**嵌入式系統設計期末專題作業**

**微型嵌入式資料庫系統設計**

一個微型嵌入式資料庫系統的階層式模組架構如圖1所示，最上層系統A0；第二層有三個模組A1, A2,和A3。當我們取得感測資料之後，會由資料加解密模組A1將資料進行處理，再將其透過資料庫模組A2的演算法及檔案系統模組A3與外部儲存媒介的溝通將資料進行存取。



圖1 嵌入式資料庫引擎系統階層式架構

1 資料庫引擎模組

## 資料庫與結構化查詢語言

結構化查詢語言(Structured Query Language，SQL)，是普遍用於資料庫撰寫及應用的語言。以下將簡介結構化查詢語言的發展及其語法組成的元素，本論文會使用結構化查詢語言的指令集作為輔助，著重在資料查詢的指令來進行支援實作。

最早由IBM公司San Jose, California研究實驗室的研究員Edgar Frank Codd於1970年發表提出關聯式資料庫（Relational Database）的概念[13]，主張將原本單獨以檔案形式儲存在磁碟中的資料以表格的形式來處理及呈現，透過表格的呈現模式，讓使用者能輕鬆進行資料查詢及處理，對於大量的資料也能夠有效率的管理，而Codd也因此篇論文獲得了1976年的Turing Award，而被世人稱為「關聯式資料庫之父」。之後Donald Dean Chamberlin與Raymond Francis Boyce在1974年時提出了一套用於關聯式資料庫的語言「Structured English Query Language，SEQUEL」[14]，後來因為註冊商標權的問題，於1980年改名為結構化查詢語言。

結構化查詢語言剛開始只被使用在IBM自行開發的關聯式資料庫系統System R中，1986年美國國家標準局（American National Standards Institute，ANSI）採用了X3H2資料庫技術委員會（X3H2 Technical Committee on Database）所提出的結構化查詢語言之關聯式查詢語言標準，包含函式及語法作為關聯式資料庫系統的標準語言，稱之為ANSI SQL-86。1987年，國際標準化組織（International Organization for Standardization，ISO）將結構化查詢語言採納為國際標準。之後歷經了數次改版，衍生出ANSI SQL-89、ANSI SQL-92、SQL:1999等不同標準，其中以ANSI SQL-92的標準受到最多資料庫的支援。

### 語法元素

結構化查詢語言實際上是一組分別進行不同工作的指令集，依據各指令的作用不同，又可分類為以下四種：

* 資料查詢語言（Data Query Language， DQL）：負責進行資料查詢的指令，其核心指令為SELECT，是資料庫中最常被使用的一個，因此本論文主要使用此指令作為查詢輔助用。SELECT基本語法結構如圖2所示。



圖2 SELECT 語法架構

* 資料定義語言（Data Definition Language，DDL）：負責定義資料庫物件與資料表中的資料結構，由CREATE、ALTER和DROP三組陳述句組成，分別代表建立資料庫物件、修改資料庫物件結構與刪除資料庫物件。
* 資料操作語言（Data Manipulation Language，DML）：對資料庫物件執行資料存取工作，由INSERT、UPDATE和DELETE三組陳述句組成，分別代表新增資料、更新資料與刪除資料，通常與DQL中的SELECT指令合稱為CRUD，其中C代表Create，對應INSERT陳述句實際上進行的建立資料動作；R代表Read，對應SELECT陳述句實際上進行的讀取資料動作。INSERT、UPDATE、DELETE基本語法結構如圖3所示。



圖3 INSERT、UPDATE、DELETE基本語法架構

* 資料控制語言（Data Control Language，DCL）：控制特定使用者帳戶對於資料表、檢視表等資料庫物件的操控權限，由GRANT和REVOKE兩組陳述句組成，分別代表設定操作權限與撤銷操作權限。

進行資料庫開發，首要動作為系統分析，規劃出需要的欄位資訊，定義其長度、屬性等等。表1為感測資料表相關欄位及說明、表2為使用者資料表相關欄位及說明，為本論文於實驗開發上必須要先定義的兩個資料表，將每一個資料表對應至檔案配置文件系統，即為一個文件檔案，故本模組將會有兩個文件資料，分別存取感測資料及使用者資料，如圖4。



圖4資料庫資料表及欄位關係圖

表1 感測資料表相關欄位及說明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表格名稱 | 感測資料表T\_Value | | 時間：2014/10/01 | |
| 欄位中文說明 | 欄位名稱 | 屬性 | 長度 | 範例 |
| 裝置名稱 | Device\_Name | Char | 4 | 0001 |
| 年 | YY | Char | 4 | 2014 |
| 月 | MM | Char | 2 | 10 |
| 日 | DD | Char | 2 | 01 |
| 時 | HH | Char | 2 | 12 |
| 分 | Mm | Char | 2 | 05 |
| 測量值 | Value | Char | 3 | 28/-05 |

