國 立 交 通 大 學

電機資訊學院 資訊學程

碩 士 論 文

跨平台瀏覽器在嵌入式環境的研究

A FML Browser for the Stack-based Embedded Systems

研 究 生：楊惠親
指導教授：張瑞川　教授

中 華 民 國 九 十 三 年 七 月

跨平台瀏覽器在嵌入式環境的研究
A FML Browser for the Stack-based Embedded Systems

研 究 生：楊惠親　　Student : Hui-Chin Yang

指導教授：張瑞川　　Advisor : Ruei-Chuan Chang

國 立 交 通 大 學

電機資訊學院 資訊學程

碩 士 論 文

A Thesis

Submitted to Degree Program of Electrical Engineering

Computer Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Computer Science

July 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年七月

跨平台瀏覽器在嵌入式環境的研究

研究生：楊惠親　　指導教授:張瑞川 教授

國立交通大學電機資訊學院 資訊學程(研究所)碩士班

摘　　要

近年來重要業界技術，如 Java VM、微軟的.NET CLR 都是堆疊式虛擬機器(stack-based VM)，它們的指令集(instruction set)都是採用堆疊運算的方式：執行時的資料都是先放入堆疊再行運算。因堆疊架構具有架構簡潔、程式碼短小、容易達到跨平台需求等優點。

本論文提出的 FML 構想，是一個結合 Forth(Stack-based Language)與 HTML 特色的 Forth Markup Language，以簡易的語法與彈性的擴充功能，以建立一個以堆疊架構為處理器核心的嵌入式系統環境，藉由實作完成新的應用程式開發平台-FML Browser。

A FML Browser for the Stack-based Embedded Systems

Student：Hui-Chin Yang Advisors：Prof. Ruei-Chuan Chang

Degree Program of Electrical Engineering Computer Science

National Chiao Tung University

ABSTRACT

Important commercial technology in recent years has included the Java VM and Microsoft's .NET CLR, which are stack-based VMs. Their instruction sets use a stack computation method: at execution time, parameters are first placed on the stack and computations are then performed. This stack-based architecture has advantages such as concision, small code, and ease of portability across platforms.

The Forth Markup Language (FML) construct discussed in this thesis is a language that combines distinctive features of HTML and the stack-based language Forth. FML's simple syntax and flexible extensibility were used to establish an embedded system environment with a processing kernel using a stack-based architecture and to perfect a new application programming development platform: A FML Browser.

目 錄

圖目錄

第一章 緒論

1.1 研究動機

在 1960 年代，記憶體容量有限，價格不菲，以 stack model 為執行基礎的指令集，可產生較佳的 code density 節省記憶空間，當時的 B5000,B6500 即是堆疊架構的機器(Barton[1961],Hauck and Dent [1968]).但因受限於堆疊頂端的執行效率(Amdahl,Blaauw,and Brooks [1964],Bell et al. [1970])，在 1970 後期，除了 Intel 80x86 的浮點架構外，堆疊機幾乎在市場上消失無蹤。直到 1990s 網際網路興起，JAVA Virtual Machine、.NET CLR 皆因使用堆疊式指令集產生跨平台的優勢而盛行。由於網路間檔案的傳輸日益頻繁、嵌入式系統之應用發展蓬勃；在這些資源受限的環境下，以堆疊架構的精簡程式碼，不啻為節省傳輸時間及降低耗用資源的另類考量或思維。

為研究 stack-based 架構在嵌入式系統開發環境中尚可發揮的特性或優勢，本論文透過 Stack-based 的 Forth 語言，從嵌入式視窗元件設計、跨平台瀏覽器的移植、到 FML(Forth Markup Language)在 Forth 嵌入式環境的應用，評估此開發環境的可行性，希望在瞭解更多元的嵌入式系統核心技術與特色後，能否因此針對不同需求的嵌入式應用，增加更自主且具彈性的選擇方案，以因應未來嵌入式系統環境之演變與其應用需求驟增之可能。

1.2 現況與問題

1.2.1 現況

近十年來由於全球各地興起的資訊熱潮，肇致電腦運用的普及，與應用技術的蓬勃發展。這巨大的變革，衝擊且改變了世人既有的生活方式，更創造了新一波的文明。運用在客戶端設備上的嵌入式系統之技術，若具備輕巧便利及特有功能的特性，則可讓客戶端設備擁有更多的競爭

利基。因此，在 1980 年代即以架構精簡、節省資源又可輕易開發特殊功能，其系統穩定性亦足以擔任在美國太空總署的特殊應用的 Stack-based 架構之 Forth 嵌入式技術，是否可運用在資源有限的客戶端設備環境下，完成其精簡及特有功能之應用，以減少不必要之架構的硬體成本，若此核心技術能被成熟地開發應用，則可提供更多元的選擇與競爭利基。

現今的資訊環境中，除了 JVM、.NET CLR 的堆疊式指令集外，目前技術成熟且應用廣泛的的嵌入式系統構件設計中，從嵌入式處理器、嵌入式作業系統、到開發嵌入式應用程式的語言，幾乎很少討論到 Stack-based 架構的技術。一個應用系統，通常需要許多構件緊密地整合才能運作完備，因此，大部份的嵌入式應用產品，幾乎都採取整合現成可得的構件，例如以合適的處理器核心、嵌入式即時作業系統、現有可用的 IP，加上新功能的特殊需求設計，即可迅速地達到新功能的要求。若要以 Stack-based 架構為處理器的核心，則也必須有相搭配的合適構件，才能彰顯其效率與效益。

1.2.2 問題

由於堆疊機架構過於簡單，且執行時效率受限於資料堆疊頂端的限制，因此，以堆疊架構為主的處理器核心技術，在高處理效率的環境中，極少受到重視。堆疊架構的嵌入式系統，在目前業界雖有少數單位仍然採用運用於美國太空總署的太空船計劃的堆疊架構處理器核心，但除了以精簡的雙堆疊 Forth 虛擬機作為簡單即時作業系統核心外，在應用程式的開發則以 Forth 為高階語言進行應用程式開發。與資源豐沛的 RISC 架構的嵌入式系統相較之下，堆疊架構的嵌入式系統技術在學術界及業界幾乎沒有任何相關的研究與支援。雖然堆疊架構具備精簡或客制化地擴充彈性與通透地整合性，若欲在此環境下開發嵌入式應用程式，會受限於簡陋的開發環境，欲迅速完成高產能的應用程式開發，實屬相當艱辛。有鑑於此，在此堆疊式嵌入式系統環境中，提供一套精簡的視窗元件使用者圖形界面工具組，將可節省開發的視窗應用程式的工作，以提

昇應用程式設計師的工作效率，使其輕易地開發具使用者親和力的界面與更具功能核心的應用。

1.3 設計構想

1.3.1 建立堆疊架構的嵌入式系統的環境

以運行為 NASA 計畫的 EP21，加以改良為 32bit 的 EP32 為此嵌入式系統的處理器核心，以運行於 EP32 和 PC 端的 Forth VM -- FSharp 為作業系統核心。

1.3.2 設計應用程式開發平台

設計一個堆疊架構指令集的應用程式語言與軟體開發平台，以提供有效率地開發具生產力及親和力的嵌入式系統應用程式。

堆疊架構嵌入式系統環境中，可用的軟體，僅具有簡單的基本功能，沒有完善的開發環境，或提供更進階的功能以快速產生具使用者親和力的應用程式，針對此問題，綜合目前應用程式發展環境，分析各架構的使用狀況，提出一個可符合未來需求的堆疊架構嵌入式系統的開發環境，以 Forth 語言撰寫出 GUI 工具組及開發應用程式的描述式語言--Forth Markup Language (FML)，在 PC 端運用此語言與發展平台，第一步先產生跨平台(WINCE、MS-Windows、PalmOS、Macintosh)電子書瀏覽器的應用，再進一步移植此平台到 EP32，做為嵌入式應用程式開發環境。

有了跨平台 FML Browser 提供設計好的視窗元件，再加上語法類似 HTML 簡單易學的 FML 語言，則在此堆疊式的嵌入式系統中開發具親和力的視窗應用程式將更容易，也更具效率了。

