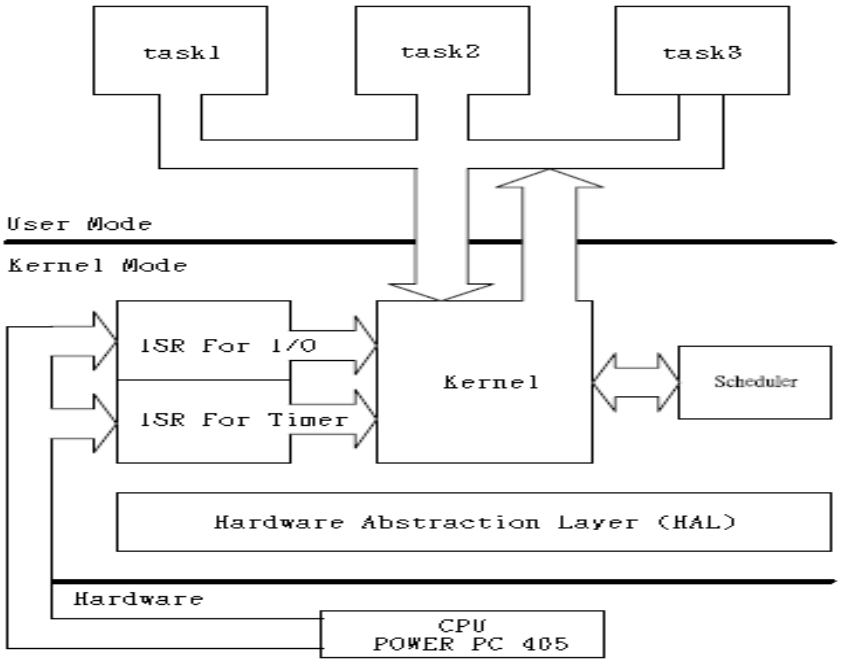
另外在軟體方面由圖三可知一般來說由核心、裝置驅動程式、套裝軟體構成。核心主要進行應用程式的執行、監視與控制，利用核心控制輸出入裝置提供應用程式執行輸入輸出之功能。所以若是要控制週邊硬體則必須透過OS核心和裝置驅動程式。



圖三 嵌入式系統軟體構成概要圖

嵌入式即時作業系統架構簡介

由前面介紹可知一個較複雜的嵌入式系統必須要有OS的支援，方可達成開發目的。而一般的嵌入式即時作業系統的架構主要又可以分爲核心模式(Kernel Mode) 、使用者模式(User Mode)、硬體抽象層(Hardware Abstraction Layer)這三個層面來討論如圖四所示，使用者模式(User Mode)主要描述程式設計者對程式上的應用設計。至於核心模式(Kernel Mode)，其基本成員有核心結構(Kernel Structure)、任務管理(Task Management)、時間管理(Time Management)、任務間的通信與同步(Interprocess Communication and Synchronization)、記憶體管理(Memory Management)、中斷管理(Interrupt Management)、檔案系統(File System)、等單元[3][4]，硬體抽象層(Hardware Abstraction Layer)是介於實際硬體裝置和作業系統核心之間的一個仲介角色，當要與硬體溝通，則需透過硬體抽象層來負責當之間的媒介，這樣處理的好處在於作業系統可以相容於不同的硬體平臺上，核心的程式碼不必因爲硬體的不同而需要修改，所以硬體抽象層可增進軟體的可移植性。



