國立交通大學

資訊科學系

碩 士 論 文

超寬頻介質存取控制器之軟硬體協同設計

Hardware/Software Codesign of an Ultra-Wideband MAC Controller

研 究 生：吳秉南

指導教授：張瑞川　教授

中 華 民 國　九 十 四　年 六 月

超寬頻介質存取控制器之軟硬體協同設計

Hardware/Software Codesign of an Ultra-Wideband MAC Controller

研 究 生：吳秉南　　Student：Ping-Nan Wu

指導教授：張瑞川　　Advisor：Ruei-Chuan Chang

國 立 交 通 大 學

資 訊 科 學 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer and Information Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer and Information Science

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

超寬頻介質存取控制器之軟硬體協同設計

研究生：吳秉南　　指導教授：張瑞川教授

國立交通大學資訊科學所

論文摘要

超寬頻(Ultra-Wideband，UWB)是一種高傳輸速率且短距離的無線通訊技術，目標是希望以無線來取代有線，使個人爲中心的電子設備(例如：電腦、PDA、以及其他的行動電子產品)能夠透過低成本的 UWB 裝置相互連結，擺脫電纜的束縛與不便，讓生活更加便利。

本篇論文以軟硬體協同設計(Hardware/Software Codesign)的方式來設計一個以 IEEE 802.15.3 MAC 爲基礎的 UWB 介質存取控制器。我們以軟體爲主來實作 UWB 介質存取控制器的好處是：軟體實作可以縮短系統開發的時間，而且版本更新上也比較容易。但是也無法完全由軟體來實作，這是因爲封包的接收，還是必須透過 Interrupt 事件得知，所以至少要設計會發 Interrupt 機制的硬體。因此我們採用了軟硬體協同設計的方式來設計。

我們以軟體爲主來設計 UWB 介質存取控制器所遭遇到的主要困難是 Superframe Timing 和 TDMA Timing 準確性的問題。因此本篇論文的貢獻是如何讓軟體爲主的設計近似於純硬體設計之 Superframe Timing 和 TDMA Timing。除了 Timing 的問題，我們也設計了一套記憶體管理的方式，來加速封包的收送，以滿足 UWB 高速傳輸的特性。

Hardware/Software Codesign of an Ultra-Wideband MAC Controller

Student: Ping-Nan Wu Advisor: Prof. Ruei-Chuan Chang

Institute of Computer and Information Science
National Chiao-Tung University

Abstract

Ultra-Wideband (UWB) is a wireless technology features high transfer speed and short distance range communication. UWB aims personal devices (i.e. PC, PDA, and others) at wire-replacement. By connecting with UWB, personal devices becomes convenient and easy to use.

In thus paper, we design an media access control unit (MAC) of UWB based on IEEE 802.15.3 specification by hardware/software co-design. The merit of software implementation comes from shorter design time and easier version update. However, software implementation is lack of interrupt scheme handles packet receiving. We still need some hardware implementation and thus we adopt hardware/software co-design.

The timing accuracy of superframe and TDMA is the main problems of UWB MAC design by software implementation. We propose a software scheme that approaches hardware timing accuracy of superframe and TDMA. In addition to timing problem, a memory management scheme used to accelerate packet sending and receiving efficiency in order to achieve high transfer speed feature of UWB.

誌謝

終於完成這篇論文，首先要感謝我的指導教授張瑞川博士這兩年以來給我的指導與鼓勵，不但提供完善的資源讓我學習與實作，因此讓我在作業系統的領域中獲益良多。除此之外，也要感謝張大緯博士給我許多建議與幫忙，讓我對超寬頻介質存取控制器的設計又更深入的了解與認識。再者，我要感謝實驗室每一位同學這兩年以來的照顧與提攜，在這實驗室裡不僅讓我學習到了做學問的方法以及對專業領域深入的研究，更讓我交到了一堆志同道合的好朋友，這是我這段時間除了學問外所獲得的最大資產。再者，我要感謝女友盈君的包容與鼓勵，讓我在低潮時能夠快速恢復精神。最後更要感謝我的家人給我精神上與經濟上的支持，使我能夠順利完成碩士的生涯。

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

第1章　序論

近年來，隨著科技的進步，無線網路相關技術呈現大幅度的發展，再加上無線網路相關設備的成本越來越低廉且其傳輸的速率越來越快，因此人們使用無線網路來傳輸數位資料已經成爲一種趨勢。

無線區域網路(Wireless Local Area Network，WLAN)可以讓使用者使用無線的方式連結網際網路，擺脫傳統有線上網的束縛與不便，WLAN 傳輸範圍大約 100 公尺，可以涵蓋一般企業、學校或者家庭等環境。相關於 WLAN 的無線通訊技術有 802.11a 、802.11b 和 802.11g 等等。而無線個人網路(Wireless Personal Area Network，WPAN)可以讓以個人爲中心之電子設備進行短距離的無線連結，擺脫電纜的束縛與不便，WPAN 傳輸範圍大約 10 公尺，可用於一般室內的各種電子產品。相關於 WPAN 的無線通訊技術有藍芽(Bluetooth)和超寬頻(Ultra-Wideband，UWB)等等。

本篇論文以軟硬體協同設計(Hardware/Software Codesign)的方式來設計一個以 IEEE 802.15.3 MAC 爲基礎的 UWB 介質存取控制器。我們提出了以軟體爲主來實作 UWB 介質存取控制器的好處，並且也解釋其無法完全由軟體來實作的理由，因此我們採用了軟硬體協同設計的方式來設計。我們以軟體爲主來設計 UWB 介質存取控制器所遭遇到的主要困難是 Superframe Timing 和 TDMA Timing 準確性的問題。因此本篇論文的貢獻是如何讓軟體爲主的設計近似於純硬體設計之 Superframe Timing 和 TDMA Timing。

1.1　動機

爲了解決以個人爲中心之電子設備(例如：電腦、投影機、數位電視等等)相互連結所需的高傳輸速率之需求，IEEE WPAN[8]工作小組中的 802.15.3 次工作小組開始著手制訂標準，針對多媒體資料的傳輸制訂了高傳輸速率(11~55Mpbs)、低功率、低成本的 IEEE 802.15.3[6]標準草案，其中包含了 MAC 層與 PHY 層。

除此之外，IEEE WPAN 工作小組還成立了 802.15.3a 次工作小組，其目的是爲了制訂更高傳輸速率的 PHY 層。因此多頻帶 OFDM 聯盟 (Multi-Band OFDM Alliance，MBOA)[10]向 802.15.3a 次工作小組提出了以 UWB 技術爲基礎之高傳輸速率 PHY 層提案，其最高速率可達 480Mbps。MBOA 希望藉由 UWB 技術來實現 Wireless USB 和 Wireless 1394，讓個人爲中心之電子設備進行短距離的高速率無線連結，擺脫電纜的束縛。

本篇論文在設計一個以 IEEE 802.15.3 MAC 爲基礎的 UWB 介質存取控制器。爲了能夠與 MBOA 之 UWB Baseband 作溝通，我們硬體設計一些暫存器，如 MAC Switch Registers。因此，UWB 介質存取控制器能夠與 MBOA MAC-PHY 與 MBOA-PHY 做溝通。

實作 UWB 介質存取控制器有兩種設計方式，一是以純硬體設計，二是軟硬體協同設計。**錯誤! 找不到參照來源。**爲兩種設計方式的優缺點比較。而我們希望能夠快速地開發 UWB 介質存取控制器，再加上 802.15.3 次工作小組目前核定的 IEEE 802.15.3 爲標準草案，802.15.3a 次工作小組目前採用的高速 PHY 協定皆爲提案，所以爲了 UWB 介質存取控制器在實作之後，其功能上很容易需要被更新，未來 IEEE 802.15.3 MAC 也可能被新增一些功能，其亦需要被更新。因此，由**錯誤! 找不到參照來源。**分析，軟硬體協同設計之硬體開發時間較短，而且版本更新較容易，故我們採用軟硬體協同設計來設計 UWB 介質存取控制器。

表 1　純硬體設計 vs.軟硬體協同設計方式　優缺點比較

MAC 設計	優點	缺點
純硬體	✣ 不需要使用微處理器 ✣ 處理速度較快	✣ 硬體較複雜，成本較高 ✣ 硬體開發時間較長 ✣ 版本更新較不容易
軟硬體	✣ 硬體較簡單，成本較低 ✣ 硬體開發時間較短 ✣ 版本更新較容易	✣ 需要使用處理器 ✣ 處理速度較慢

軟體實作可以縮短系統開發的時間，而且版本更新上也比較容易。但是我們無法完全由軟體來實作 UWB 介質存取控制器，這是因爲封包的接收，還是必須透過 Interrupt 事件得知，所以至少要設計會發 Interrupt 機制的硬體。

我們所設計的 UWB 介質存取控制器是以 IEEE 802.15.3 MAC 爲基礎。在 IEEE 802.15.3 MAC 協定中，它提供服務品質保證(Quality of Service，QoS)，可以保證使用者在同時傳輸語音與資料時不必擔心語音或者資料會被中斷或傳輸速率變慢。而 QoS 服務是採用多時分工存取(Time Division Multiple Access，TDMA)機制來實現。

TDMA 機制是以時間爲基礎，每一個裝置在某段時間內會被分配數個時槽(time slots)。只有在這些被分配的時槽中，這個裝置才能傳送或接收資料。因此，時槽的時間排序與時間準確度非常地重要。本篇論文採用軟體爲主的方式來設計 UWB 介質存取控制器。因此，若由軟體鬧鐘來決定時槽是否可以傳送或接收資料，可能會遭遇到 Superframe Timing 或 TDMA Timing 不準確的問題，這是因爲處理器可能處理某一件工作，導致軟體鬧鐘無法立即決定時槽是否可以傳送或接收資料，導致時間不準確。針對此一問題，我們使用一些簡單的硬體鬧鐘來輔助軟體爲主的設計近似於純硬體設計之 Superframe Timing 和 TDMA Timing。因此，軟硬體協同設計爲我們設計 UWB 介質存取控制器的設計方法。

1.2 貢獻

純硬體實現 TDMA 機制很容易，但軟體實現 TDMA 機制卻很困難，但又希望能夠快速地開發 UWB 介質存取控制器，而且實作之後容易被更新。因此，本篇論文的貢獻是：如何以軟體爲主的設計方式來達成近似於純硬體設計之準確的 TDMA Timing 以滿足 UWB 的 QoS，並且還設計了一套記憶體管理的方式，來加速封包的收送，以滿足 UWB 高速傳輸的特性。

我們使用一些簡單的硬體來輔助軟體以達到準確的Superframe Timing和TDMA Timing，於是設計了Timers & Interrupts的機制。另外記憶體管理方式來決定封包所暫存

的記憶體起始位址，並且透過Ring的結構來加快存取速度，以滿足UWB之高Throughput的要求，如表 2。因此，本篇論文的強處有：(1)能夠快速地開發UWB介質存取控制器，(2)讓UWB介質存取控制器的版本容易被更新，(3)軟體爲主卻可以實現TDMA機制，滿足UWB之QoS的條件，(4)透過記憶體管理，滿足UWB之高Throughput的要求。