1.4 研究目的

(1) 在嵌入式客戶端設備上：

- 提高客戶端設備之瀏覽器功能，使客戶端不需仰賴伺服端的功能，其與伺服端達成平權式功能。

- 不需要靠外掛，即具執行處理圖形、程式邏輯等能力。

(2) 在堆疊架構的嵌入式系統開發環境上：

- 提供一個堆疊架構的嵌入式應用程式開發平台，以提昇開發視窗應用程式之設計效率，更容易設計出具使用者親和力的嵌入式應用程式。
- 提供一個簡單易學的描述語言，開發嵌入式應用程式，可自行增加延伸擴充功能之彈性。

(3) 在嵌入式系統開發環境上：

- 針對不同需求的嵌入式應用，提供一個精簡且節省成本的嵌入式硬體環境，及有效率具彈性的嵌入式系統開發的核心技術。增加更自主且具彈性的選擇方案，以因應未來嵌入式系統環境之演變與其應用需求驟增之可能。

1.5 論文架構

第一章描述研究動機、現況與問題、設計概念、研究目的；第二章為研究環境建立，介紹堆疊架構處理器 EP32 之特性、Forth VM 操作系統流程與架構；第三章描述為什麼要提出 FML Browser-應用程式開發平台，將從現行環境的問題分析並提出 FML 構想；第四章為 FML 平台的實現、FML 應用程式語法與範例說明、FML BROWSER 移植到 EP32，與 FML 應用程式在 EP32 上的執行結果。第五章針對實作與移植做評估與檢討。第六章是本研究之結論與未來工作。

第二章 研究環境建立

2.1 Stack-based Architecture Processor

2.1.1 EP32 簡介

EP32 前身是 MuP21 是二十位元的處理器，由創造 Forth 語言的摩爾先生所設計，它的指令集很簡單，只有二十五個指令，可以用五個位元來界定。這 25 個指令是摩爾先生是依硬體設計的方便而決定的，並不考慮其後續軟體發展的需求。由於 Forth 系統在電腦的軟硬體上的一貫性，MuP21 指令集與 Forth 基本指令集相互比較，大部分的指令是相同的。

丁陳漢蓀博士把 Forth 系統移植到 MuP21，這樣發展出了 P16，P24，EP32 和 EP64 等系統。P24 系統並得到加州理工學院的太空輻射實驗室採用，做為太陽能輻射研究衛星上，電腦裝置的核心架構。台灣易符公司接著發展出支援更多指令的系統，作為 Forth 更高階的應用平臺。EP32 即為易符公司發展之 32 位元嵌入式處理器核心。

2.1.2 EP32 的架構

EP32 的架構如圖 1,大致可以分為 4 部份，各部份的組織及功能如下：

(1) 資料處理部份在上圖右下方。中央資料暫存器 T，為算術邏輯運算的核心，所有的資料處理都經過此中心暫存器；S 輔助資料暫存器，T 暫存器中資料暫存處；參數堆疊 T 和 S 暫存器中資料堆置暫存處；ALU 算術邏輯運算器，取參數堆疊 T 及 S 暫存器的資料，經過運算後結果存回 T。

(2) 程式地址儲存部份。R 返回地址暫存器，為 P 暫存器中地址暫存處；返回堆疊，為 R 暫存器中的地址堆置暫存處。

(3) 地址產生部份。P 暫存器，為指令地址暫存器；X 暫存器，為資料地址暫存器；AMUX 地址多工器，選擇 P 或 X 暫存器中的地址，送至地址排線。

(4) 有限狀態控制機部份。I 指令暫存器，儲存目前執行的指令字元；C 指令計數器，控制有限狀態的順序；IMUX 指令多工器，選擇指令字元中的機器碼；DECODER 指令解碼器，產生所有的控制訊號，執行機器碼。

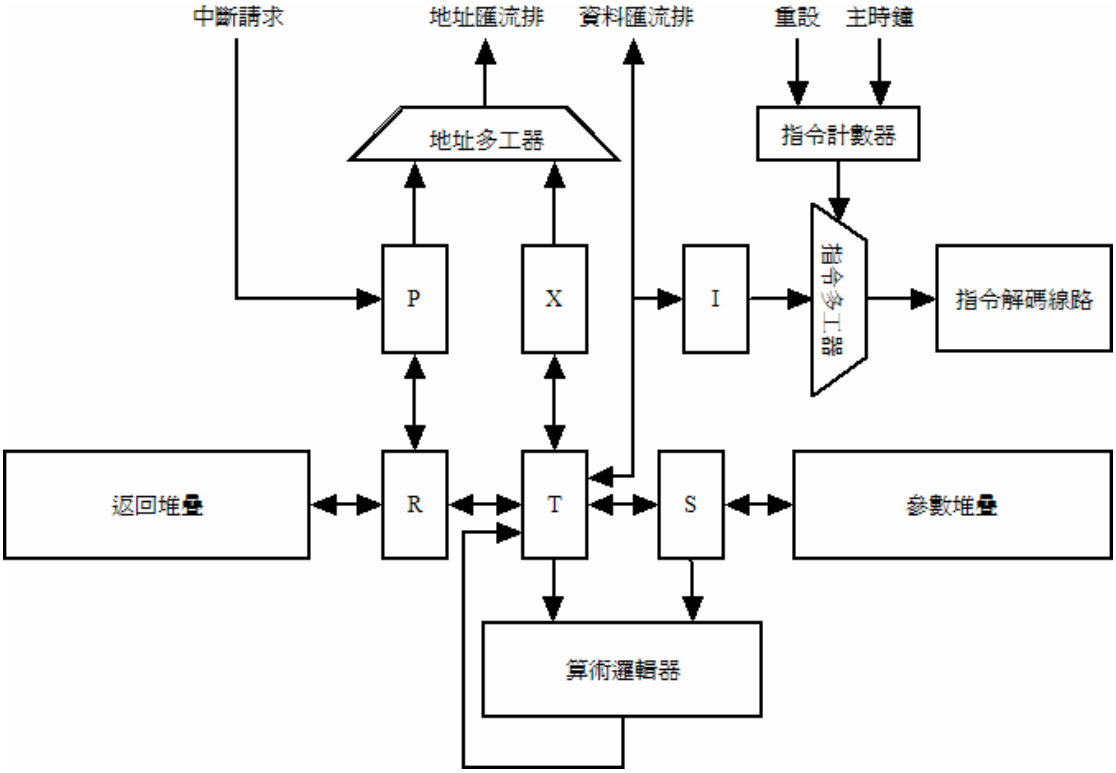

圖 1. EP32 架構圖

2.1.3 EP32 的架構和設計

EP32 能執行以下一組精簡但功能全備的指令，指令有兩種格式，控制程式流程的長指令有這樣的格式：

00	Cccccc	aaaaaa	aaaaaa	aaaaaa	aaaaaa

其他的短指令的格式是：

00	Cccccc	cccccc	cccccc	cccccc	cccccc

每個指令用六位元(cccccc)，在 32 位元的指令字元中可以容納 5 個

短指令，或是一個長指令。在長指令中有 24 位元的地址欄(aaaaaa)。指令字元最高的兩位元現在並不使用，留給有經驗的 IC 設計師擴展系統的硬體功能。

以下是各種 EP 系列 CPU 共用的指令集，其中短指令的名稱，代碼和功能如下表所列：

算術運算指令	COM，SHL，SHR，RR8，AND，XOR，ADD，MUL，
堆疊和暫存器指令	PUSHR，POPR，TX，XT，PUSHS，POPS
記憶體存取指令	LDX，STX，LDXP，STXP，LDI，LDC，STC
其他指令	RET，NOP