圖四 嵌入式即時作業系統架構圖

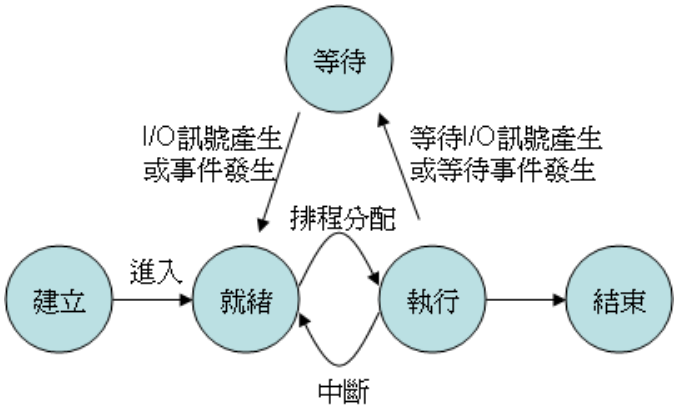
系統中程式設計者所設計的應用程式都是放在使用者模式，然在使用者模式中每個應用程式都視爲一個任務。每個任務的執行過程、順序都依據嵌入式作業系統的設計依序實現、執行。程式設計者在使用者模式設計的應用程式若要與核心模式做溝通一般可以藉由系統呼叫(System Call)的方式，「系統呼叫」是讓使用者與系統核心（kernel）直接溝通的介面之一。說得更清楚些，所有用戶產生的任務（Task）都要透過系統呼叫(System Call)才能完成類似檔案的存取、處理程序間的通訊(interprocess communication)、記憶體管理、硬體的控制等較低階(low level）的工作。

所謂即時作業系統（Real-Time Operating System，RTOS），是指作業系統本身要能在一個固定時限內對程式呼叫做出正確的反應，亦即對於時序與穩定度的要求是十分嚴格的，即時系統還有分硬即時（Hard Real Time）和軟即時（Soft Real Time）[5]，其中的差異，就是硬即時裡的所有工作都不能夠延遲，而軟即時裡可以允許少量的工作延遲。

即時作業系統不外乎有任務(Task)、號誌(Semaphores)、排程(Scheduler)、信號(Signals)、計数器(Timer)、記憶體管理(Memory Management)、網路通訊(Network Communication)、輸出入系統(I/O System)、中斷(Interrupts)、檔案系統(File System)等[3][4]，這些都是目前在即時作業系統上常看到的一些基本功能。

▮任務(Task) ：

所指的是正在執行的程式，在作業系統中的工作是按照順序來逐一進行的。換句話說，即在任一個時間點內只執行該任務某部份的指令，一般在作業系統中的任務會被區分成成建立、就緒、執行、等待、結束等五個狀態[6]，圖五爲任務狀態流程示意圖。



圖五 任務狀態流程示意圖

當使用者建立一個任務後，即被系統設成就緒狀態進入佇列中等待系統排程(Scheduler)分配執行。當執行完後若是處於等待的狀態(等待I/O 訊號或是事件發生)，將會被放入等待的佇列中；或是當有發生中斷時，則有新的任務會再度的被置入就緒狀態重新由系統排程執行。若是處於等待的任務，直到I/O訊號產生或是事件發生後，才會被置入就緒佇列中繼續等待系統排程的動作，這樣週而復始的執行，直到使用者將之移除，或是符合某些條件而停止執行。此時，任務會被擺至結束狀態佇列中，最後作業系統再將之停止與移除排程。

▮排程(Scheduler)：

排程(Scheduler)功用就是說當任務(Task)要取得CPU的控制權是由排程來決定的，排程就像佇列一樣，由佇列首先出現的任務取得CPU的控制權，一段時間後被中斷，任務被推入佇列尾端，而新的佇列再出現下一個任務使用CPU，如此反覆過程使系統達到分時多工的目的。

▮ **號誌**(Semaphore)：

號誌最早是由Dijkstra 在1960 年代所提出的觀念[6]，主要是用來解決系統任務同步之問題，當發生兩個以上的任務要同時使用一個資源時，必須透過號誌的保護來完整的使用該資源。在一般的作業系統的號誌可分爲二元號誌(Binary Semaphore)及計數號誌(Counting Semaphore)。

(1)二元號誌(Binary Semaphore)