表2 使用者資料表相關欄位及說明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表格名稱 | 使用者資料表guest | | 時間：2014/10/01 | |
| 欄位中文說明 | 欄位名稱 | 屬性 | 長度 | 範例 |
| 帳號 | ID | Char | 4 | A001 |
| 密碼 | PW | Char | 4 | 1234 |
| 權限 | LEV | Char | 1 | 0沒有權限  1有權限 |

資料庫引擎模組配合控制器屬於上層的應用，對底層的檔案系統模組做資料存取及查詢。資料庫引擎模組的離散事件建模如圖5、圖6所示，說明如表3。首先進行環境初始化及變數的設定，接下來則依需求進行新增語法(insert)或是查詢語法(Select)來處理資料，新增語法的Grafcet如圖7所示，表4說明新增語法的狀態轉移與執行動作；查詢語法的Grafcet如圖8所示，表5說明查詢語法的狀態轉移與執行動作。



圖5資料庫引擎模組Grafcet模型



圖6資料庫引擎模組Grafcet模型(續)

表3 資料庫引擎模組Grafcet狀態轉移與執行動作

|  |  |  |  |
| --- | --- | --- | --- |
| 現在狀態 | 執行動作 | 轉移條件 | 下個狀態 |
| 30 | -- | 1 | 31 |
| 31 | 環境初始化，變數設定 | 1 | 32~36 |
| 32 | 進行新增的演算法 | 1 | 30 |
| 33 | 進行查詢的演算法 | 1 | 30 |
| 34 | 進行建立表格的演算法 | 1 | 30 |
| 35 | 進行修改的演算法 | 1 | 30 |
| 36 | 進行刪除的演算法 | 1 | 30 |



圖7 新增的Grafcet模型

表4 新增的Grafcet狀態轉移與執行動作

|  |  |  |  |
| --- | --- | --- | --- |
| 現在狀態 | 執行動作 | 轉移條件 | 下個狀態 |
| 320 | -- | 1 | 321 |
| 321 | 變數設定 | 1 | 322 |
| 322 | 取到暫存位置寫入資料 | 1 | 323 |
| 323 | 計數加一 | 資料未寫入完  資料已寫入完 | 322  324 |
| 324 | 回傳資料 | 1 | 320 |



圖8 查詢的Grafcet模型

表5 查詢的Grafcet狀態轉移與執行動作

|  |  |  |  |
| --- | --- | --- | --- |
| 現在狀態 | 執行動作 | 轉移條件 | 下個狀態 |
| 330 | -- | 1 | 331 |
| 331 | 變數設定 | 1 | 332 |
| 332 | 找到暫存位置取得資料 | 1 | 333 |
| 333 | 計數加一 | 資料未讀取完  資料已讀取完 | 332  334 |
| 334 | 回傳資料 | 1 | 330 |

2 資料加解密模組設計

我們使用進階加密標準(Advanced Encryption Standard，AES)，於2001年由美國國家標準技術研究所(National Institute of Standards and Technology，NIST)發表[11]，為了取代舊有的資料加密標準(Data Encryption Standard，DES)並使其成為各種應用的公定標準，屬於對稱式加密系統。

資料加解密系統的發展，根據金鑰的類型可分為非對稱金鑰和對稱式金鑰加密系統。非對稱金鑰加密系統是使用兩種不一樣的加密和解密金鑰，並提供金鑰傳輸穩固機制；對稱式加密系統則是在加解密均使用相同的金鑰，架構較為簡單，對於效能的評估上，適合用於大量資料處理。

在眾多對稱式加密系統之中，進階加密標準是適用於軟硬體實現皆合適的有效系統。其演算法主要是利用數值置換和數值排列兩種運算所組成的，數值置換是將原本的數值經由查表法換成新的數值，而數值排列運算則是將原本的數值進行移位或是XOR的運算，可使用128、196、256位元的金鑰，利用128位元區塊做加解密的數據，其加密過程是在一個4×4的位元組矩陣上運作，加密時，進階加密標準的加密迴圈除了最後一次是以另一個AddRoundKey取代並省略MixColumns步驟，其他輪循均包含四個步驟：AddRoundKey，矩陣中的每一個位元組都與該次回合金鑰（Round Key）做XOR運算；每個子金鑰由金鑰生成函式產生。如圖9所示；SubBytes，矩陣中的各個位元組透過一個8位元的S-box進行轉換。S-box與上的乘法反元素有關，具有良好的非線性特性，以S-box替換位元組的結果會相當於錯排的結果。如圖10所示；ShiftRows，將矩陣中的橫列進行移位，其中的每個位元都向左循環位移某個[偏移量](http://en.wikipedia.org/wiki/Offset_(computer))。在進階加密標準中，其區塊大小為128位元，第一列維持不變，第二列裡的每個位元組都向左循環移動一格。同理，第三列及第四列向左循環位移的偏移量就分別是2和3。如圖11所示；MixColumns，使用線性轉換混合每行內所包含的四個位元組。而其中每一列的四個元素分別當作，，，的係數，合併即為中的一組多項式，接著則將此多項式和一個固定的多項式：2，於modulo 下相乘。MixColumns函式接受4個位元組的輸入，輸出4個位元組，每一個輸入的位元組都會對輸出的四個位元組造成影響。因此ShiftRows和MixColumns兩步驟為進階加密標準提供了[擴散性](http://en.wikipedia.org/wiki/diffusion(cryptograohy))。如圖12所示。