表 2　Timers & Interrupts 和 Buffer Management

設計	說明
Timers & Interrupts	使用數個 Hardware Timers 與 Interrupt Events 來近似於純硬體設計之準確的 Superframe Timing 和 TDMA Timing，以實現 UWB 之 QoS
Buffer Management	決定封包所暫存的記憶體起始位址，並且透過 Ring 的結構來加快存取速度，並且能夠重複使用記憶體空間，以降低要求記憶體時所帶來的延遲，以滿足 UWB 之高 Throughput 的要求

1.3　論文架構

在第二章中，我們將簡介超寬頻技術。第三章描述相關的研究，主要是簡介 802.15.3 MAC 協定。第四章描述 MAC 的設計與實現。在第五章中，我們展示模擬與結果。第六章爲結論與未來工作。

第2章　簡介超寬頻

超寬頻(Ultra-Wideband，UWB)[1-3][5][9]是一種高傳輸速率且短距離的無線通訊技術，目標是希望以無線來取代有線，使個人爲中心的電子設備(例如：電腦、PDA、以及其他的行動電子產品)能夠透過低成本的超寬頻裝置相互連結，讓生活更加便利。

美國聯邦通訊委員會(Federal Communications Commission，FCC)[11]在 2002 年 2 月規範超寬頻通訊技術的無線頻段爲 3.1~10.6GHz。超寬頻使用低功率的短脈衝波(Impulse Radio，IR)[4]，範圍大約 10 公尺，輸速率最高可達到 480Mbps。因爲其功率很低，所以不會對人體造成傷害。

超寬頻在傳輸的過程中使用短脈衝波，每一個脈衝波都非常短，從數毫微秒(nano-second，ns)至數百微微秒(pico-second，ps)，因此很難被截收。超寬頻可傳輸一般資料、聲音或影像，並且可以選擇是否在傳輸中加密。因此，超寬頻提供可靠而安全的無線通訊環境。

2.1　超寬頻簡史

超寬頻是一種使用短脈衝波的高傳輸速率且短距離無線通訊技術。其實它並不是新興的技術，早從 1960 年代開始，已經有科學家針對脈衝式微波的特性進行相關的研究，在 1980 年的時候，短脈衝波技術被應用在天線、通訊雷達等方面，在 1989 年的時候，美國國防部提出超寬頻這個專有名詞，在 1990 年的時候，美國國防部開始著手超寬頻發展計畫，主要是針對軍事上雷達系統、定位系統與通訊技術進行開發與研究。

在 2002 年 2 月，美國聯邦通訊委員會制訂關於超寬頻的規範，將其無線頻段訂爲 3.1~10.6GHz，使得超寬頻技術從軍事用途轉爲商業用途。此後，越來越多的廠商開始對超寬頻感到興趣，並著手制訂相關的超寬頻標準。

美國聯邦通訊委員會對於超寬頻的定義如下：(1)中心頻率大於 2.5GHz時，其傳輸頻段的頻寬大於等於 500MHz，或(2)中心頻率小於等於 2.5GHz，其傳輸頻段所佔用的頻寬比(Fractional Bandwidth，FB)大於等於 0.20[1]。如圖 1所示。

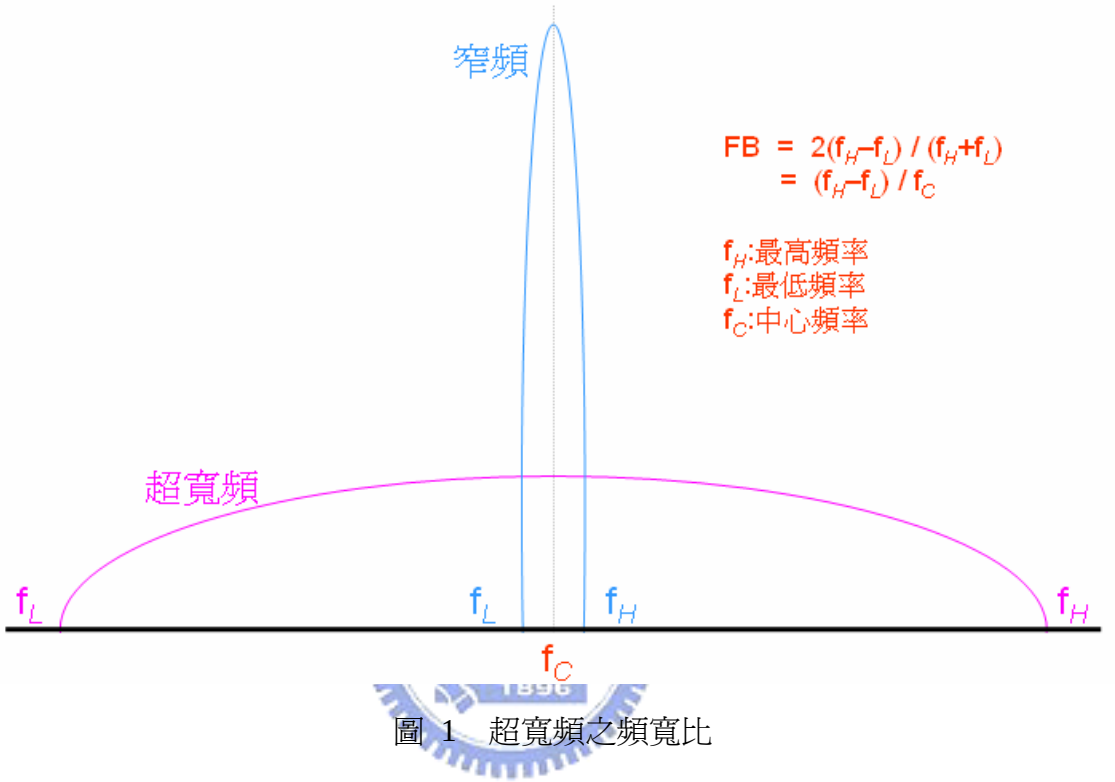

圖 1　超寬頻之頻寬比

2.2　超寬頻特性

超寬頻使用短脈衝波的方式進行傳輸，而不採用傳統無線通訊所使用的載波進行傳輸，因此超寬頻具有極低的功率消耗、超高的傳輸速率、抗多途徑的干擾等特性。

所謂的脈衝波，即是訊號以毫微秒(ns)寬的脈衝在幾個毫秒(milli-second，ms)區間內傳送，而傳統無線通訊使用載波來傳輸訊號，因爲載波是連續的正弦波，因此一直有功率消耗，但是超寬頻使用短脈衝波的方式來進行傳輸，故只有在脈衝波發射的時候，才有功率的消耗，因此超寬頻具有極低的功率消耗的特性。

[1]一般的窄頻或展頻通訊技術(例如：802.11 a/b/g)，其頻寬比小於 0.01。

根據 Shannon 的最大頻道容量(Maximum Channel Capacity)公式[12]：傳輸速率將隨著使用頻段的頻寬增加而呈線性增加。因此，理論上使用頻段的頻寬越大的時候，則傳輸速率也就越大，也就是因爲如此，超寬頻傳輸速率可達到 480Mbps，因此超寬頻具有超高的傳輸速率的特性。

$$C = B \times \log_2(1+S/N)$$

C:最大頻道容量(bits/sec)

B:頻道頻寬(Hz)

S:訊號功率(Watts)

N:雜訊功率(Watts)

傳統無線通訊使用載波進行傳輸容易造成多途徑干擾，這是因爲相同的載波訊號會因爲附近物體的反射而產生兩個到多個不同的路徑，所以反射訊號不會同時抵達接收器，有時訊號幾乎還會彼此抵消。但是超寬頻使用短脈衝來進行傳輸，其接收器就可能解析出不同的多路徑訊號流，因此超寬頻具有抗多途徑的干擾的特性。

2.3 超寬頻空間容量(Spatial Capacity)

空間容量(Spatial Capacity)[2]是由Intel實驗室所提出的，是判斷短距離無線通訊系統的運作效能之指標。所謂的空間容量是以每平方公尺每秒的位元數作爲單位。由圖 2可以知道 802.11b、藍芽、802.11a與超寬頻的空間容量之比較，因此超寬頻的空間容量遠大於其他短距離無線通訊系統，因此超寬頻非常適合在無線個人區域網路中進行高速資料傳輸。

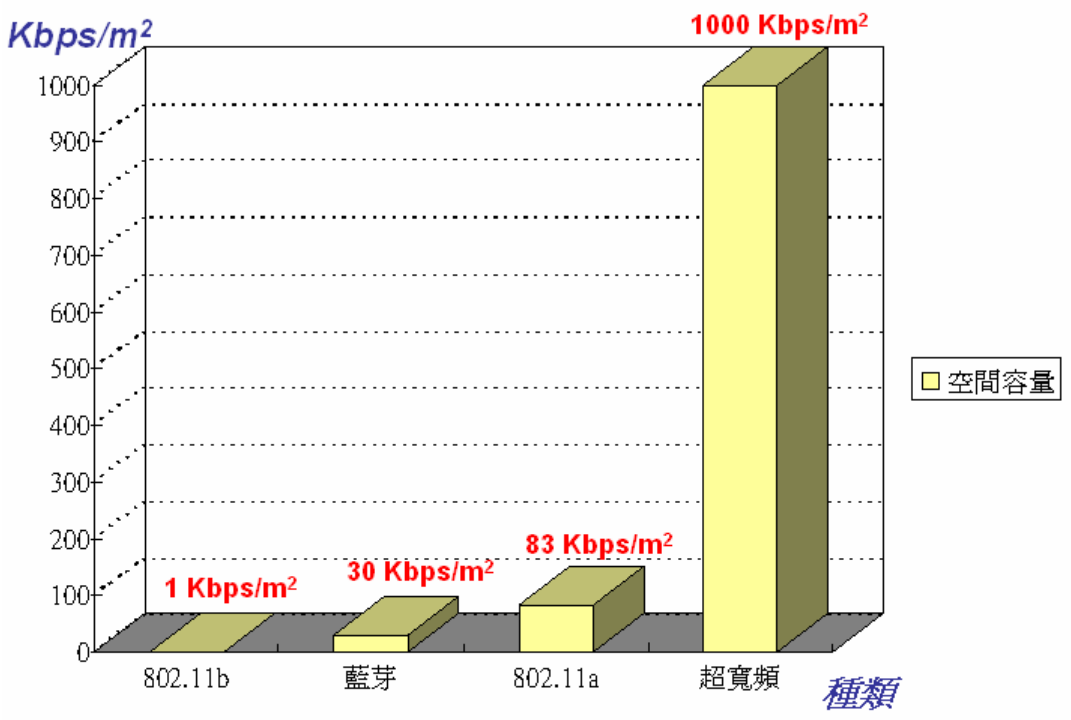

圖 2　各種通訊技術空間容量之比較

2.4　超寬頻目前進展

在 2002 年 2 月的時候，美國聯邦通訊委員會制訂關於超寬頻的規範，使得超寬頻技術從軍事用途轉爲商業用途，因此 IEEE 成立 802.15.3a 次工作小組，著手制訂有關超寬頻無線連結的規格與標準。到了 2003 年底，802.15.3a 次工作小組討論主要的超寬頻規格標準有兩大版本：(1)以 Motorola、Freescale[7]爲首之 UWB Forum 所提出的直接序列超寬頻(Direct Sequence Ultra-Wideband，DS-UWB)草案，和(2)以 Intel、TI 爲首之 Multi-Band OFDM Alliance(MBOA)[10]所力推的多頻帶正交頻分多工技術(Multi-Band Orthogonal Frequency Division Multiplexing，MB-OFDM)。