一個二元號誌的數值只能爲0或1，當值爲1時，代表目前無任何工作取得該資源，若是有工作取得號誌資源時，則會將值設成0以表示目前的該資源正被使用中。

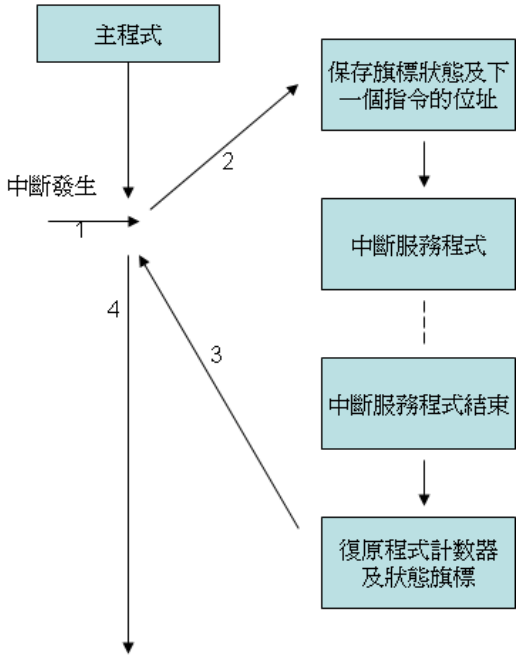
(2)計數號誌(Counting Semaphore)

有一個計數旗標，其值可以是任意一個正負號的整數值，利用計數旗標可以記錄目前有多少個工作正在等待使用該資源。

▮中斷(Interrupts)：

當CPU依某程式進行指令執行時，若有一個信號使的CPU暫停目前的程式進行，而去做另一項特定的工作，此事件的發生就稱爲中斷。中斷的種類大體來說，不外乎硬體中斷、軟體中斷這兩類。硬體中斷的形成，通常是外界的硬體裝置利用由CPU 拉出的中斷要求信號線來通知CPU中斷的請求。而軟體中斷，通常是CPU自己引發的，比如說執行了不該執行的指令、計算錯誤或者是執行某個用來產生軟體中斷的指令。

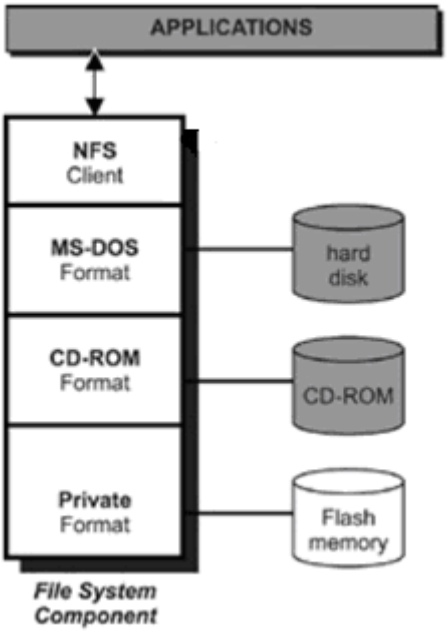
對於處理中斷的一般原則是將目前執行CPU狀態紀錄下來，然後跳到中斷處理程式做進一步處理；中斷處理完成後，回復中斷發生前的狀態，然後繼續正常的程式執行其步驟如圖六。每個中斷都有中斷的優先順序，當有兩個（含以上）中斷同時發生的時後，優先權越高則越先執行。不論是硬體中斷或軟體中斷系統內都有設計其對映的中斷服務常式(ISR, Interrupt Service Routine)執行中斷後要處理的事件。



圖六 中斷步驟

▮檔案系統(File System)：

一個儲存裝置的管理系統，作業系統可以透過檔案系統對儲存的裝置做讀寫的動作，這一些的儲存裝置有CD-ROM, Floppy Disk, Hard Disk, 或 Flash memory 裝置如圖七，像是CD-ROM透過標準的 ISO 9660檔案系統作管理，或是像Floppy Disk還是Hard Disk透過MS-DOS FAT檔案系統作管理[4]，



圖七 檔案系統

■ 網路通訊(Network Communication)