長指令含有一組 24 位元的地址，若要跳到新的地址，將此 24 位元取代 P 暫存器中右邊的 24 位元，即是新地址。長指令的名稱，代碼和功能如下表所列：

指 令	代 碼	功 能
BRA	000000	跳到指令中所指的地址。
BZ	000010	若 T＝0，跳到指令中指定的地址。
BC	000011	若 Carry 在上次的算數運算中設為 1， 跳到指令中地址。
CALL	000100	呼叫副程式。將 P 中的地址壓入 R 堆疊，並跳到指定的地址。
NEXT	000101	若 R 不是 0，將 R 減 1 并跳到指令中指定的地址。

2.2 Forth Systems

2.2.1 Forth Systems 特色

Forth 系統的特色，是將一部電腦的必要功能，選出最核心的 31 個基本指令。對於任何一個 CPU，只要重寫 31 個 Forth 基本指令，整個 Forth

系統就可以在這一個 CPU 上活起來。因此，Forth 容易應用在精簡的嵌入式系統環境中。

2.2.2 Interpreter and Compiler 共同運行之機制與特色

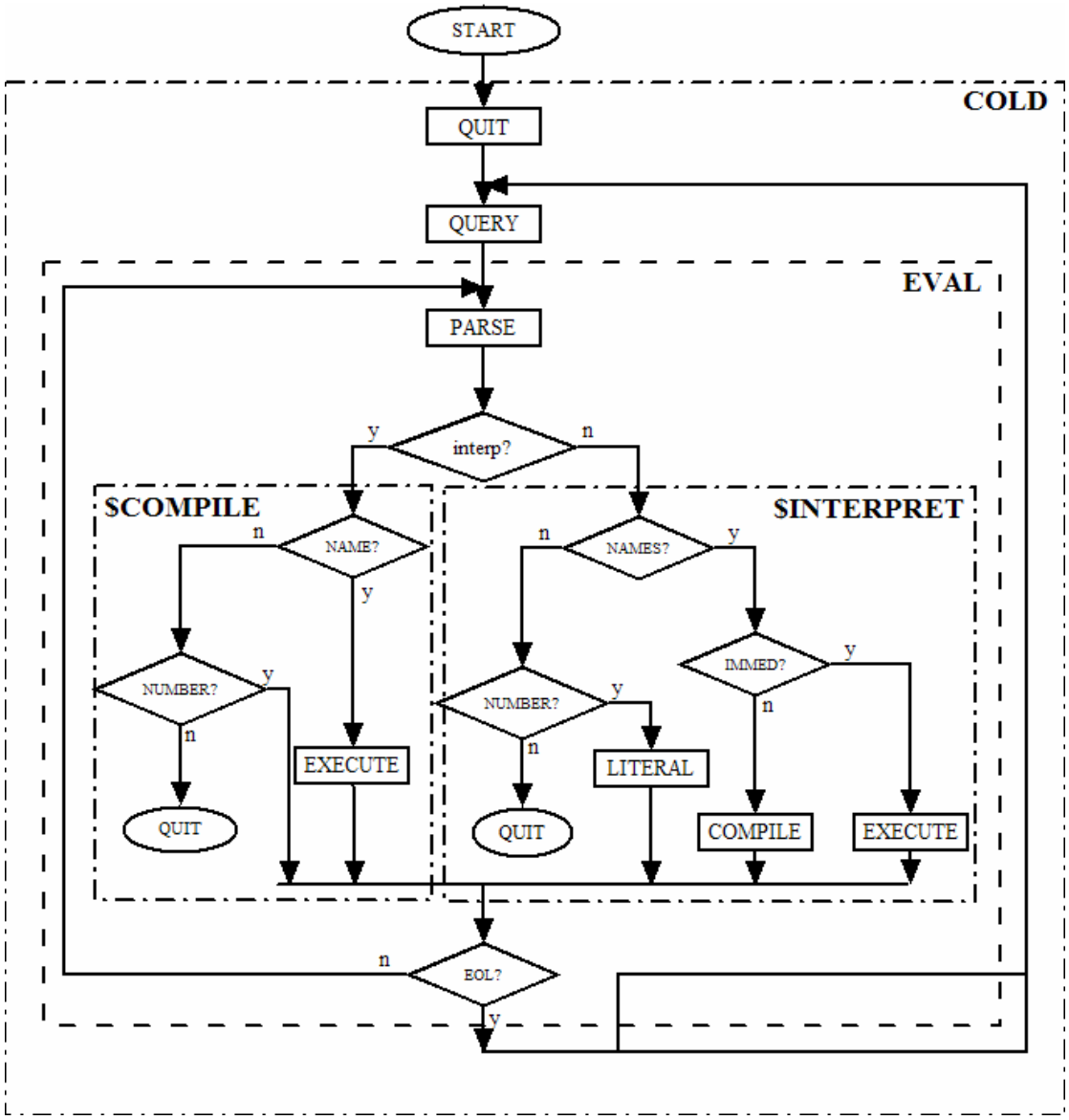

圖 2. Forth 處理指令程序圖

Forth 系統運作如圖 2。當系統接收到以鍵盤或檔案輸入的程式或資料時，text interpreter 會將輸入的資料以空白鍵為分隔字元，將所有的資料分成一個一個獨立的字 word。接著判斷字 word 的性質，若為一般的指令，則在 interpretation mode 直接執行中，若是屬於以’：‘開頭的新指令，則進入 compilation mode，在 Dictionary 中，建立新指

令和新指令的定義。一經建立編入 Dictionary 之後，新指令即可在 interpretation mode 中被執行。

2.2.3 使用的 Forth 系統

本研究使用的 Forth 系統有下列二種:

(1) FSharp

FSharp 是一個非常精簡的系統，只有大約 6 Kbyte 的程式碼，所以很容易被植入 SoC 系統中。這具有 Forth 的系統晶片，事實上就是一部內建的小電腦，只要 CPU 讓這小電腦裡的 Forth 先動起來，其餘再複雜的工作，就容易處理了。在這樣的晶片系統中，所有 CPU 所可以到達的周邊裝置，就都可以加以規劃，測試。而 EP32 系統的程式發展，也都得以經由這 FSHARP 來進行。

(2) dsForth

dsForth 是一個與 ANS Forth 94 相容運行於 Win32 平台的 Forth 語言編譯器， dsForth 支援 Windows 95/98/ME/NT/2000 (Intel) and Windows CE (MIPS,SH3,ARM,Intel)……等平台.

為了加快檢驗 FML Browser 的功能，在 PC 端使用 dsForth 撰寫 Forth 程式，可較有效率產生具視窗元件設計功能的跨平台瀏覽器。

第三章 研究內容-- 跨平台瀏覽器 FML 構想

3.1 開發平台的問題與分析

在記憶體昂貴的年代， Forth 即以原始碼編輯器、編譯器、除錯器一體的架構，成為具整合環境的便利開發方式。但時至今日，在 PC 及 Windows 的資源豐沛和強大的運算能力的環境下，Forth 架構並未充份發揮其特色，且新一代的程序員已習慣了視覺化的開發方式，文字模式的 Forth 開發環境更相形失色。

3.1.1 Web 架構的發展現況

在 Web 架構中，依 Browser 端、邏輯、客戶端和伺服端的發展現況描述如下：

Browser 端的視覺呈現方面：

(1) 純 HTML：基本的文字、大小字型、顏色設定、超聯結

(2) 表單：提供基礎的互動功能

(3) TABLE：呈現更豐富的內容

(4) FRAME：減少 與 server 之間的通訊量

(5) DHTML/CSS：Document Model 非 Application Model。CSS 提供精準版面定位，較好地利用客戶端的字形資源。主要問題-各品牌及版本 browser 之間的不一致性，製作人員必需費很大力氣才能保證客戶端呈現出理想的效果。

(6) 多媒體擴充：Macromedia Flash 的成功證明了使用者對聲光效果和互動性的需求，遠超學者們的想像，Flash 充份利用客戶端的圖形能力，搭配功能強大的製作軟體，讓非 IT 背景的人員能夠發揮創意，製作高度互動，多采多姿的動畫和遊戲。有些網站更完全以 Flash 來建構。