第3章　相關研究

IEEE WPAN 工作小組中的 802.15.3 次工作小組制訂了高傳輸速率(11~55Mpbs)、低功率、低成本的 IEEE 802.15.3。除此之外，它還提供了 QoS，可以保證使用者在同時傳輸語音與資料時不必擔心語音或者資料會被中斷或傳輸速率變慢，因而滿足了行動消費性的影音多媒體應用。這些特性讓 IEEE 802.15.3 適用於 UWB。

但是 IEEE 802.15.3 裡制訂的 PHY 協定傳輸速率最高只達 55Mbps，因此，IEEE WPAN 工作小組還成立了 802.15.3a 次工作小組，其目的是爲了制訂更高傳輸速率的 PHY 層。MBOA 向 802.15.3a 次工作小組提出了以 UWB 技術爲基礎之高傳輸速率 PHY 層提案，其最高速率可達 480Mbps。本章將描述相關的研究，主要是簡介 802.15.3 MAC 協定。

3.1　IEEE 802.15.3 MAC

IEEE 802.15.3 MAC是一種以TDMA爲基礎的MAC，並適用於高傳輸速率的WPAN。每一個IEEE 802.15.3 裝置(Device，DEV)在一個小區域內可以相互傳輸資料，此區域稱之爲piconet。一個piconet是由 1 個到多個裝置所組成，且其中一個裝置另有一個特殊身份，以協調和分配整個piconet的資源，擁有此特殊身份的裝置亦稱PNC(PicoNet Coordinator)。有此可知，IEEE 802.15.3 piconet屬於中控式的協調網路，如圖 3所示。

PNC 提供基本的網路時序同步(network timing synchronization)、服務品質保證(Quality of Service)、省電模式(power saving)、配置時槽資訊(timing allocation information)等功能。除此之外，每一個 piconet 是一個隨建即連的網路(ad-hoc network)，點對點(peer-to-peer)方式來傳輸資料。

IEEE 802.15.3 MAC的時序是以superframe爲架構，如圖 4所示。每一個superframe可以切割成三個部分：Beacon、CAP(Contention Access Period)、CTAP(Channel Time

Allocation Period)。緊接著下面幾小節將詳細介紹每一個部分。

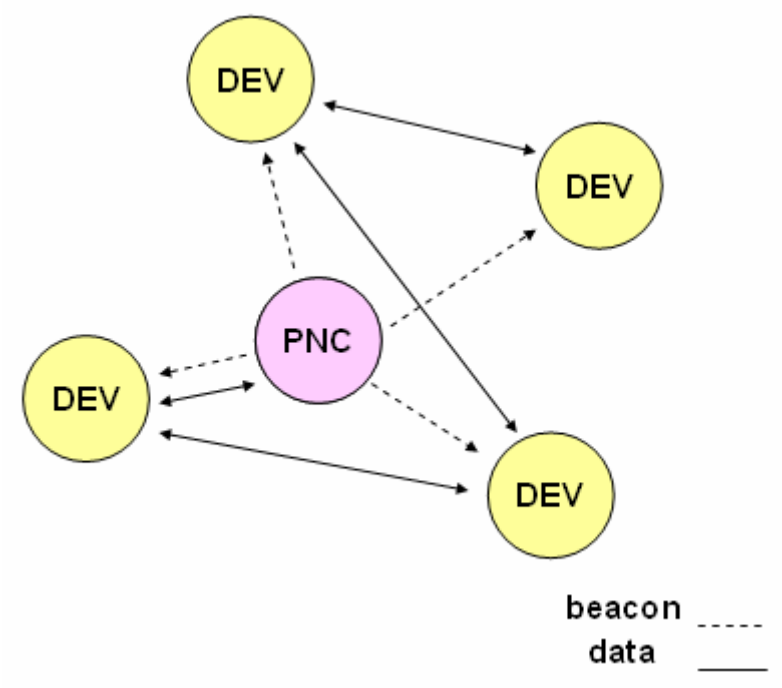

圖 3　IEEE 802.15.3 piconet

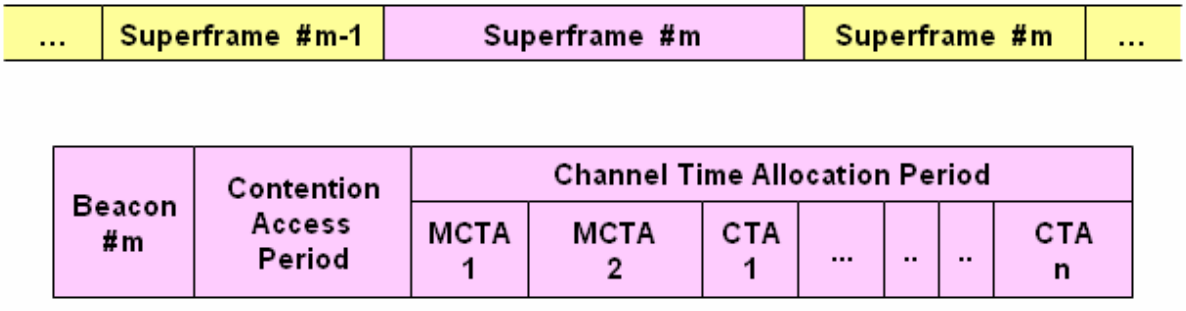

圖 4　IEEE 802.15.3 superframe

3.1.1 Beacon

Beacon 是由 PNC 在每一個 superframe 開始時所發送出去的，以廣播方式傳遞給所有屬於此 PNC 所管的 DEVs。Beacon 的功能爲網路時序同步、省電裝置資訊、配置時槽資訊和 piconet 之間互相溝通的命令。

3.1.2 CAP (Contention Access Period)

接著在 Beacon 之後的是 CAP。CAP 可以用來傳輸命令封包與 asynchronous 資料封包。CAP 採用 CSMA/CA 的機制，因此所有的 DEVs 在此區間，會以競爭的方式，來取得傳送封包的機會。

3.1.3 CTAP (Channel Time Allocation Period)

CTAP 是 superframe 的最後一個部分，它採用 TDMA 的機制，因此 IEEE 802.15.3 分割 CTAP 成許多的時槽，稱之 CTAs(Channel Time Allocation)或 MCTA(Management CTAs)。MCAT 屬於 CTA 的一種，但它被獨立出來給 PNC 對其他 DEVs 作爲傳輸命令封包的時槽。如果 CTAP 內的某 CTA 被 PNC 分配給某 DEV，則此 DEV 可以在此 CTA 傳輸資料或接收資料。DEV 如果要傳輸大量資料或串流資料，此 DEV 必須在 CAP 內傳送要求命令封包給 PNC，PNC 允許之後，DEV 才會被配置一個 CTA，此 CTA 內可傳輸大量資料或串流資料。

第4章　MAC 設計與實現

4.1　系統架構

在我們所設計的UWB介質存取控制器中，其軟體與硬體分層的架構圖如圖 5。包含了 802.15.3 次工作小組之IEEE 802.15.3 MAC與 802.15.3a次工作小組之MBOA所提的高速PHY草案。因此，我們還設計了MAC Switch，目的是要讓IEEE 802.15.3 MAC可以操作並控制MBOA PHY。

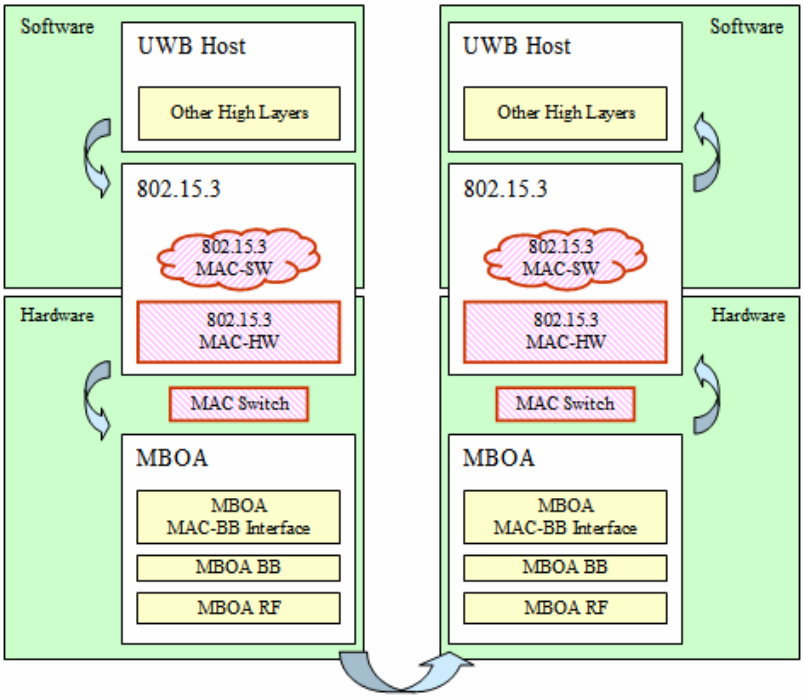

圖 5　UWB 系統架構圖

在我們的實作上，爲了要快速實現與驗證 802.15.3 MAC功能的正確性，我們使用CoWare公司所提供的ConvergenSC Tools來開發我們的UWB介質存取控制器。ConvergenSC Tools內含有SystemC Compiler，可以模擬與驗證我們的MAC-HW和MAC

Switch，並且透過ConvergenSC IPs內的ARM Processor、AMBA等IP libraries，我們可以快速地建構出ARM-based Platform的模擬環境如圖 6所示。ConvergenSC Tools可以讓我們模擬兩個ARM-based Platform，每一個Platform跑屬於自己的MAC-SW，並讓其MAC-HW對接。因此，兩個UWB系統相互傳送封包，因而驗證 802.15.3 MAC功能的正確性。