圖9 AddRoundKey



圖10 SubBytes



圖11 ShiftRows



圖12 MixColumns

資料加解密模組的目的是在於控制器接收到新增資料或是搜尋資料的指令之後，進行資料加密及解密的動作，再將資料回傳。圖13表示了資料加解密模組的Grafcet離散事件，狀態21為初始化及變數的設定，狀態22及狀態23則為加密及解密的行動，最後狀態24則為將加解密後的資料回傳。表6說明資料加解密模組Grafcet狀態轉移與執行動作。



圖13資料加解密的Grafcet模型

表6 資料加解密模組Grafcet狀態轉移與執行動作

|  |  |  |  |
| --- | --- | --- | --- |
| 現在狀態 | 執行動作 | 轉移條件 | 下個狀態 |
| 20 | -- | 1 | 21 |
| 21 | 初始化、變數設定 | 1 | 22 |
| 22 | 資料加密副程式 | 新增資料狀態 | 23 |
| 23 | 資料解密副程式 | 查詢資料狀態 | 24 |
| 24 | 回傳加解密後結果 | 1 | 20 |

資料加密的Grafcet離散事件模型如圖14，表7說明狀態轉移與執行動作。狀態221進行變數的設定，接著將欲加密的資料針對每個字執行數字置換的演算法做處理，如狀態222至狀態224，直到狀態225加密迴圈確認每個字都執行完畢，最後於狀態226進行AddRoundKey。SubBytes、ShiftRows、MixColumns以及AddRoundKey的Grafcet離散事件建模，均列於附錄當中。



圖14 資料加密的Grafcet模型

表7 資料加密的Grafcet狀態轉移與執行動作

|  |  |  |  |
| --- | --- | --- | --- |
| 現在狀態 | 執行動作 | 轉移條件 | 下個狀態 |
| 220 | -- | 1 | 221 |
| 221 | 變數設定 | 1 | 222 |
| 222 | SubBytes方式進行加密 | 1 | 223 |
| 223 | ShiftRows方式進行加密 | 1 | 224 |
| 224 | MixColumns方式進行加密 | 1 | 225 |
| 225 | 加密迴圈次數減一 | 迴圈未完成  加密迴圈結束 | 222  226 |
| 226 | AddRoundKey方式進行加密 | 1 | 220 |

資料解密的Grafcet離散事件模型如圖15，表8說明狀態轉移與執行動作。狀態231進行變數的設定，接著將欲解密的資料針對每個字執行數字置換的演算法做處理，如狀態232至狀態234，直到狀態235解密迴圈確認每個字都執行完畢，最後於狀態236進行DeSubBytes。DeSubBytes、DeShiftRows以及DeMixColumns的Grafcet離散事件建模，均列於附錄當中。



圖15 資料解密的Grafcet模型

表8 資料解密的Grafcet狀態轉移與執行動作

|  |  |  |  |
| --- | --- | --- | --- |
| 現在狀態 | 執行動作 | 轉移條件 | 下個狀態 |
| 230 | -- | 1 | 231 |
| 231 | 變數設定 | 1 | 232 |
| 232 | DeShiftRows方式進行解密 | 1 | 233 |
| 233 | DeSubBytes方式進行解密 | 1 | 234 |
| 234 | DeMixColumns方式進行解密 | 1 | 235 |
| 235 | 解密迴圈次數減一 | 迴圈未完成  解密迴圈結束 | 232  236 |
| 236 | DeSubBytes方式進行解密 | 1 | 230 |