邏輯方面：

(1)Browser Plug-ins：Plug-ins 是各家瀏覽器允許的擴充機制，直接以目的碼形式運行，因此不但不能跨平台，甚至也不能跨 Browser 。

(2)Java Applet：JAVA 最初就是為了要解決分散式運算一項重要的需求。程序編寫一次，處處可行。Java 以 byte code 型式分配到客戶端，必需依賴虛擬機器方可運行。

(3)Scripting ： Microsoft 試圖用 VBScript 來取代 JavaScript，意外地沒有成功。 事實上 JavaScript 在此扮演著一個重要的角色：文件物件模型(DOM)的控制語言。舉凡事件處理、表單核查、各類視覺(DHTML/SVG)元素的控制，都非用 Javascript 不可。有了 Javascript，更多的邏輯可以移到客戶端運行，從而降低頻寬的需求和系統的複雜度。

因此，經仔細分析思考，雖具優勢的 Web 架構，若要適應未來應用，客戶端與伺服端的技術仍存在待解決的問題。

3.1.2 客戶端技術的問題

(1) 圖形能力的嚴重不足

HTML 的圖形能力極為有限，圖形必需在伺服端轉成圖片傳輸到客戶端顯示。也就是說，如果想呈現特殊的字形變化，甘特圖，統計圖表等，都必須以圖檔形式傳遞，頻寬消耗很大。Flash 可以克服這個問題，但由於 Flash 以 Plug-in 形式存在，是一個獨立的顯示區域，終究無法和 HTML 網頁圖文做充份的整合。

(2) Microsoft 的獨霸

目前 Browser 在 PC 上的產品，只剩下 IE, Netscape 和 Opera，其中 IE 的佔有率超過 90% ，和 Windows 作業系統的佔有率很接近。 Netscape, Konquerer 主要以 Linux 使用者為主。Microsoft 挾作業系統、編製工具(FrontPage)和使用者基數的優勢，全面地主

導瀏覽器的發展方向，短期內不可能出現能與之抗衡的瀏覽器，在架構不變的前提下，任何的挑戰都是相當艱辛的。

(3) 複雜度

隨著 HTML 功能越來越多，複雜度已經到了難以控制的程度，而且由於能夠上網的設備日益多樣化，HTML 的複雜度已嚴重影響到系統開發的成本和效率。因此才有 WML 等減重的嘗試。

3.1.3 伺服端的發展

原本 Web 伺服器只能提供靜態網頁，所謂的「靜態」，指網頁在伺服端和客戶端的內容一致，網頁內容必須事先編寫完成。但真正有用的是「動態網頁」：根據客戶的需求，即時產生的網頁。搜尋引擎、討論區、訂購服務、競標等等應用，必需是動態生成的網頁。如：

(1) CGI： 這是最早實現動態網頁的技術，幾乎所有的 Web 伺服器都支持這個方案，缺點是：對每次的客戶端需求，必需執行一次程式，造成潛在的問題。

(2) Server Extension： 針對 CGI 的效率問題，某些伺服器會將內部的函式和資料結構開放出來，讓人們可以進行擴充。缺點是程式很難移植到其他的伺服器。

(3) Server Side Scripting： CGI 和 Server Extension 皆是以傳統程序為中心的思考，程式碼為主，HTML 標誌以字串型式輸出，這種型式的程式很容易造成維護的困難。PHP 擴充了 Server-Side-Include 的觀念。提出了以 HTML 為中心，嵌入程式碼的做法，由伺服器端執行這些程式碼(存取資料庫之類)，轉換成純 HTML 文件送到客戶端，這個架構大幅降低了 web 程序開發的困難度，更棒的是，美工人員也可以參與系統的開發，(HTML 編製工具只要將嵌入的程式碼當作註解略過即可)有了 Server Side Scripting ，動態網頁變得多采多姿起來。除了

PHP，還有 ASP（以 VB 為語言）、JSP 之類，皆為相同的架構。

(4) Servlet, J2EE： 由於 Server Side Scripting 門檻很低，漫不經心的程序員很容易寫出缺乏整體規劃，邏輯操作雜亂地散佈在一堆網頁之中，又造成了維護的困難。Java 提出 Servlet 和 J2EE 架構，讓程式員可以將重覆使用，安全和效率要求較高的程式碼預先編成二元程式。

3.1.4 伺服端技術的問題

(1) 無法準確掌握客戶端的事件：每次互動必需發出 HTTP 請求。

(2) 無法得知客戶端實際呈現效果。

(3) 沒有單一的解決方案：由於每種技術各有優缺點，各自解決部份問題，因此在一個實際運行的系統中，往往要混合使用幾好種技術。比方說，後端資料庫存取用 SQL，企業邏輯使用 PHP，前端表單驗證用 JavaScript，畫面的控制用 CSS，語法各異，並有各自的設定和調校方式，這造成系統整體複雜度大增，開發及維護成本大幅增加。

3.1.5 系統造成的障礙

為了要保證在各個系統中皆能呈現 HTML ，因此在設計之初，只能採取最基本的功能，以表單（FORM）為例，HTML 僅支援最簡單的按鈕，下拉選單，文字輸入盒等，進階的視覺元件如樹狀結構、滑桿(Slider)、複合盒 (ComboxBox)還有自訂的元件 (custom controls or widgets) 都沒有相應的 HTML 標誌。簡言之，HTML 為了普及度的緣故只能取各系統的交集。如果採用越多「特殊功能」，越難保證在客戶端的呈現效果，由於這個限制，Web 架構無法實現高級的圖形介面系統。

另外，由於 Web 是架構在現行的圖形作業環境上，作業系統的不足也一併繼承，除了英文，其他語言必需依賴底層系統的字形，否則無法正確顯示，這更顯突出其他語言在網際網路世界呈現的問題

但由於 Web 的發展遠遠超出原先「交換文件」的設計，Web 架構的不足隨著應用的拓展逐漸突顯出來，為改善現行 Web 架構的不足與缺失，結合 Forth 提出一個像 HTML 簡單語法的 Forth Markup Language (FML)的構想，以解決問題，因 Forth 具有下列特點，非常適合作為網際網路所有子系統之間溝通母語。

(1) Forth 非常精簡易學，並有非常好的彈性。

(2) Forth 的執行效率非常高。

(3) Forth 也同時是一個 interpreter，處理結構化文件最為直接有效。

(4) Forth 是一個虛擬機器，在任何硬體上都很容易實作出來。

(5) Forth 很容易以硬體線路實現，這對嵌入式設備來說意義重大。

簡而言之，Forth 兼具編譯語言的效率嚴謹和直譯語言的彈性易學，如果說 XML 是描述資料的超語言，那 Forth 可說是同時能描述邏輯和資料的超語言。在下節的內容，我們將提出解決方案--以 FML 為基礎的網路服務構想，期望解決 WEB 架構面臨的問題。

3.2 提出 FML 構想

現行 Web 架構的缺失，不但造成了成本的增加，更嚴重地妨礙到中文資訊的傳播，因此若針對 WEB 架構的問題，進行重新規劃，以找出可行的解決方法。

3.2.1 Forth and FML

2003 年 7 月，首先由葉健欣先生提出了一個 FML 的構想，想一舉解決伺服端及客戶端所有邏輯和資料的描述需求，FML 和傳統 Forth 源碼的唯一差別是，註解和指令的位置互換。

傳統 Forth 源碼：指令是預設態，註解文字是特殊態（ ）、 \ 之後的文字。

FML　　：文字是預設態，指令是特殊態。＜＞內是指令。

FML 檔案內容為：

<Forth 指令>...文字...<Forth 指令>...文字...