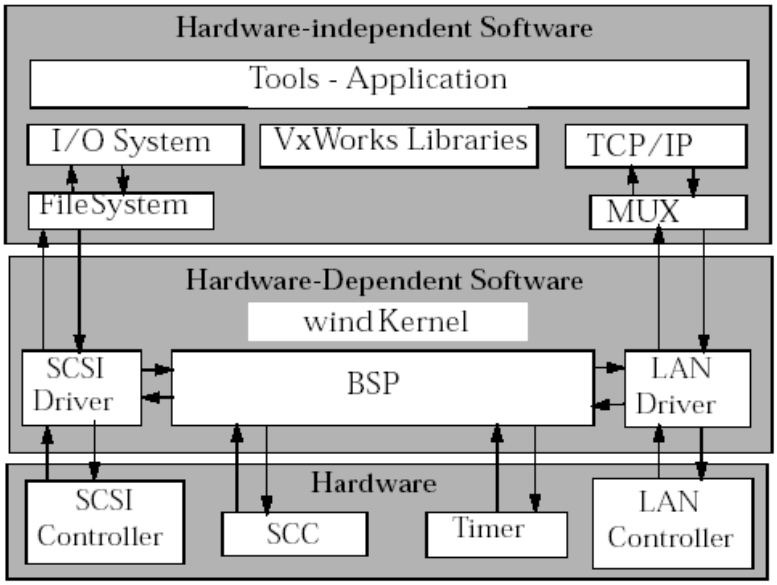
通訊協定簡單的定義就是電腦與電腦間，爲了透過網路來做通訊時所相互約定的相關事項，符合這樣規範的電腦在網際網路中才能夠正確的進行電腦間通訊，一般常見的IP、ICMP(Internet Control Message Protocol)、TCP、UDP、Telnet、FTP、HTTP…等，這些的通訊協定在目前的作業系統大都有支援。

■ 輸出入系統(I/O System)

作業系統的目的之一就是要讓使用者不需要瞭解硬體設備複雜細微的操作方式所以透過了作業系統對硬體做有效的管理,因此一般在標準device driver基本都是通過I/O系統來存取的，透過輸出入系統這樣做的好處是可以遮罩底層硬體，對上層應用程式提供統一的介面。

運動控制與作業系統應用

工研院機械所爲了整合之前在PC-Base上的運動控制到IMP上運作，選用了一套即時的作業系統VxWorks作爲核心的管理，並將所有的MCCL運動控制函式庫移植到IMP裡做運動控制運算，有效的減輕之前在PC端處理器大量運算的負擔，因此對於PC端的部分只做簡單運動界面控制，和負責下達使用者命令的人機介面，至於VxWorks其主要的架構如圖八



圖八 VxWorks架構圖

一般 VxWorks 作業系統的基本構成部分主要有以下五個部分[7][8]：

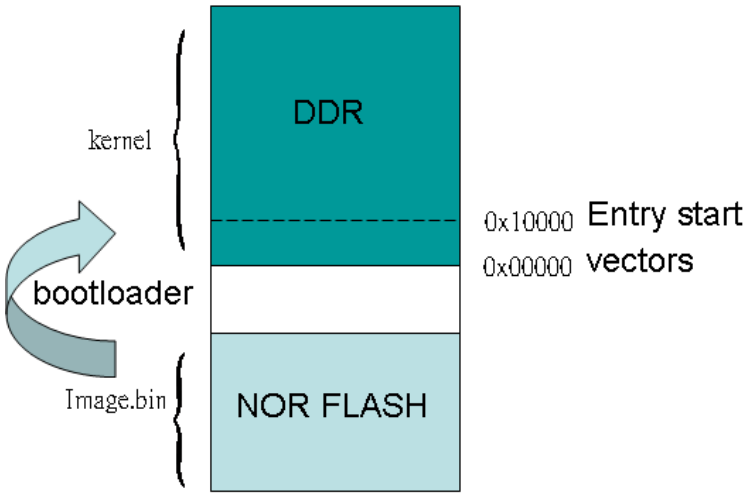
- BSP(Board Support Package)
- 微核心(wind micro kernel)
- 網路系統
- 檔案系統
- I/O 系統