3 檔案系統模組設計

檔案系統是一種儲存與組織電腦資料的方法，能夠使得對於存取和尋找資料變得容易。檔案配置表(File Allocation Table，FAT)在1977年由微軟為了管理磁碟而發明[9]，是一種應用於磁片(floppy disk)上的檔案系統，隨著時間的演進亦能支援容量越來越大的儲存裝置。主要的概念為，將儲存裝置上一個或數個連續磁區(sector)集合成一個叢集(cluster)，叢集即為檔案系統的最小單位，每個檔案都是由數個資料叢集(data cluster)串聯在一起，而檔案配置表即用來管理所有的資料叢集。

檔案配置表中每個欄位(entry)的值就是此叢集的下一個叢集代號，若為0表示是可用叢集，以圖16為例，當一個檔案的第一個叢集是a，則檔案配置表的第a個欄位，以FAT[a]表示，其值就代表下一個叢集代號b，由FAT[b]則可得到下一個叢集c，循序下去直到叢集鏈結結尾(End of Cluster Chain,EOC)便可讀取到整個檔案的資料。



圖16 檔案配置表

目前依支援的儲存空間大小分為三種型態：FAT12、FAT16、FAT32。其差異為，FAT12的檔案配置表欄位為12位元、FAT16為16位元、FAT32則為32位元。隨著位元數的增加，檔案配置表能涵蓋的範圍也就越大，所能支援的儲存裝置空間也越大。其缺點為，長期使用後會使檔案資料變得逐漸分散，而降低了讀寫速度。[磁碟重組](http://zh.wikipedia.org/w/index.php?title=%E7%A2%8E%E7%89%87%E6%95%B4%E7%90%86&action=edit&redlink=1)是一種解決方法，但並不適用於小型嵌入式裝置的使用。

檔案配置文件系統(File Allocation Table File System，FatFs)是基於檔案配置表，為了使得嵌入式開發人員能夠輕鬆且快速整合主要的作業系統至任何終端設備均能夠有兼容的操作性，並且亦可以於無作業系統的環境下應用於小型嵌入式系統。撰寫的方式須符合ANSI C，能與輸入/輸出層分開來，硬體架構上屬獨立。可使用於低成本的控制器，如AVR、8051、ARM、Z80，或本課程使用的STM32H7。

由於檔案配置文件系統能夠完整的和輸入/輸出層分開，當底層硬碟的輸入/輸出模組不是檔案配置文件系統的一部分，則必須由用戶端提供處理及存取的功能。檔案配置文件系統能與上層的應用層及下層的輸入/輸出及時間模組切割，故與存取介面的聯繫使用上，需用到功能函數如：驅動初始化([disk\_initialize](http://elm-chan.org/fsw/ff/en/dinit.html))、取得磁碟狀態([disk\_status](http://elm-chan.org/fsw/ff/en/dstat.html))、讀取磁區([disk\_read](http://elm-chan.org/fsw/ff/en/dread.html))、寫入磁區([disk\_write](http://elm-chan.org/fsw/ff/en/dwrite.html))、控制裝置相關功能([disk\_ioctl](http://elm-chan.org/fsw/ff/en/dioctl.html))、取得當前時間([get\_fattime](http://elm-chan.org/fsw/ff/en/fattime.html))等。

檔案系統模組主要是與外部儲存媒介做溝通，於本實驗做為驅動SD CARD裝置的功能，確認是否能與記憶卡進行連結，於檔案系統產生或是搜尋我們所需要的資料表。如圖17所示，狀態41及狀態42，主要是做檔案系統的初始化以及相關變數的設定，然後進行掛載檔案系統的步驟；狀態43及狀態44則表示，當我們成功掛載檔案系統並運行於控制器上之後，給予檔案系統我們需要的檔名，如檔案系統無該檔名的檔案，則新增，若存在，則開啟該文件；狀態45則為成功開啟該文件之後，如我們現在要做的是存取資料，則進行儲存動作，如為搜尋資料，則提供資料給上層的資料庫去進行查詢；最後，卸載檔案系統。檔案系統模組的狀態轉移說明如表9。



圖17檔案系統模組Grafcet模型

表9 檔案系統模組Grafcet狀態轉移與執行動作

|  |  |  |  |
| --- | --- | --- | --- |
| 現在狀態 | 執行動作 | 轉移條件 | 下個狀態 |
| 40 | -- | 1 | 41 |
| 41 | 檔案系統初始化 | 1 | 42 |
| 42 | 掛載檔案系統 | 是否存在文件 | 43、44 |
| 43 | 建立文件 | 1 | 45 |
| 44 | 開啟文件 | 1 | 45 |
| 45 | 讀取文件資料 | 1 | 46 |
| 46 | 卸載檔案系統 | 1 | 40 |

附錄

## SubBytes副程式



## ShiftRows副程式



## MixColumns副程式



## AddRoundKey副程式



## DeSubBytes副程式



## DeShiftRows副程式



## DeMixColumns副程式