3.2.2 FML 和 HTML 的不同

(1)HTML 每個標記只能有一個功能，和若干個參數。

例如，要呈現「有底線」的「大字」，必需用二個標記：<FONT SIZE=7><U>。FML 也可以這樣寫：<7 FontSize underline>，指令和參數可以結合起來，減少大量的＜＞。

(2)FML 可以立即定義新的標誌(指令)。

<: F7U 7 FontSize underline ;> 此後，F7U 就是一個新的標誌，<F7U>是有底線大字。這個方式可以讓網頁大幅簡化，清晰易讀。要創造中文標誌，也輕而易舉。

3.2.3 FML 的目標

在 FML 的構想下，FML 能夠：

(1) 取代 HTTP，成為機器與機器之間的控制及資料傳輸協定。
(2) 取代 HTML，成為頁面的排版語言。
(3) 取代 XML ，成為結構化資料的描述語言。
(4) 取代 Java Applet，可製作跨平台的視覺元件。
(5) 取代 Javascript ，成為互動事件和文件結構的控制語言。
(6) 取代部份 Flash 的功能，可允份利用客戶端的圖形資源，字形、圖形和報表皆可以直接用指令的方式於客戶端產生。
(7) 取代 PHP, Serverlet, CGI 等所有伺服端技術，直接操控伺服端所有軟硬體資源。
(8) 取代所有的遠端傳呼協定，如 RPC、DCOM、COBRA，以一致的語法和介面，傳遞跨平台邏輯。

(9) 加上了基本硬體和繪圖能力，甚至可取代現行的作業系統，做為嵌入式設備的標準圖形環境(GUI)。在這個環境下，所有的應用程式都是以 FML Browser 為展現平台。

3.2.4 FML 與瀏覽器

傳統的作業系統，將應用程式當作系統穩定性的大敵，重重的保護，來限制應用程式的活動範圍，其代價即是複雜度的增加和效率的損失。從 Forth 的觀點，應用程式即作業系統的延伸，兩者是沒有差別的。

同樣地，傳統的瀏覽器，將 HTML 當作資料來處理，HTML 加入新的標誌語法，瀏覽器就必需內建更多的處理程序來應付之，這直接導致瀏覽器日益龐大和複雜化，而這些先進的功能，往往不被充份使用，使用者卻必需揹負額外的開銷。

從 FML 的觀點，超文件即是瀏覽器的延伸，瀏覽器既有的功能，直接調用之，瀏覽器不足或不適之處，FML 定義出新的操作方式，被編譯後隨即和瀏覽器融為一體。因此瀏覽器可以變得非常精巧，而又可以處理所有新的需求。

3.2.5 FML 的影響

(1) FML 解決了版本升級問題，因為每一份超文件，都能引導瀏覽器「學習」新的能力。

(2) Web 由原先的主從架構，蛻變為平權式分散架構，伺服端和客戶端的角色變得模糊，換句話說，要求服務時，是客戶端，提供服務時，則為伺服端，這雙重角色，可由同一個 Forth 系統扮演。

由於 FML 語言的單純化，省去各種語法之間的介面轉換和彼此牽制，整體系統因此變得非常簡潔高效，並且允許開發者徹底拋開作業系統的限制，完全從應用本身的角度出發，量身訂作出最適當、最精簡的系統。

第四章 FML 應用程式開發平台之實現步驟

4.1 實現步驟

根據 FML 構想，應用程式平台的實現，初步分為下列二階段：

第一階段：FML Browser，運行於 Windows, WinCE, Linux, FPGA

(1) 硬體虛擬層：模擬出基本的圖形能力和滑鼠鍵盤介面，從這之後，都是平台無關的程式碼。

(2) 一個不考慮重疊的視窗系統，處理座標轉換及訊息分派。

(3) 基本視覺元件集：按鈕、下拉式選單、文字輸入盒等等。

(4) 自動排版：利用巢狀語法，可排出任意複雜度的頁面。排版單位可使用百分比讓不同解析度之下有一致的呈現效果。

此階段的目標為一個支援使用圖形界面的視窗應用程式開發平台。

第二階段：

- 圖檔格式：支援 Jpg , Png 格式
- 向量繪圖能力：實作部份 Flash 功能
- 整合 TCP/IP 通訊協定。
- 整合易符字形產生器及輸入法，徹底解決字形問題。
- 表格處理：可排各式表格。
- 中文排版：中文直排、左右上中下對齊、分散對齊。

此階段待移植到 EP32 運行後，評估檢討結果後再進行。故不在此論文實作範圍內。

4.2 建立 FML 瀏覽器

為快速建立 FML Browser 以驗證其設計目的之可行性。在 PC 端選擇可編製出含 WinCE、Macintosh、Windows 等跨平台功能的 dsForth 編譯器，以 ANSI Forth 撰寫具 GUI framework 功能的 FML Browser。

完成階段一之工作，僅占 130K 的 FML Browser 已確實具備了下列

HTML 所缺乏的功能：

(1) 視窗元件設計功能，如：HTML 缺乏自行畫圖功能，FML 不但可以任意畫線或其他圖形，還具開有發視窗應用程式的功能。

(2) 靈活的排版功能，如：可選擇絕對座標或相對座標的方式，可任意縮放應用程式之排列畫面，又如經 TEXTs 元件設定，同一水平列上，可同時包含向上、向中、向下等不同的對齊方式。

(3) 具程式邏輯及自行擴充指令標記的能力：可靈活地新增指令功能，解決等待昇級問題。

有了基本硬體和繪圖能力，即可取代現行的作業系統，做為嵌入式設備的標準圖形環境。在這個環境下，所有的應用程式都是以 FML Browser 為展現平台。

4.2.1 視窗元件設計：基本的 GUI 工具組

在 MS-Windows 或 Macintosh 上的程式設計工具都已經有許多高階的功能，像是按鈕、捲軸、顏色處理、字形以及其他視覺元件。 但在其他的系統環境，只提供畫出基本圖形的功能，例如畫直線與矩形，設定前景與背景顏色，此侷限的繪圖功能難以用來撰寫程式。因此，一些 UNIX 作業系統如 Solaris, HP-UX 所附的通用桌面環境(Common Desktop Environment, CDE)即是利用 GUI 工具組發展出來的。若沒有一套外觀感覺(look-and feel)標準的 GUI 元件，設計複雜的使用者介面，將會花費程式設計員太多時間產生使用者界面，而非集中精神於寫用軟體的核心。利用 Application Framework 的程式設計系統，應用程式看不到無聊、容易出錯的低階細節。如此，設計師就能更專注於發展應用軟體的核心功能，產生使用者介面時也變得更簡單容易了。

FML Browser 即是一套跨平台的 Forth Framework，由硬體虛擬層程式碼模擬出基本的圖形能力和滑鼠鍵盤介面，從這之後，都是平台無關的程式碼。

Forth GUI 在視窗元件設計的主要重點如下：

- Widget：「window 視窗」與 「gadget 裝置」的結合字，即是一個視窗的控制項，在使用者界面中看到的所有東西都是 widget。按鈕是一個 widget,捲軸是一個 widget，文字輸入欄是一個 widget，一個 widget 也可以有子 Widget (subwidget)，例如一個 Combobox 中包含了一個文字輸入欄和一個 list box。

FML BROWSER 之基本的 Widget sets 含非視覺元件如：FRAME、VIEW、CONTAINER、TEXTS 及 視覺元件 如：BOX、BUTTON、PUSH BUTTON、RADIO BUTTON、CHECK BUTTON、HYPERLINK、LINE EDIT、LISTBOX、COMBO、SCORLLBAR 等。

為了程式方便，使用物件導向的設計方法(object orientation)，把這些會依照不同事件做出不同反應的視窗做個分類。例如：螢幕上一個矩形，所以它有一個座標值表示它 的位置及大小 X , Y , Width , Height 再來它能夠接收需要的事件(event)，而且我們能在上面畫出基本的圖形。

接著我們再從 Window 中細分一類 Button 這一類視窗是專門用來處理 click 的動作，當我們把滑鼠移到它們的上面按下之後，他就會有特定的反應。然後我們從 Button 中細分出兩種：一種 CheckButton ，這種 Button 上面有一個特殊的方框，當我們在上面按一下時就會產生選取的動作，接著方框上會產生一個打勾的記號，表示我們已經選取。

另一種是 RadioButton 這種 Button 上面有一個圓形，當我們在上面按下時，同樣會產生一個圓形的記號表示我們已經選取。因為 CheckButton 跟 RadioButton 同樣都屬於 Window ，也都屬於 Button 所以它們當然也會具有 Window 跟 Button 的特性。進一步把 widget 擴充到由多個視窗所組成，如 COMBO 由 PULLDOWN BUTTON、EDIT、POPUP+LISTBOX 所組成設。