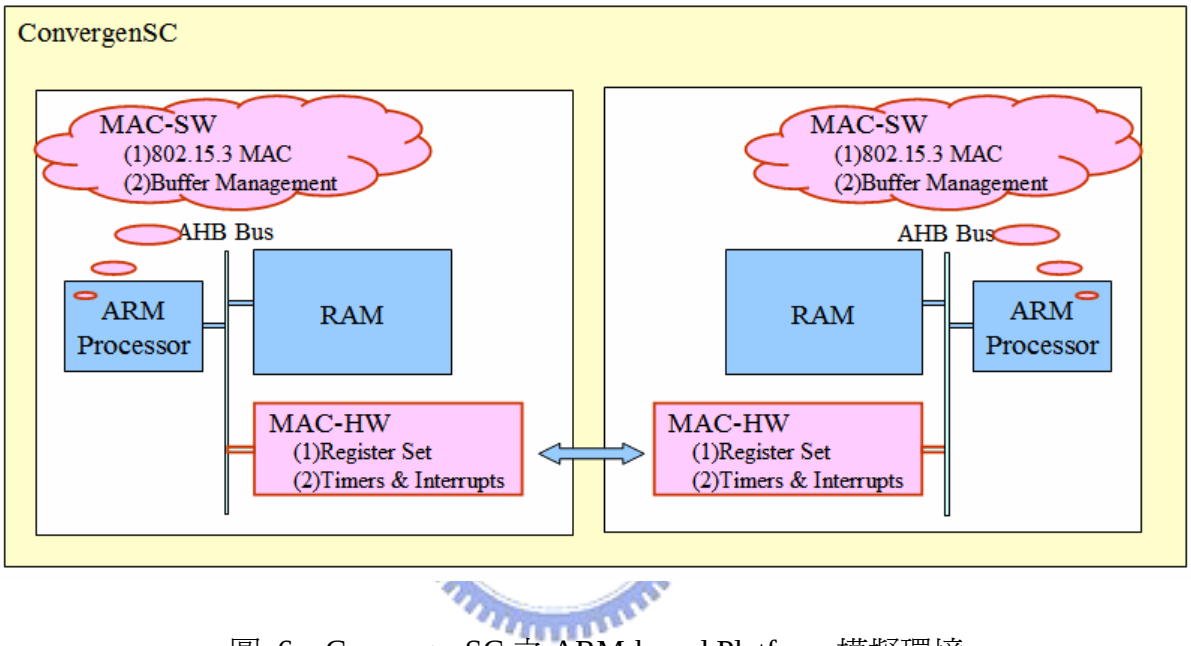

圖 6　ConvergenSC 之 ARM-based Platform 模擬環境

在圖 6當中，我們所要實現的部分爲MAC-SW、MAC-HW(內包含MAC Switch)，如圖 6所示。MAC-SW這部分佔了我們實作的大部分，包含了 802.15.3 MAC和Buffer Management機制。Buffer Management機制目的：決定收送封包的記憶體起始位址，並且重複使用記憶體，以滿足UWB的高速傳輸的要求。MAC-HW爲簡單的硬體架構，包含了 802.15.3 MAC與 802.15.3a PHY相關之Register Set和Timers & Interrupts機制。Timers & Interrupts機制目的是要讓軟體爲主的設計達到近似於純硬體設計的Superframe Timing和TDMA Timing。

首先，表 3列出我們UWB介質存取控制器中實作了 802.15.3 MAC的功能。接著後面幾小節，我們會說明Register Set、Timers & Interrupts機制和Buffer Management機制的一些設計理念。

表 3　802.15.3 MAC 實作部分

功能	802.15.3	my prototype
Scan channels	○	○
Start / Stop a piconet	○	○
PNC handover	○	○
Association / Disassociation	○	○
Stream Management	○	○
Fragmentation / Defragmentation	○	○
Dynamic piconet parameters	○	○
Multiple rate	○	○
Power management	○	○
Security	○	X

4.2 Register Set

我們在設計 UWB 介質存取控制器時，我們在 MAC-HW 內也設計類似 DMA Controller 的功能，讓 MAC-HW 能透過 AHB 直接把封包從 Memory 送出或接收封包到 Memory 中，降低複製封包時的 overhead，目的是讓以軟體爲主的設計近似於純硬體設計的 Timing。

因此，我們在規劃Register Set時，把Register Set分類如圖 7，包含了TX Buffer、RX Buffer1、RX Buffer2、Timers & Interrupts、802.15.3 MAC和 MAC Switch。其中MAC Switch主要是要操作與控制MBOA-PHY。下面幾小節將會對每一類Register作一些說明。

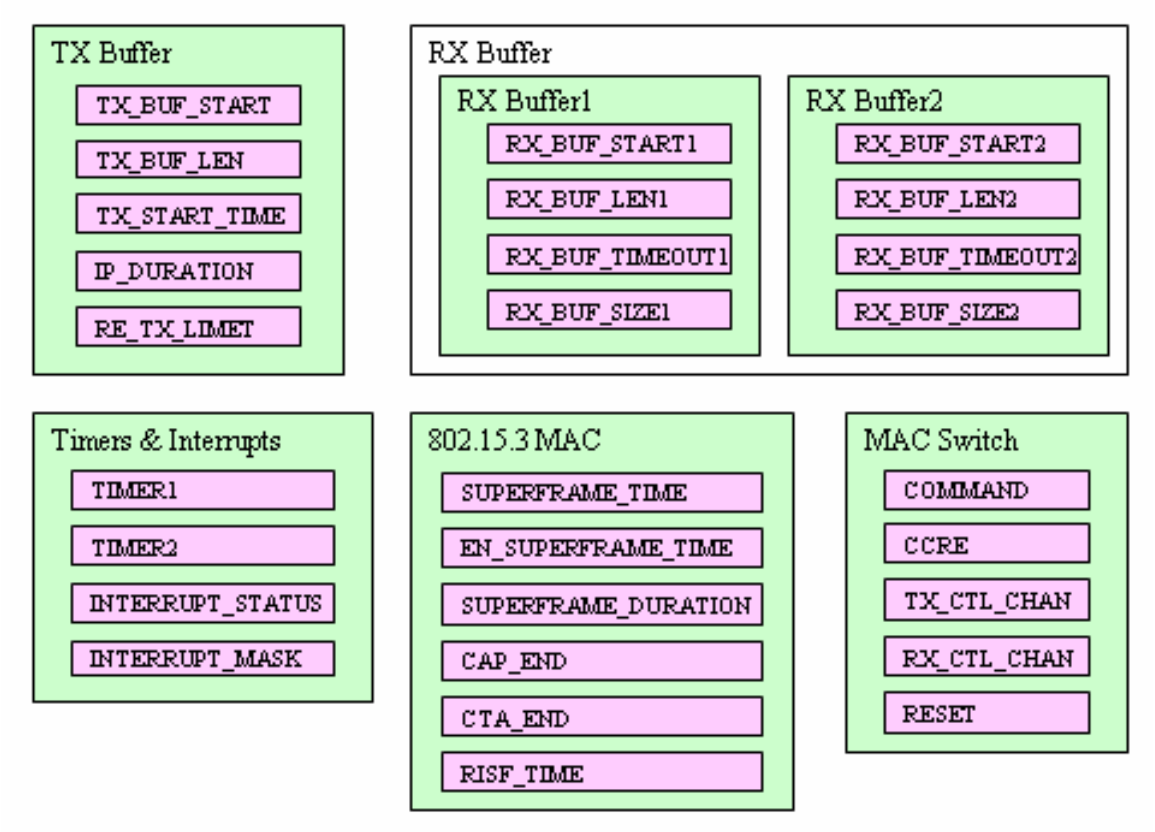

圖 7　Register Set

4.2.1 TX Buffer

TX Buffer暫存器包含TX_BUF_START、TX_BUF_LEN、TX_START_TIME、IP_DURATION和RE_TX_LIMIT，如圖 7。

在每一個 CTA(i.e., TDMA 時槽) 開始之前，軟體先設定好欲傳送出去的封包所在記憶體之起始位址 (TX_BUF_START) 和欲傳送封包之總 byte 數 (TX_BUF_LEN) 。接著設定封包與封包之前的等待時間 (IP_DURATION) 和每一個封包最多可重傳的次數 (RE_TX_LIMIT) 。最後才設定封包傳送的起始時間 (TX_START_TIME) 以告訴硬體何時要把該封包傳送出去。

當傳送的起始時間等於 Superframe 目前時間，硬體便開始從 Memory 把封包經由

MAC-HW 直接傳送出去，直到要求欲傳送封包傳送完畢。如果傳送的封包要求 Immediate ACK 時，每個封包之間會等待 SIFS (Short Interframe Spacing)，目的是判斷是否有收到 Immediate ACK。若沒收到時，硬體會重傳該封包，直到最多重傳次數到達。如果傳送的封包沒有要求 Immediate ACK 時，每個封包之間會等待 MIFS (Minimum Interframe Spacing)。

4.2.2 RX Buffer1 和 RX Buffer2

RX Buffer1 暫存器包含RX_BUF_START1、RX_BUF_LEN1、RX_BUF_TIMEOUT1 和RX_BUF_SIZE1。RX Buffer2 暫存器包含RX_BUF_START2、RX_BUF_LEN2、RX_BUF_TIMEOUT2 和RX_BUF_SIZE2，如圖 7。

軟體首先會設定好 RX Buffer 1 在 Memory 的起始位址(RX_BUF_START1)，接著設定 RX Buffer 1 最多可暫存的 byte 數(RX_BUF_SIZE1)。硬體知道目前使用 RX Buffer 是 Buffer1 或 Buffer2，假設目前 RX Buffer 爲 RX Buffer1，每當硬體收封包之前，會先判斷此 RX Buffer 1 接收該封包是否以達 RX Buffer 1 飽和度(RX_BUF_SIZE1)，若還沒裝滿，則收封包，若裝滿了，收完該封包之後，硬體會切換至 RX Buffer 2，並且透過中斷讓軟體解析滿載之 Buffer 內的封包。

若 RX Buffer 1 沒有滿載，但裡頭確有封包，過一段時間(RX_BUF_TIMEOUT1)之後，硬體會切換至 RX Buffer 2，並且透過中斷讓軟體分析滿載之 Buffer 內的封包。因此可以保證所有接受的封包可以被立即解析。RX Buffer 2 運作機制與 RX Buffer 1 雷同。

最後，RX_BUF_LEN1 和 RX_BUF_LEN2 是硬體中斷之後，軟體會去讀此暫存器，以得知目前硬體接收到封包的總 bytes 數。

4.2.3 Timers & Interrupts

Timers & Interrupts暫存器包含TIMER1、TIMER2、INTERRUPT_STATUS和

INTERRUPT_MASK，如圖 7。

我們的系統提供 2 個硬體鬧鐘。TIMER1 和 TIMER2 以 us 爲單位，軟體設定好 Timer 的時間以後，每 1us 會遞減 1，直到 Timer 暫存器的值等於 0，於是發生了 Timeout 中斷事件。透過 TIMER1 和 TIMER2 倒數產生中斷，讓以軟體爲主的設計有了高解析度的時鐘，來達到近似於純硬體設計的 Superframe Timing 和 TDMA Timing。4.3 節會詳細介紹 Timers & Interrupts 機制是如何運作之。

除了硬體Timer會產生中斷之外，在收送封包的期間也會有中斷的產生，軟體可以透過寫入INTERRUPTE_MASK暫存器來選擇自己所要接收的中斷事件種類，而中斷事件的內容可以透過讀取INTERRUPT_STATUS知道。表 4表示INTERRUPT_STATUS對應到何種中斷事件。

表 4　INTERRUPT_STATUS 對應的中斷事件

Bit	事件名稱	說明
0~15	number of entries are filled	表示 RX Buffer1 或 RX Buffer2 內所暫存封包的個數 (必須 Bit 30/31 被設成 enable)
16	Timer1 expires	硬體 Timer1 倒數至 0
17	Timer2 expires	硬體 Timer2 倒數至 0
18	Retransmission limit has been reached	送出封包已達最大重傳次數
19	TX is completed	欲傳送封包以全部傳送完畢
20	Beacon reached	硬體收到 Beacon 封包
21	Equal Superframe duration	SUPERFRAME_TIME 等於 SUPERFRAME_DURATION
22	Equal CAP end time	SUPERFRAME_TIME 等於 CAP_END
23	Equal CTA end time	SUPERFRAME_TIME 等於 CTA_END