其中又以BSP在我們的開發中占極爲重要的部份，該術語通常用於嵌入式領域，主要指在開發嵌入式應用時，系統開發商提供的各種驅動程式函式庫，方便程式開發人員進行移植，BSP是介於硬體和作業系統之間的一層，應該說是屬於作業系統的一部分，主要目的是爲了支援作業系統，使之能夠更好的運行於硬體板子上。BSP相對於作業系統而言，不同的作業系統對應於不同定義形式的BSP，例如VxWorks的BSP和Linux的BSP相對於某一CPU來說儘管實現的功能一樣，可是寫法和介面定義是完全不同的，所以寫BSP一定要按照該系統BSP的定義形式來寫（BSP的編程過程大多數是在某一個成型的BSP範本上進行修改）。這樣才能與上層作業系統保持正確的介面和良好的支援。

VxWork 的 wind 核心主要有任務(Task)管理、號誌(Semaphore)服務、記憶體管理、中斷服務程式、時鐘管理、計時器服務、和高度的可裁剪性核心模組可以讓使用者依所需要作設定與運用，以及相容即時系統標準 POSIX 協定，在網路方面 VxWorks 提供了強大的網路功能，能與其它許多主機系統進行網路通信。網路完全相容 4.3BSD，也相容 SUN 公司的 NFS。通過乙太網路採用 TCP/IP 和 UDP/IP 協定在不同主機之間傳送資料，其它還有像 DNS、DHCP、PPP、FTP、Telnet 等傳輸協定，在檔案系統 vxwork 則提供像是 MS-DOS-Compatible File System(DosFs)、Ram disk file system (RamFs)、CD-ROM File System(CdromFs)，

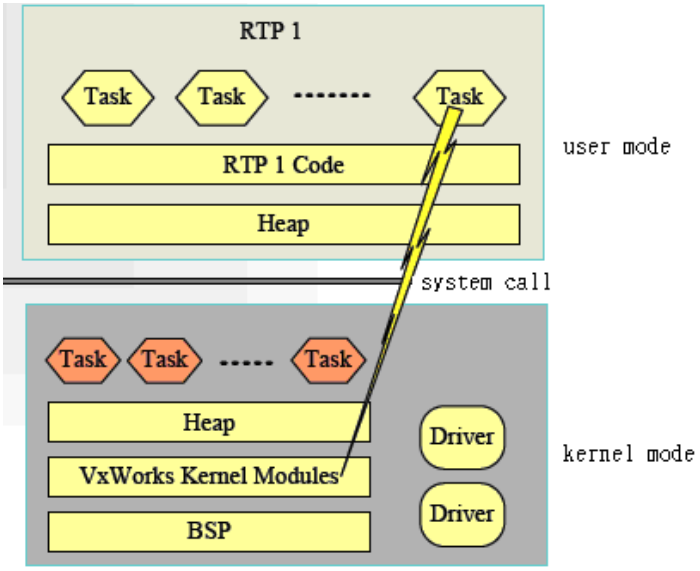
Ture Flash File System(TrueFFS)等。

爲了將整個作業系統與運動控制函式庫完全移植到IMP裡做運用，因此在IMP上必須要有ROM與RAM的記憶體，而所謂ROM記憶體指的就是Flash主要是用來儲存整個OS和運動控制函式庫的Image檔案，而RAM記憶體指的就是DDR記憶體，主要在開機後經由bootloader程式將原先存放在Flash記憶體裡的OS Image檔案搬移到DDR上做運算，至於bootloader簡單地說就是在作業系統核心運行之前的一段小程式。通過這段小程式，開機時會將作業系統載入記憶體，並執行作業系統起始程序的過程，而後再去執行第一個所要執行的Process，示意如圖九所示。