- 事件處理

在事件處理 (event handling)的設計上，介紹 FML Browser 在訊息偵測與分派處理及座標轉換的處理。

當使用者輸入按鍵或滑鼠，告訴應用程式要做哪些動作，而不是由應用程式來主導要做什麼，稱為事件驅動(event driven)，而用 window 來管理這些事件就相當的方便。像這樣的 window system 一般在內部實做的方式由 device 送進來的 event ，送到系統的 event queue 中，然後再由系統把各個 event 送到各個應用程式專屬的 event queue 中，然後應用程式再從自己的 queue 中，取得要送給它的事件。這時我們就可以把各個事件分配到所屬的視窗中，然後由該視窗視窗做出適當的反應。接著應用程式再到它專屬的 event queue 中取得下一個要處理的事件。

當我們把各個 widget 都定義好之後，剩下要做的大概就是下列這幾項：把 event queue 中的 event 分配到各個 widget 中。其作法是用個表格紀錄各個 widget 的 identify 及所需的 event，檢查 widget 跟 event 中的 identify，當 identify 相符時就把該事件送到相對應的 widget 中。widget 與 widget 之間要互動，如一個文書編輯器，在編輯視窗旁邊附有一個 Scrollbar，當移動 Scrollbar 時編輯視窗的內容要跟著捲動到合適的地方。這時就需要 Scrollbar 把它的值傳送給編輯視窗。當有事件發生時 widget 要適當的做出反應，當按下 "Quit" 的 Button 時，應該要做出必要的反應並結束程式。這些問題，大多數的 widget set 用"callback"的方式來處理。

- FML Browser 的訊息偵測與分派處理：

開始處理 FML 程式時，即對 Main widget 與 widgets 間，建造親子關係。再針對建好關係的 Widgets，利用簡單的 LINK LIST 架構，進行訊息偵測與分派處理。

BEGIN c WHILE

```
c mouseinrange?
c f.visible AND c f.enabled AND
IF                                    \偵測成功，訊息分派
  lp wp msg c NOTICE -> o
  -2 o = IF o -> w EXIT THEN
THEN
c f.next -> c                         \偵測未成功，繼續偵測
REPEAT
```

圖 3 為含視窗元件的 FML 電子書應用之執行結果。

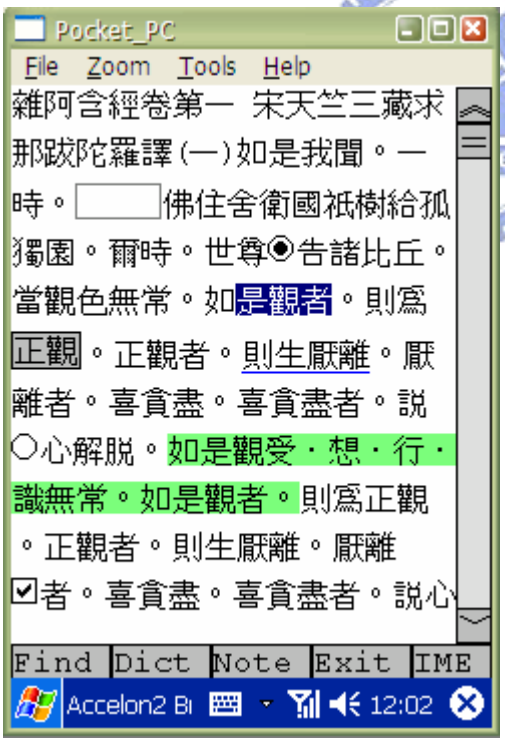

圖 3. FML 電子書的應用

4.2.2 FML 的排版功能

- 自動排版原理

利用 RECURSIVE CONTAINER 巢狀語法，可排出任意複雜度的頁面。排版單位可使用百分比，讓不同解析度之下有一致的呈現效果。

在 FML 中使用排版說明如下：

1. 整個畫面就是一個空的容器 CONTAINER
2. 空白區是一個矩形，未安置任何元件時，空白區就是整個畫面
3. 新元件加到容器時，空白區會縮小
4. 新元件只能擺放在上下左右四個角落
5. 用 % 或 PX 來指定新元件的大小， 50 % 表示新元件將佔據 50 %的空白區.
6. 擺放方式和大小必需在元件產生前指定
7. 容器也是一種元件

FML 語法與簡單的版面設定範例

範例一 ：三個方框，

指令：<50 % aTOP BOX
CONTAINER 50 % BOX BOX
/CONTAINER>

範例二 ：放置在中央，必須先用 VIEW 佔據位置

指令：

< 25 % aTOP VIEW aBOTTOM VIEW
100 % CONTAINER
25 % VIEW aRIGHT VIEW
100 % BOX
/CONTAINER >

- TEXTS 元件功能

TEXTS 是一個擁有 TEXT FORMATTING 功能的視窗控制元件,支援預設好的自動換行 auto-wrapping 和基本的 HTML TAGS 如 BR、H1、H2、PRE,除此之外，使用者也可容易地自訂需要的 TAG。

和 HTML 不同的是，在 FML Browser 中，允許在同一行的文字中做多種不同的對齊方式。在同一列的水平文字時，設定為 aMIDDLE，文字會排在行的中間位置，設定為 aMIDDLE 時，文字會排在行的中間位置，設定為 aEND，文字會排在行的末端位置。aSKY aHOVER aGROUND 控制文字的垂直對齊方式。

TEXTS 文字對齊的範例如圖 5 所示:

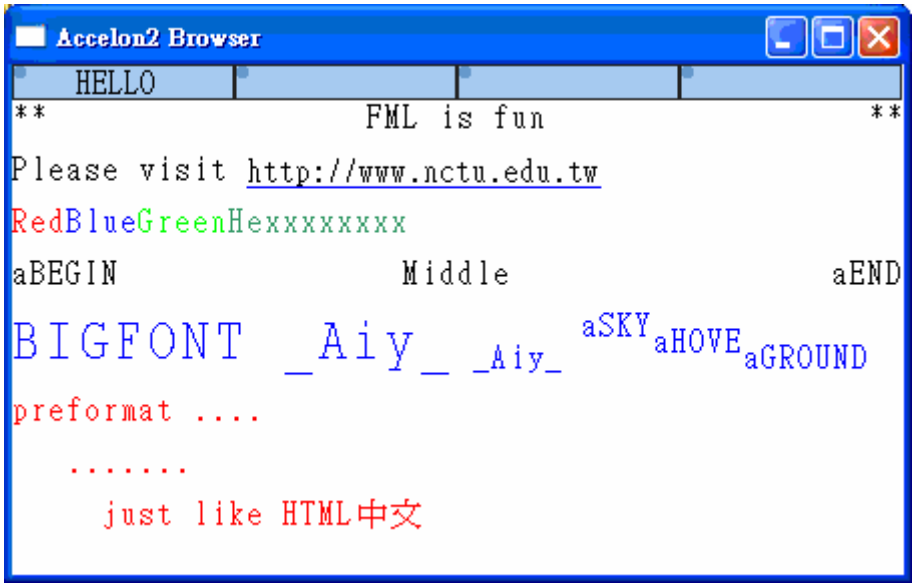

圖 4. FML 之 TEXTS 範例

圖 4 為 FML 之 TEXTS 範例，可處理文字格式的變化與設定，對齊方向的語法與編排和文字段落與換行。RED、 BLUE、 GREEN 為內建的顏色設定值,TEXTCOLOR 可 接受 RGB 值。HYPERLINK 是 BUTTON 的一種型態,可在 ONCLICK handle 中設定觸發後所要處理的事項。Before line feed 計算行高，調整下一行應留的行距，再沿用原設定繼續排版。