24	CCA busy now	目前 Channel 爲 Busy 狀態
25	CCA clean now	目前 Channel 爲 Idle 狀態
30	RX buffer1 is ready	RX Buffer1 裝滿了或 timeout
31	RX buffer2 is ready	RX Buffer2 裝滿了或 timeout

4.2.4 802.15.3 MAC

802.15.3 MAC暫存器包含SUPERFRAME_TIME、EN_SUPERFRAME_TIME、SUPERFRAME_DURATION、CAP_END、CTA_END和RIFS_TIME如圖 7。

SUPERFRAME_TIME 表示每 Superframe 目前的時間，以 us 爲單位。每 Superframe 開始時其值爲 0，每經過 1us 值加 1，直到等於 Superframe duration 後，SUPERFRAME_TIME 值更新爲 0，這表示另一個新的 Superframe 開始。而 EN_SUPERFRAME_TIME 是 PNC 裝置爲了讓 SUPERFRAME_TIME 這個鬧鐘運作，從 0 開始，每 1us 遞增 1。而 DEV 裝置接收到 Beacon 封包後，會將 SUPERFRAME_TIME 值更新爲 0，一樣每 1us 遞增 1。CAP_END 表示 CAP 區間在 Superframe 中何時被結束，CTA_END 表示某 CTA 區間在 Superframe 中何時被結束，RIFS_TIME 表示 RIFS Duration。

4.2.5 MAC Switch

MAC Switch暫存器包含COMMAND、CCRE、TX_CTL_CHAN、RX_CTL_CHAN和RESET如圖 7。

這些暫存器主要是要控制MBOA PHY而準備的。COMMAND暫存器是控制MBOA MAC-PHY interface內的FSM狀態，如表 5**錯誤! 找不到參照來源。**描述和圖 8所示。CCRE爲CCA Request，是控制CCA機制啓動或失效。TX_CTL_CHAN暫存器是控制傳送時使用long preamble或short preamble與控制傳送的Channel編號。RX_CTL_CHAN暫存器是控制接收時使用long preamble或short preamble與控制接收的Channel編號。RESET則是

重設MAC-HW。

表 5　COMMAND Register

Bit	FSM 狀態	說明
0~15	The time for switch	倒數多少 us 後進行轉換
16	SWITCH_TO_RESET	To Reset state
17	SWITCH_TO_STANDBY	To Standby state
18	SWITCH_TO_TX	To TX state
19	SWITCH_TO_RX	To RX state
20	SWITCH_TO_SLEEP	To Sleep State

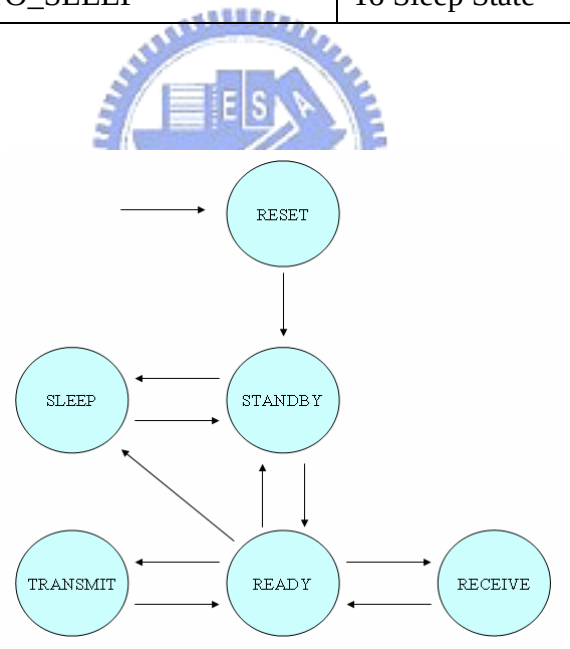

圖 8　MBOA-PHY State Diagram

4.3　Timers & Interrupts

在這一節當中，我們將介紹如何以軟體爲主且搭配簡單硬體的設計來近似於純硬體

所設計的 Superframe Timing 和 TDMA Timing。我們使用硬體 Timer 與一些有關於 Timing 的 Registers 來適時的產生 Interrupt 事件，以告知軟體下一步動作。後面幾小節，我們將解釋如何以軟體爲主設計來近似於純硬體之 Timing。

4.3.1 PNC 傳送 Beacon

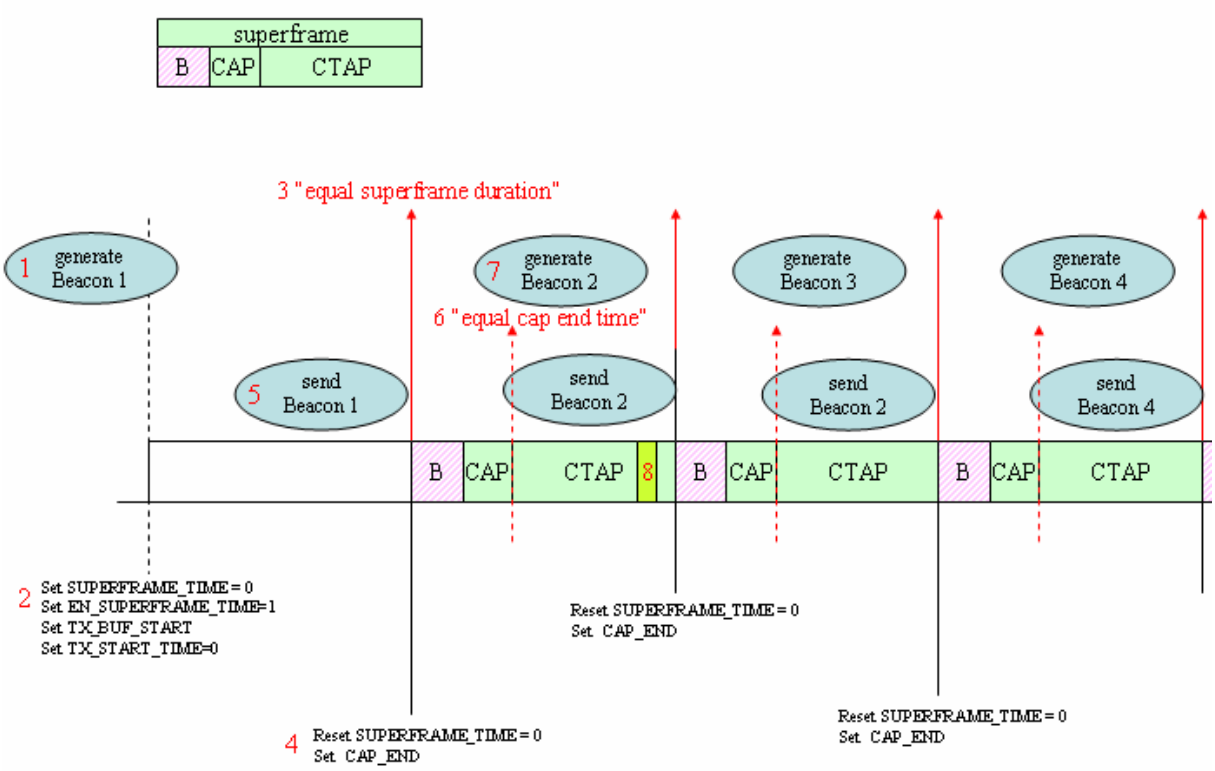

圖 9　PNC 傳送 Beacon

圖 9爲PNC傳送Beacon時序的示意圖。第 1 步驟：一旦決定PNC角色之後，便開始製作Beacon封包。第 2 步驟：製作Beacon封包完畢後，接著設定SUPUERFRAME_TIME的值爲 0，表示Superframe初始時間爲 0。然後設定EN_SUPERFRAME_TIME的值爲 1，表示SUPERFRAME_TIME開始每 1us其值遞增 1。之後再設定TX_BUF_START等於暫存Beacon Fame Buffer的起始位址。設定TX_START_TIME爲 0，表示TX_START_TIME與SUPERFRAME_TIME皆爲 0 時，開始傳送Beacon封包出去。這是爲了要讓每一個Superframe開始時都能迅速地把傳送Beacon封包，我們必須提早製作Beacon封包。因此，

在第一個Beacon封包被製作之後，會等待一個Superframe Duration的時間。

第 3 步驟：當 SUPERFRAME_TIME 等於 SUPERFRAME_DURATION 的時後，會產生 Interrupt 事件「equal Superframe duration」。第 4 步驟：此 Interrupt ISR 會重設 SUPERRAME_TIME 等於 0，並且設定目前 Superframe 內的 CAP end time 的值爲多少。第 5 步驟：因爲 SUPERFRAME_TIME 被重設爲 0，因爲 TX_START_TIME 也爲 0，因此，第一個 Beacon 就被 PNC 所傳送出去。

第 6 步驟：等到 SUPERFRAEM_TIME 等於 CAP_END，會產生 Interrupt 事件「equal cap end time」。第 7 步驟：此 Interrupt ISR 會要求 Beacon Thread 產生下一個 Superframe 要傳送出去的 Beacon 封包。第 8 步驟：等到最後一個 CAT 結束的時後，會設定 TX_BUF_START 等於暫存 Beacon Fame Buffer 的起始位址。設定 TX_START_TIME 爲 0。之後每一個 Beacon 封包的製作與傳送的步驟與 3、4、5、6、7 步驟類似。