圖九 bootloadt示意圖

vxwoks是一個多工的Task系統，因此在作業系統環境中也將其運作的模式分成兩個模式，一個爲核心模式(Kernel Mode)、另一個則爲使用者模式(User Mode)，在開發的階段，可以利用vxworks提供的IDE介面，將開發的系統程式設計爲核心模式(Kernel Mode)和使用者模式(User Mode)兩種，系統設計者運行在核心模式(Kernel Mode)，使用者開發程式運行在使用者模式(user mode)底下，在vxworks中使用者模式其名稱又稱之爲Real Time Processes(RTP)[9] , 其主要功用是用戶端可以根據其所需要功能，動態的創建或刪除RTP，RTP下的任務(TASK)可以隨時動態載入運行，每個RTP任務完全獨立，程式在RTP任務內部出現的任何錯誤都被限制在RTP任務內部，刪除RTP即時任務時，自動釋放所有資源。從而使核心模式免受運行於RTP下用戶模式應用程式的影響。當使用者模式(User Mode)要與核心模式(Kernel Mode)做溝通時，同樣可透過system call的方式做溝通其示意如圖十，達成Kernel Mode與User Mode的溝通。



圖十　RTP的示意圖

系統中在核心的部份有一個最接近底層硬體的運動控制驅動程式,而這一個驅動程式主要負責控制如像ENCODER、DAC或ADC、DDA等一些I/O介面控制程式，另外像MCCL運動控制軌跡函式庫則規劃在RTP的模式下，因此當使用者下達一個MCCL運動軌跡運算時，作業系統接收到這個命令，此時作業系統便會規劃一個新的Task，運算使用者規劃的軌跡命令直到軌跡運算結束，然後等待下一次運動命令的下達，另外爲了方便使用者儲存開發完後的程式，而不用每次開機完成後還必須透過網路再傳輸一次程式，在IMP裡也規劃了一塊讓使用者透過作業系統提供的檔案系統(file system)作爲儲存，而這個檔案系統稱之爲TrueFFS(Ture Flash File System)，主要是對Flash記憶體做規劃與管理，當我們要對Flash記憶體作存取時只要透過TrueFFS便可以很簡單透過指令動作可以完成。

未來發展

目前工研院機械所對於IMP運動控制平台已提供了可供網路傳輸之介面，最終目標則預計在IMP運動控制平台上整合入web sever的服務功能，當使用者完成所開發規劃之運動軌跡控制程式後，只需透過網路，在遠端使用諸如網際網路瀏覽器一類的連線軟體，即可對IMP進行人機介面之操作，進而達到所需之控制功能。這樣的好處是當有多部機台分散各處時，或許只要一台電腦即可從遠端隨時掌握每一部控制機台之資訊。

參考資料

[1]林家慶 “CPU-Based運動控制技術簡介”機械工業雜誌，2008.10月

[2] Jianhua Bai, Luyong Sun, Jianfeng Pan, Yaodong Xu, Ji Zheng ” Research and Development of Embedded Numerical Control System Based on Digital Signal Processor” IEEE Aug. 2006

[3] Jean J. Labrosse, “MicroC/OS-II The Real-Time Kernel”, R&D Books 1998, November.

[4] Qing Li and Carolyn Yao “Real-Time Concepts for Embedded Systems” CMP Books

[5] 胡繼陽, 李維仁, 柯力群, 張志龍，嵌入式系統導論，學貫行銷股份有限公司，台北市，2004。

[6] **駱詩軒、鄧俊彥編譯**，“**作業系統概念**”，**東華書局股份有限公司**，2000。

[7] 林嘉樹，蔣鈴鴿 “VxWorks操作系统BSP和BootLoader介绍”， 維普資訊 2005 年第 7 期

[8]WindRiver “Kernel Programmer’s Guide VxWorks 6.3”

[9]WindRiver “VxWorks APPLICATION PROGRAMMER’S GUIDE 6.3”