4.2.3 可彈性擴充的程式邏輯與自訂新功能

由於兼具同時能描述邏輯和資料的能力,在FML檔案中可直接處理由Forth撰寫的程式或自訂新功能的指令，讓應用程式的開發，不必使用額外的支援或等待具新功能的版本，在一個具有嵌入式視窗應用程式開發功能，且僅占140K空間的瀏覽器程式中，即可進行更有彈性的擴充功能。例如欲在FML中自行撰寫的圖形切換程式，首先可在playwav.f中以Forth寫所需執行的程式邏輯，在FML檔案中，即可執行playwav.f所新增的指令，自行新定義的標記，也可立刻被使用。

4.3 移植FML瀏覽器到堆疊架構處理器EP32

由於EP32上的作業系統核心－FSharp 僅有6K，功能極為精簡，並沒有處理複雜的視窗繪圖等功能。而FML Browser是使用dsForth所完成。因此，將PC端調用作業系統的基本視窗處理機制等功能移植到堆疊架構的EP32平台上，與硬體無關的程式即可在EP32上運行。

4.3.1 移植方式及其優缺點

移植前，評估可進行的移植方式及優缺點，簡述如下：

(1) 將dsForth的指令功能全數移植到EP32的FSharp上，使EP32可執行由dsFORTH開發之FBrowser程式。

優點：此為最根本的解決方式，能解決版本的一致性。PC端與EP32嵌入式環境所執行的dsFORTH版本相同，未來FBrowser程式的維護、新增與開發不會有版本不同的問題。

缺點：從比較dsForth和Fsharp二者之差異、到完成所有Fsharp缺乏的指令功能，移植dsFortht 的工作較費時。而dsForth功能複雜以評估 FBrowser 瀏覽器在 EP32 應用為目的之階

段，，若連許多不需要的指令功能也全數移植，增加移植工作量與 EP32 系統的負擔。

(2) 提供 EP32 平台的硬體指令功能，讓 dsForth 設計者增加 EP32 平台的功能。

優點：由 dsForth 設計者自行增加 EP32 之跨平台功能。可省去 EP32 環境之移植的工作。

缺點：需等待 dsForth 廠商支援。由於 EP32 尚屬研發中的架構，dsForth 廠商未必會支援。

(3) 在 FSHARP 上新增 FML BROWSER 使用之指令功能。

優點：移植工作較少，節省移植時間，僅補齊 FBROWSER 所使用之功能即可。

缺點：僅能暫時解決第一階段功能的工作，未來仍需不斷地面對 FSHARP、dsFORTH 、FBROWSER 版本不一等的問題。

為節省時間，以進行評估研究目的之便，僅採取在 FSHARP 上新增 FML Broswer 使用之指令功能。

4.3.2 移植工作

移植工作，主要需完成：

(1) 在 EP32 的 FSharp 中新增視窗處理機制。

(2) FSharp Forth 指令集和 dsForth ANSI Forth 指令集比較與轉換工作。有許多指令執行內容相同，但不同版本，使用不同的指令名稱，需要先將指令名稱確定。

(3) LOCAL VARIABLE FRAME 之新增：因 FSharp 系統並沒有 Local Variable Frame 之架構，因此需新增此架構。

4.4 應用實例

4.4.1 EP32 模擬器

FML Browser 在 EP32 嵌入式系統之移植操作與測試，主要透過圖 5 在 PC 端的 EP32 模擬器:

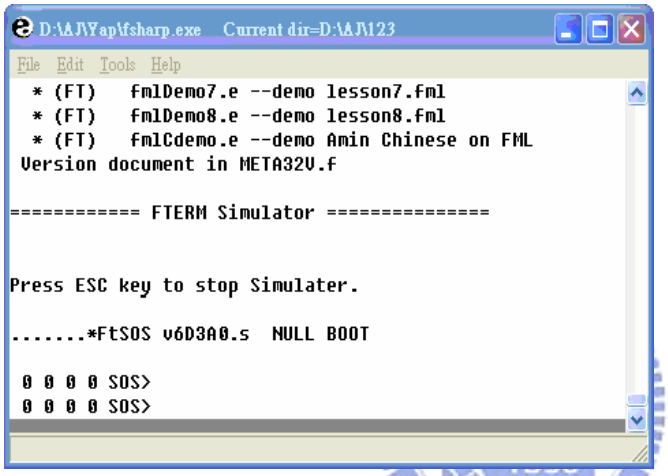

圖 5. EP32 模擬器

在此環境下，即可在 PC 端進行在 EP32 上的程式設計與測試，模擬測試之步驟流程：

輸入 XREAD,讀取可在 EP32 執行的 FBrowser 程式。則可在 PC 端進入檢示 EP32 的 LCD 顯示畫面及啟動 FBrowser 設計環境。此時，讀進 FML 檔案，執行後即可在 PC 端模擬在 EP32 + FSharp 組成的嵌入式系統中運行。

4.4.2 在 LCD 裝置上的運行結果

將 EP32 版的瀏覽器程式，和可在 PC 上以 FML 撰寫且正確執行的自動控制面版程式寫入以 EP32 為核心的嵌入式設備的 LCD 裝置上，其運行結果如圖 6。

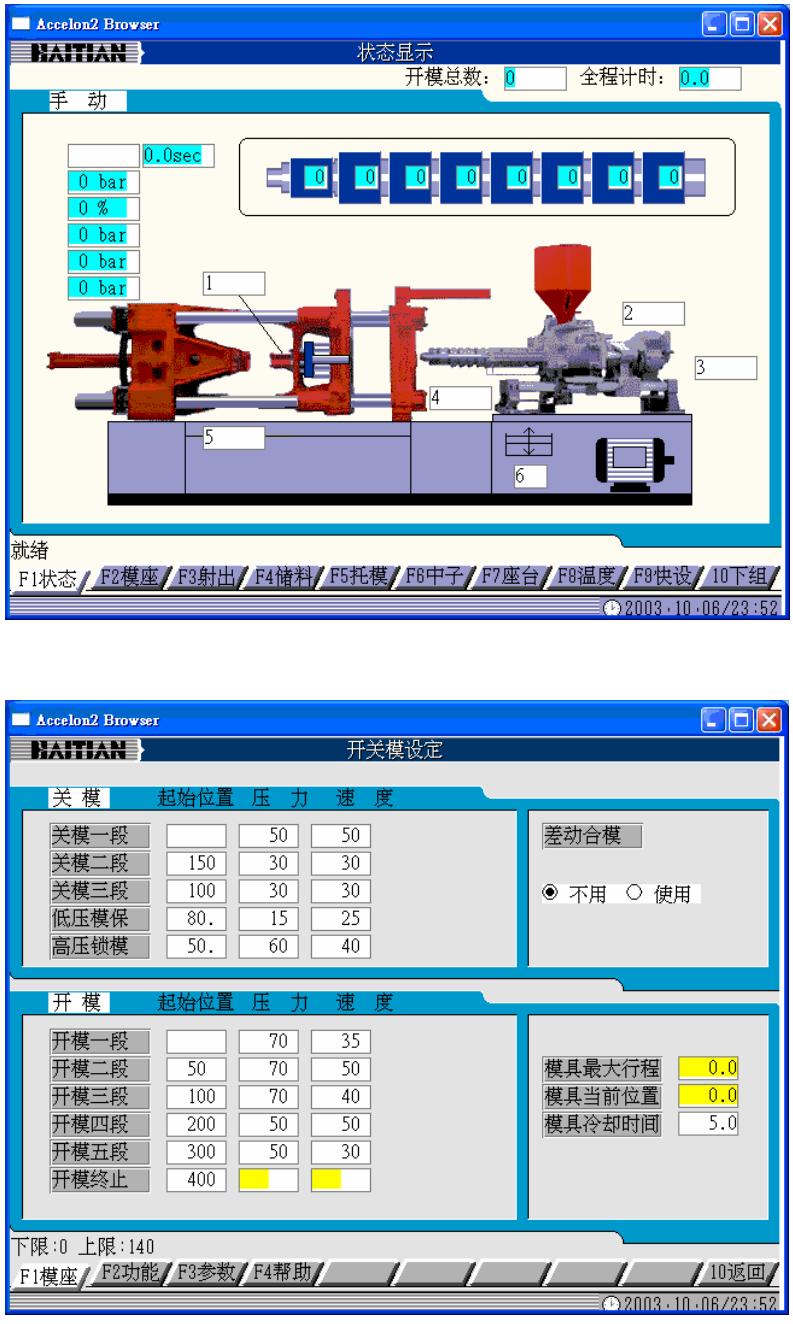

圖 6. FML 在 EP32 的運行結果

第五章 評估與檢討

5.1 FML BROWSER 評估結果

在 EP32 架構的嵌入式設備中，利用 FML 所撰寫的程式，如電子書瀏覽註解功能、工業應用的自動控制面版的測試範例等，確實可如預期般地正常運作。FML BROWSER 確實具備了下列特點：