4.3.2 DEV 接收 Beacon

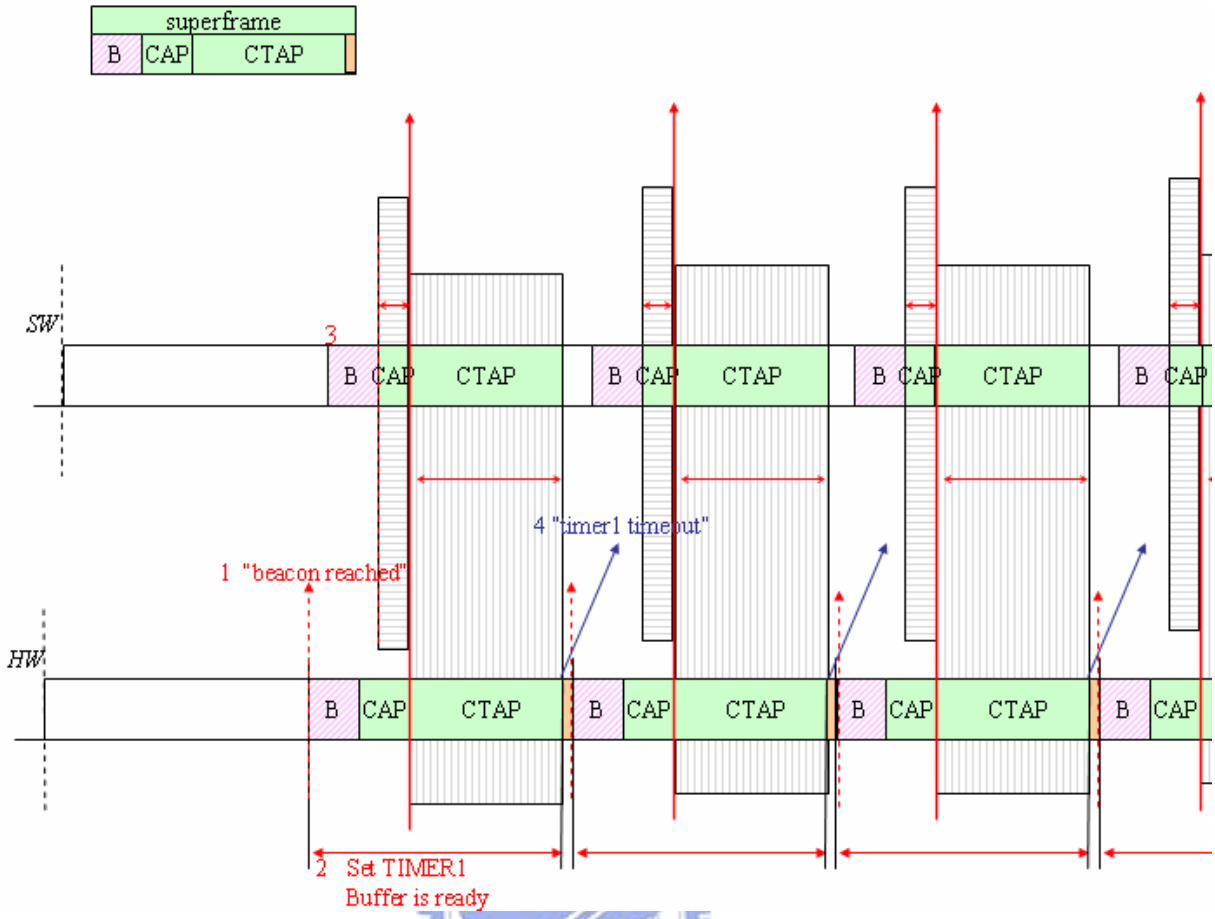

圖 10 DEV 接收 Beacon

圖 10爲DEV接收Beacon時序的示意圖。因爲硬體要通知軟體會有時差，因此在圖10中，我們把Superframe Duration分成HW與SW來看，並可以看到SW會比HW延遲一些時間。

第 1 步驟：一旦決定 DEV 角色之後，便開始接收封包到 RX Buffer1 或 RX Buffer2。若硬體接收到的封包被判定爲 Beacon 封包。第 2 步驟：設定 TIMER1 的數值，目的是讓 DEV 提早開啓接收模式，以接收下一個 Beacon 封包。接著會讓接收到 Beacon 封包的 RX Buffer 產生 Interrupt 事件「Buffer is Ready」。第 3 步驟：此 Interrupt ISR 會要求分析所有 RX buffer，並以 Beacon 封包爲優先處理。然後根據此 Beacon 封包內的 CAP End Time 資訊來設定 CAP_END 的數值。第 4 步驟：當 TIMER1 倒數到 0 時，產生 Interrupt 事件「timer1 timeout」，此 Interrupt ISR 開始設定接收模式，以接收下一個 Beacon 封包。

4.3.3 傳送與接收 Command 封包

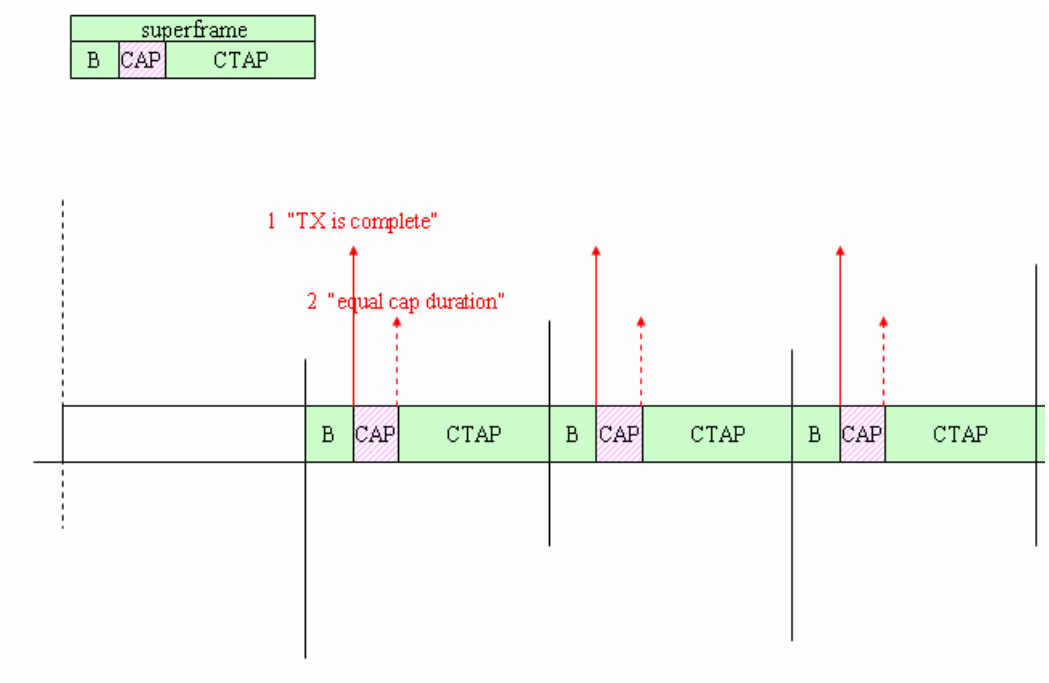

圖 11 PNC 傳送與接收 Command 封包

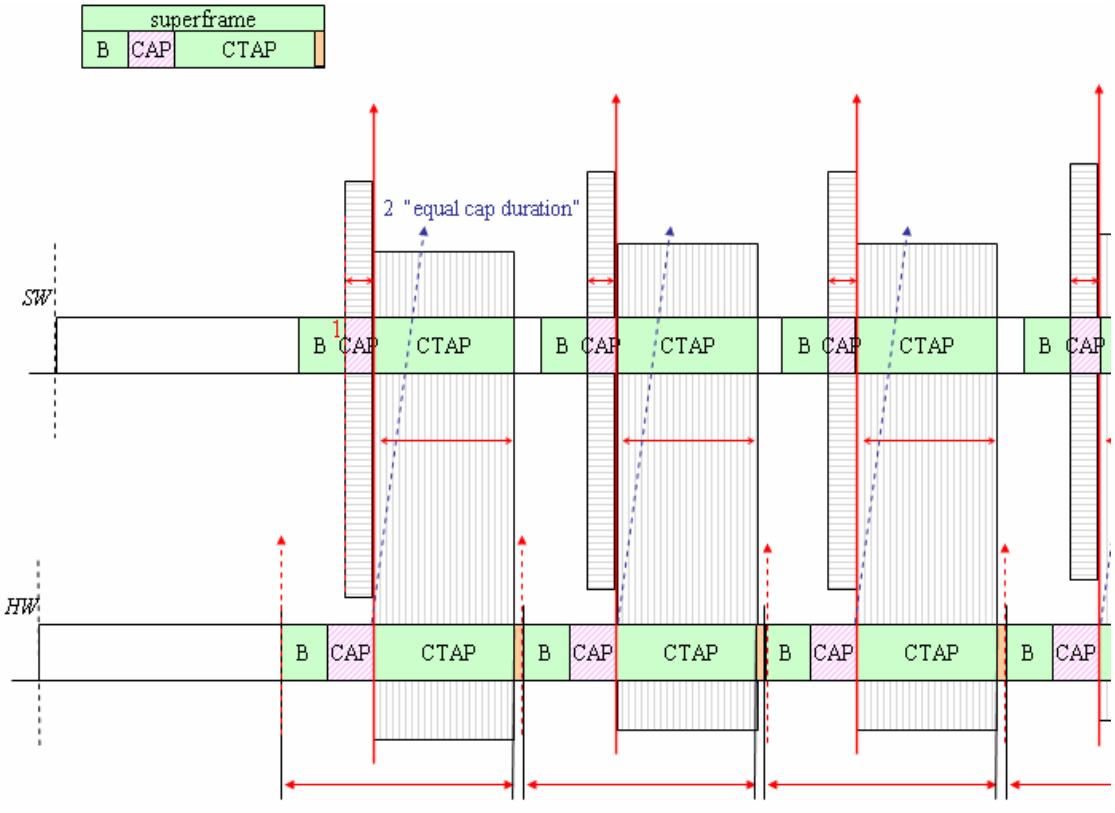

圖 12　DEV 傳送與接收 Command 封包

圖 11爲PNC傳送與接收Command時序的示意圖。圖 12爲DEV傳送與接收Command時序的示意圖。這兩張圖說明PNC與DEV開始傳送Command封包的時間點不同。

PNC 第 1 步驟：當傳送完畢 Beacon 封包的時後，會產生 Interrupt 事件「TX is Complete」。此 Interrupt ISR 會找出 Command Frame Buffer 的起始位址，並啓動 Slotted Aloha 機制。當 Channel 閒置的時後，會開始倒數，倒數至 0，便把 Command 封包送出去。當 Channel 忙碌的時後，會暫存 Slotted Aloha Count 目前的數值，直到 Channel 又閒置後，再接著倒數。第 2 步驟：因爲在傳送 Beacon 封包的時候，會更新目前傳送 Beacon 封包內的 CAP end timer 資訊到 CAP_END，當 CAP_END 等於 SUPERFRAME_TIME 的時後，會產生 Interrupt 事件「equal cap end time」。此 Interrupt ISR 便開始處理 Data 傳送與接收的動作，會在 4.3.4 節說明。

DEV 第 1 步驟：當 Beacon 封包分析完畢之後，便會找出 Command Frame Buffer 的起始位址，並啓動 Slotted Aloha 機制。當 Channel 閒置的時後，會開始倒數，倒數至 0，便把 Command 封包送出去。當 Channel 忙碌的時後，會暫存 Slotted Aloha Count 目前的數值，直到 Channel 又閒置後，再接著倒數。第 2 步驟：因爲在解析收到 Beacon 封包的時候，會更新 CAP_END，當 CAP_END 等於 SUPERFRAME_TIME 的時後，會產生 Interrupt 事件「equal cap end time」。此 Interrupt ISR 便開始處理 Data 傳送與接收的動作，會在 4.3.4 節說明。

4.3.4 傳送與接收 Data 封包

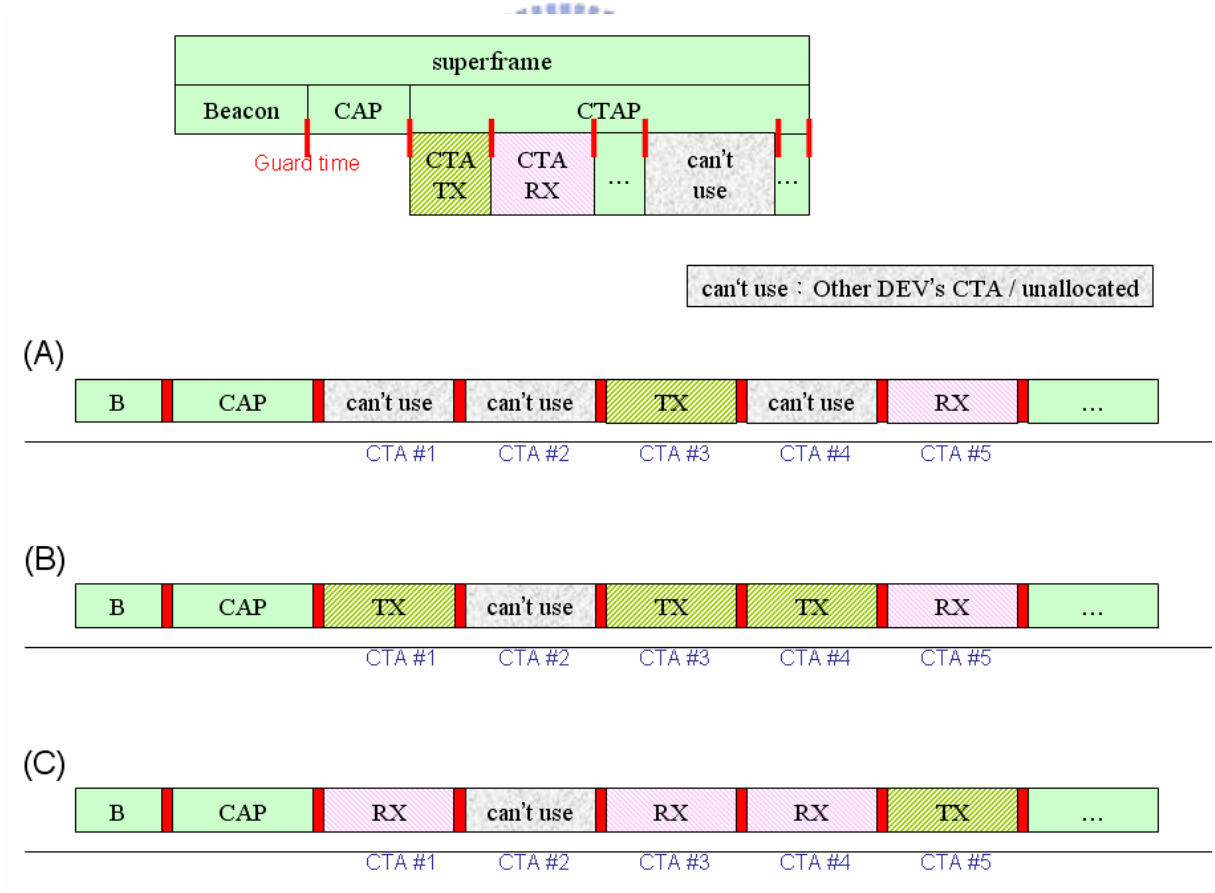

圖 13　傳送與接收 Data 封包

圖 13爲傳送與接收Data時序的示意圖。在這裡我們會利用Guard Time這段時間來找尋下一個CAT的起始位址，並設定其Start Time到Register中。一旦SUPERRAME_TIME等於Start Time，便開始傳送或接收Data封包。

在圖 13當中，我們把裝置在某CTA爲來源裝置的時後，此CTA叫做「CAT TX」，我們把裝置在某CTA爲目標裝置的時後，此CTA叫做「CTA RX」，其他區間叫做「can't use」，表示此區間無法傳送或接收封包。我們把傳送與接收Data封包分成三種Cases，如圖 13 (A)、圖 13 (B)和圖 13 (C)。

我們先來解釋圖 13 (A)。說明我們是如何利用Guard Time這段時間來找尋下一個CAT的起始位址。首先，PNC在傳送完Beacon封包之後或DEV在分析完Beacon封包之後，判斷下一個CTA爲何？因此要判斷CTA#1 是TX、RX或can't used。在圖 13 (A)中，CTA#1 等於can't used。

在 CAP 區間結束後，因爲 CTA#1 等於 can't used，所以 PHY-FSM 爲 Ready 或 Sleep，即不傳送也不接收，甚至可以把天線關閉。當 CTA#1 結束之後，會產生 Interrupt 事件「equal cta end time」。此 Interrupt ISR 會判斷下一個 CTA 爲何？CTA#2 等於 can't used，因此動作跟 CTA#1 一樣，PHY-FSM 爲 Ready 或 Sleep。

當 CTA#2 結束之後，會產生 Interrupt 事件「equal cta end time」。此 Interrupt ISR 會判斷下一個 CTA 爲何？CTA#3 等於 TX。之後便設定 TX_BUF_START 等於 CTA#3 所管的 Data Frame Buffer 的起始位址，還有設定 TX_BUF_LEN 等於傳送 Data 的總 bytes 數，最後設定 TX_START_TIME 爲 CTA#3 的 Start Time。當 SUPERFRAME_TIME 等於 TX_START_TIME 的時後，軟體會下達指令，讓 PHY-FSM 改爲 TX state，並開始傳送 Data 封包出去。

當 CTA#3 結束之後，會產生 Interrupt 事件「equal cta end time」。此 Interrupt ISR 會判斷下一個 CTA 爲何？CTA#4 等於 can't used，動作跟 CTA#1 一樣，PHY-FSM 爲 Ready 或 Sleep。當 CTA#4 結束之後，會產生 Interrupt 事件「equal cta end time」。此 Interrupt ISR 會判斷下一個 CTA 爲何？CTA#5 等於 RX。之後便設定 PHY-FSM 改爲 RX state，並開

始接收 Data 封包進來。

我們再來解釋圖 13 (B)。首先，PNC在傳送完Beacon封包之後或DEV在分析完Beacon封包之後，判斷下一個CTA爲何？因此要判斷CTA#1 是TX、RX或can't used。在圖 13B)中，CTA#1 等於TX。

在 CAP 區間結束後，因爲 CTA#1 等於 TX，之後便設定 TX_BUF_START 等於 CTA#1 所管的 Data Frame Buffer 的起始位址，還有設定 TX_BUF_LEN 等於傳送 Data 的總 bytes 數，最後設定 TX_START_TIME 爲 CTA#1 的 Start Time。當 SUPERFRAME_TIME 等於 TX_START_TIME 的時後，軟體會下達指令，讓 PHY-FSM 改爲 TX state，並開始傳送 Data 封包出去。

當 CTA#1 結束之後，會產生 Interrupt 事件「equal cta end time」。此 Interrupt ISR 會判斷下一個 CTA 爲何？CTA#2 等於 can't used。因爲 CTA#2 等於 can't used，所以 PHY-FSM 爲 Ready 或 Sleep，即不傳送也不接收，甚至可以把天線關閉。

當 CTA#2 結束之後，會產生 Interrupt 事件「equal cta end time」。此 Interrupt ISR 會判斷下一個 CTA 爲何？CTA#3 等於 TX。之後便設定 TX_BUF_START 等於 CTA#3 所管的 Data Frame Buffer 的起始位址，還有設定 TX_BUF_LEN 等於傳送 Data 的總 bytes 數，最後設定 TX_START_TIME 爲 CTA#3 的 Start Time。當 SUPERFRAME_TIME 等於 TX_START_TIME 的時後，軟體會下達指令，讓 PHY-FSM 改爲 TX state，並開始傳送 Data 封包出去。

當 CTA#3 結束之後，會產生 Interrupt 事件「equal cta end time」。此 Interrupt ISR 會判斷下一個 CTA 爲何？CTA#4 等於 TX。之後便設定 TX_BUF_START 等於 CTA#4 所管的 Data Frame Buffer 的起始位址，還有設定 TX_BUF_LEN 等於傳送 Data 的總 bytes 數，最後設定 TX_START_TIME 爲 CTA#4 的 Start Time。當 SUPERFRAME_TIME 等於 TX_START_TIME 的時後，軟體會下達指令，讓 PHY-FSM 改爲 TX state，並開始傳送 Data 封包出去。

當 CTA#4 結束之後，會產生 Interrupt 事件「equal cta end time」。此 Interrupt ISR 會判斷下一個 CTA 爲何？CTA#5 等於 RX。之後便設定 PHY-FSM 改爲 RX state，並開始

接收 Data 封包進來。

最後我們解釋圖 13 (C)。首先，PNC在傳送完Beacon封包之後或DEV在分析完Beacon封包之後，判斷下一個CTA爲何？因此要判斷CTA#1 是TX、RX或can't used。在圖 13 (C)中，CTA#1 等於RX。

在 CAP 區間結束後，因爲 CTA#1 等於 RX，之後便設定 PHY-FSM 改爲 RX state，並開始接收 Data 封包進來。

當 CTA#1 結束之後，會產生 Interrupt 事件「equal cta end time」。此 Interrupt ISR 會判斷下一個 CTA 爲何？CTA#2 等於 can't used。因爲 CTA#2 等於 can't used，所以 PHY-FSM 爲 Ready 或 Sleep，即不傳送也不接收，甚至可以把天線關閉。

當 CTA#2 結束之後，會產生 Interrupt 事件「equal cta end time」。此 Interrupt ISR 會判斷下一個 CTA 爲何？CTA#3 等於 RX。之後便設定 PHY-FSM 改爲 RX state，並開始接收 Data 封包進來。

當 CTA#3 結束之後，會產生 Interrupt 事件「equal cta end time」。此 Interrupt ISR 會判斷下一個 CTA 爲何？CTA#4 等於 RX。之後便設定 PHY-FSM 改爲 RX state，並開始接收 Data 封包進來。

當 CTA#4 結束之後，會產生 Interrupt 事件「equal cta end time」。此 Interrupt ISR 會判斷下一個 CTA 爲何？CTA#5 等於 TX。之後便設定 TX_BUF_START 等於 CTA#5 所管的 Data Frame Buffer 的起始位址，還有設定 TX_BUF_LEN 等於傳送 Data 的總 bytes 數，最後設定 TX_START_TIME 爲 CTA#5 的 Start Time。當 SUPERFRAME_TIME 等於 TX_START_TIME 的時後，軟體會下達指令，讓 PHY-FSM 改爲 TX state，並開始傳送 Data 封包出去。

4.4 Buffer Management 機制

IEEE 802.15.3 MAC 把封包分成五種類型：(1)Beacon、(2)Immediate ACK、(3)Delayed ACK、(4)Command、(5)Data。其中 Immediate ACK 封包的製作是由 MAC-HW 所負責的，目的是立即回應傳送端收到該封包。而其他類型封包的製作是由 MAC-SW 所負責。

因此，我們根據不同類型封包的製作給予不同的 Frame Buffer 暫存，以加快 UWB 系統在傳送封包時的速率，例如：(1)在 Superframe 一開始，選擇 Beacon Frame Buffer 內所暫存的 Beacon frame，(2)在 CAP 一開始，選擇 Command Frame Buffer 內所暫存的 Command frames，(3)在 CTAP 一開始，選擇目前 CTA 的 Data Frame Buffer 與 Delayed ACK Frame Buffer 內所暫存的 Data frames 與 Delayed frames。因此，軟體在傳送封包之前，會更新 TX_BUF_START，其值等於 Frame Buffer 在記憶體的起始位置。