(1) 提供一套使用者圖形界面的視窗元件，提昇了堆疊架構的嵌入式視窗應用程式的開發效率。

(2) 提供與 HTML 相同的標記功能之外，還包含了更有彈性的自動排版功能，並允許在 TEXTS 元件中，同一列文字可設定不同的垂直對齊方式。

(3) 提供彈性的繪圖及程式邏輯功能。

(4) 提供使用者自訂指令的擴充功能，在最精簡的硬體資源下，不增加額外的負擔下，以基本的功能，發展更合適的客製化功能的程式。

5.2 面臨的問題

從移植工作中發現，在 FSharp 和 dsForth 下，dsForth 採 ANSI Forth 標準，二者皆可進行 Forth 高階語言的撰寫，但使用 Forth 太容易因個人喜好或需求，迅速地編制有彈性的新指令，因此，常有具相同功能的指令，其名稱卻不盡相同。增加了程式間整合的困難度。

由於 Forth 語言過於強調客製化功能及精簡性，標準性未受重視，要發展高相容性的商業產品並不容易。在強調重覆使用與可迅速整合的環境下，堆疊式的嵌入式系統開發環境，欲取得軟硬體廠商的配合與支援，要有相當大的努力與調整。

因 FML 之彈性極大，使用者能任意擴充功能，如此一來，大幅增加

了要設計一套完整支援如此彈性的 IDE 時的困難性。

因此，針對上述問題，堆疊架構的嵌入式系統環境仍有相當大的改進空間，儘管其具有相當精簡及客制化的擴充彈性，但應考量其面臨之瓶頸，全力克服，方可為未來的嵌入式軟體環境提供更佳解決方案。

5.3 改進與加強

為了解決 FML Browser IDE 編輯工具之開發， FML Browser 可新增 pack 和 unpack 的機制。如加上 ToString 屬性記錄每個 WIDGET 的特性，完成 PACK 機制，需要切換到 VISUAL MODE 時，再以 BinarytoTEXT 完成 UNPACK 機制，來解決原架構所造成 IDE 編輯工具開發之衝突，以順利地在 TEXT MODE 與 VISUAL MODE 環境切換中，仍擁有正確之程式碼。

第六章 結論與未來工作

6.1 結論

經 FML Browser 實作與移植後的實際運作，此平台確實有下列的優點：

(1) 可做成嵌入式設備的標準圖形環境。

(2) 可快速的開發視窗應用程式。

(3) 可輕易地展現繪圖與自動排版功能。

(4) 解決伺服端及客戶端所有邏輯和資料的描述需求。

(5) 解決版本昇級問題，每一份超文件，都能引導瀏覽器學習新的能力。

(6) 客戶端僅用資源有限的精簡核心，即可支援自行擴充功能的技術。

由此亦達成了此研究預期的的目：使用一個簡單易學的描述語言，開發嵌入式應用程式，並可自行增加延伸擴充功能之彈性。

在嵌入式客戶端設備上，提高客戶端設備之瀏覽器功能，使客戶端不需仰賴伺服端的功能，其與伺服端達成平權式功能。不需要靠外掛，即可執行加外有能力處理圖形能力、程式邏輯等能力。

在堆疊架構的嵌入式系統開發環境上，提供一個堆疊架構的嵌入式應用程式開發平台，以提昇開發視窗應用程式之設計效率，更容易設計出具使用者親和力的嵌入式應用程式。為一個精簡且節省成本的嵌入式硬體環境，提供更多元的解決方案，提供有效率具彈性的嵌入式系統開發的核心技術。

儘管堆疊式架構的嵌入式系統開發環境面臨許多待克服的問題，由於產業為提昇競爭力，面臨未來應用服務的轉型，許多規模不大，但占台灣的相當高比例的傳統產業，無不思考如何以最省成本的方式尋找最佳可行性方案。因擁有自創且精簡的堆疊架構核心技術，不必面對國外

大廠專利及授權談判等問題，其在未來網路及簡單的省電型嵌入式裝置上的應用，已受到台灣中小型傳統機械產業的青睞。若能針對堆疊架構嵌入式系統所面臨的問題，繼續努力尋找解決之道，開發完善的軟硬體環境與相關的技術支援。堆疊架構的關鍵技術或可為未來嵌入式的多元化應用做出貢獻。

6.2 未來工作

本論文僅針對 FML Browser 平台實現之第一階段工作，及其移植到 EP32 之應用進行研究評估，並未探討階段二中 FML Browser 的進階功能：

- 整合易符字形產生器及輸入法，徹底解決字形問題。
- 表格處理：可排各式表格。
- 中文排版：中文直排、左右上中下對齊、分散對齊。
- 圖檔格式：支援 Jpg , Png 格式
- 向量繪圖能力：實作部份 Flash 功能
- 整合 TCP/IP 通訊協定。

除了本論文提出面對標準化及 FML IDE 編輯工具開發的問題，需克服解決外，後續亦可考慮以 FML 和微軟下一代的視窗作業平台中的 XAML(eXtensible Application Markup Language)應用程式開發語言之整合為可行的研究發展方向。

由於 FML 語法與功能和 XAML 相近，未來 FML 若能發展成為 XAML 子集，其運行在堆疊架構的處理器上，將可省去 XAML 堆疊架構的中間碼轉換時所犧牲的效率問題。此種模式的整合與改良可做為提高堆疊架構的嵌入式應用，但因微軟不斷延後推出的 Longhorn，亦未正式公開 XAML 技術，目前亦無相關研究探討此模式的應用。

參考資料

1. Amdahl, G. M., G. A. Blaauw, and F. P. Brooks, Jr., “Architecture of the IBM System 360,” IBM J. Research and Development 8:2, 87-101, 1964.
2. Barton, R. S., “A New Approach to the Functional Design of a Computer,” Proc Western Joint Computer Conf., 393-396, 1961.
3. Bell, G., R. Cady, H. McFarland, B. Delagi, J. O’Laughlin, R. Noonan, and W. Wulf, “A new architecture for mini-computers: The DEC PDP-11,” Proc. AFIPS SJCC,657-675, 1970.
4. Bell, G., and W. D. Strecker, “Computer structures: What have we learned from the PDP-11?” 25 Years of the International Symposia on Computer Architecture, ACM, 138-151, 1998.
5. Brodie, Leo, ”Starting FORTH: an introduction to the FORTH language and operating system for beginners and professionals”, Prentice-Hall, 1981.
6. Brodie, Leo, ”Thinking FORTH : a language and philosophy for solving problems”, Prentice-Hall, 1984.
7. Hauck, E. A., and Dent, B. A. “Burroughs’ B6500/B7500 stack mechanism,” Proc. AFIPS SJCC, 245-251, 1968.
8. Koopman, P. “Stack Computers-The New Wave”, Ellis Horwood, 1989.
9. Lindholm, T., and Yellin, F. “The Java Virtual Machine Specification”, Second Edition, Addison-Wesley, 1999.
10. McGhan, H., and O’Connor, M. “PicoJava: A direct execution engine for Java bytecode,” Computer 31:10, 22-30, 1998.
11. 丁陳漢蓀, ”嵌入式系統-使用 eFORTH”, O’REILLY, 2003.
12. www.eforth.com.tw 易符智慧科技股份有限公司