UWB系統接收所有類型封包到相同的Frame Buffer，這是因爲如果要以UWB HW來根據封包的類型來切換到不同的Frame Buffer，會使得UWB HW設計複雜，背離我們以簡單HW設計的理念。因此，我們統一以RX Frame Buffer來暫存所有接收到的封包。而接收到的Beacon封包需要立即解析，於是我們設計中斷的機制，要求UWB SW立即解析Beacon內容，以更新CTAP Timing相關資訊。表 6列出UWB系統使用到的Frame Buffer種類。

表 6　Frame Buffer 種類

TX Buffer	RX Buffer
◆ Beacon Frame Buffer ◆ Delayed ACK Frame Buffer ◆ Command Frame Buffer ◆ Data Frame Buffer	◆ RX Frame Buffer

圖 14、圖 15與圖 16說明Frame Buffer的設計理念。首先，圖 14中struct buf_info tx_buf或rx_buf爲global指標，表示是傳送的Frame Buffer或者是接收的Frame Buffer。配置Frame Buffer的步驟：(1)根據不同種類的Frame Buffer，配置不同大小的記憶體，如圖16。這是因爲不同類型封包需求不同大小的暫存空間，例如每Superframe開始才一個

Beacon傳送出去，而同一個CTA中Data卻有可能很多個被傳送出去。(2)Frame Buffer的記憶體管理，其分成兩個部分：buf_info與buf_ring，如圖 14與圖 15。而buf_info紀錄Frame Buffer目前所暫存的封包個數，並以Ring的模式來暫存之。

封包以 Ring 模式來暫存的原因是爲了效能考量，我們必須預先配置一塊連續的記憶體空間，提供 UWB SW 依序暫存已經製作好的封包，而 UWB HW 也依序傳送封包出去，這樣「預先配置記憶體」與「先進先出」的觀念，因此可以用 Ring 來實現。以 Ring 實現的好處：(1)可以重複使用記憶體，降低因配置記憶體之要求所帶來的 overhead，(2)因爲連續記憶體的關係，故指定 UWB HW 需傳送封包的起始位址即可，降低因從 UWB SW 複製封包到 UWB HW 所帶來的 overhead。

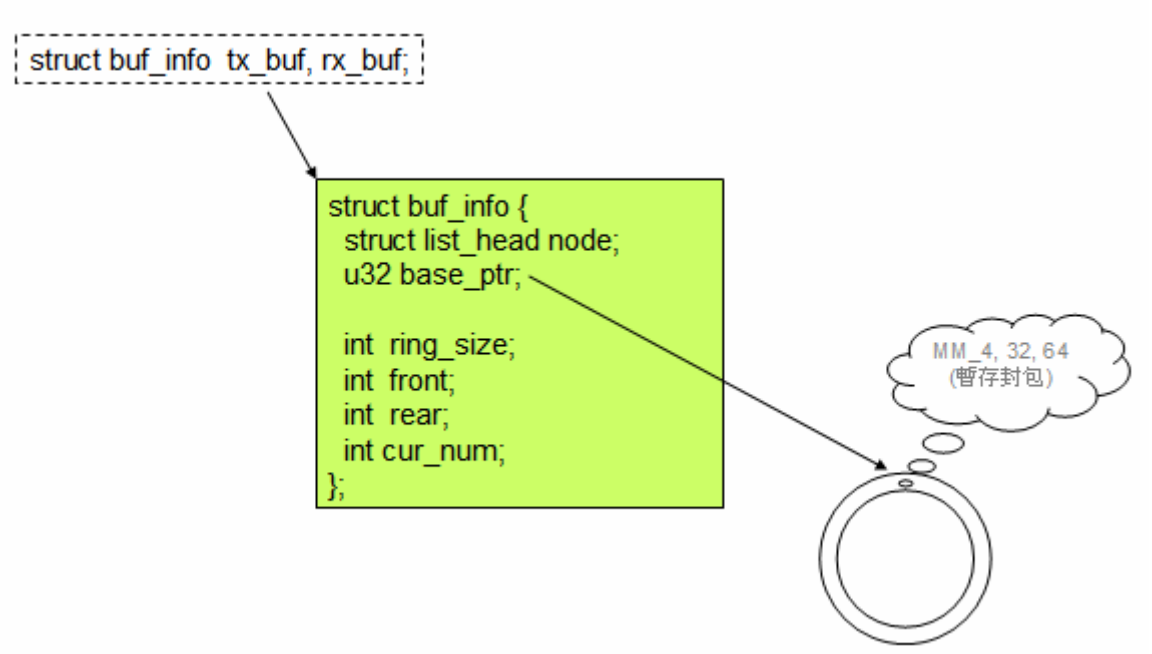

圖 14　Frame Buffer 示意圖(1)

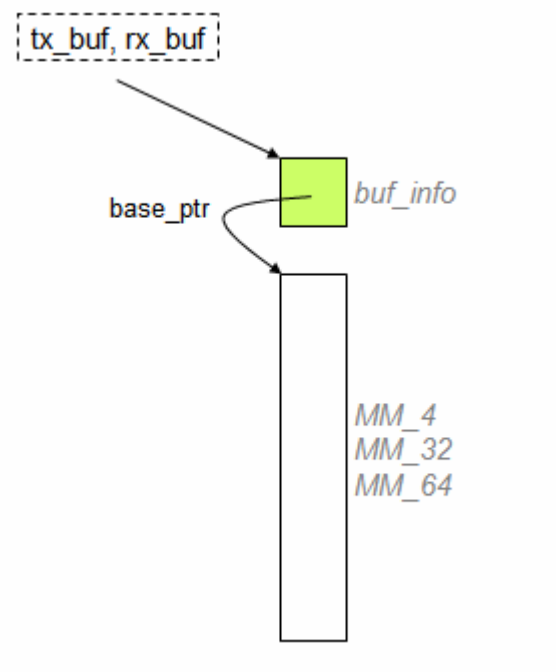

圖 15　Frame Buffer 示意圖(2)

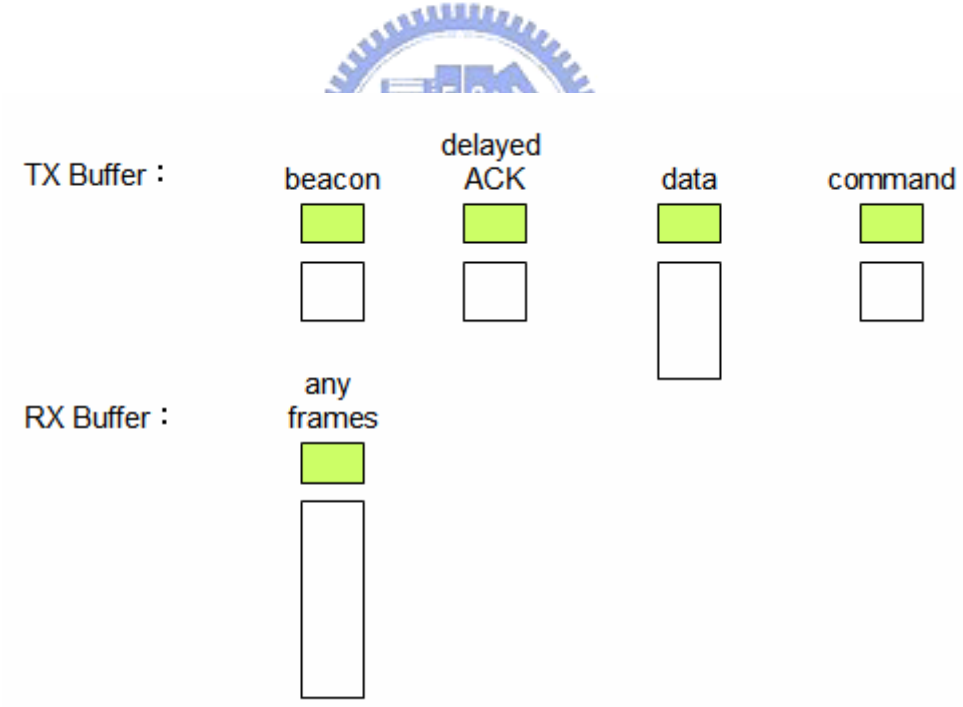

圖 16　Frame Buffer 種類

緊接著，我們將說明 RX Buffer 與 TX Buffer 在 UWB SW 中是如何運作之。

4.4.1 RX Buffer

圖 17爲RX Frame Buffers初始示意圖。rx_hw1 和rx_hw2 初始之後，會被指定給RX_BUF_START1 和RX_BUF_START2，以立即提供接收封包時所需的暫存區。而且rx_pre_hw Ring也預先配置好了，一旦RX Buffer1 或RX Buffer2 is Ready(Full或Timeout)，可以馬上更改指標之後，又可以繼續接收封包，避免因爲要求記憶體所帶來的延遲。並且由圖 18、圖 19、圖 20和圖 21來解釋RX Frame Buffer的運作過程。

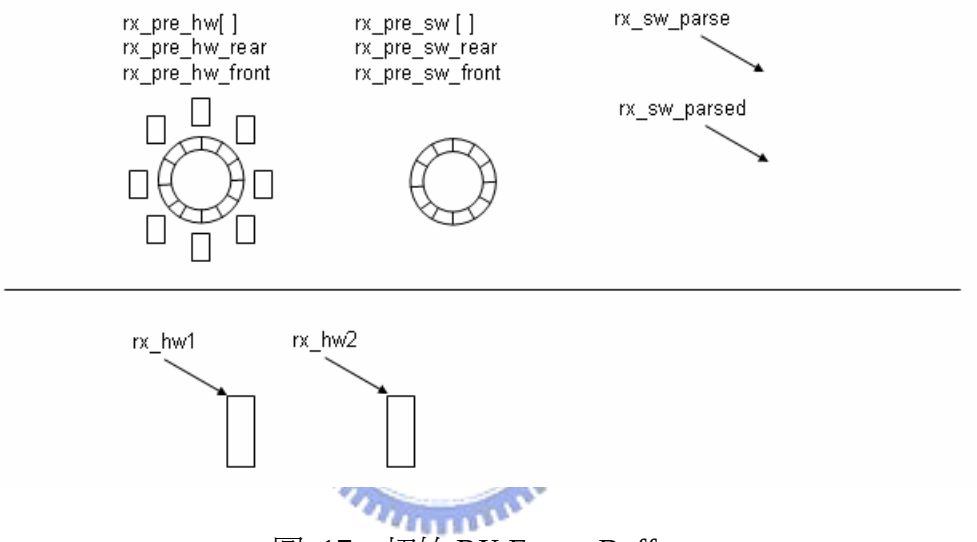

圖 17　初始 RX Frame Buffers

圖 18爲Interrupt事件「Buffer1 is ready」。第 1 步驟：從rx_pre_hw Ring中取出一個RX Frame Buffer當新的rx_hw1。第 2 步驟：而裝滿或時間超過的rx_hw1 Frame Buffer則被加入rx_pre_sw Ring中等待著封包被分析。相同地，圖 19爲Interrupt事件「Buffer2 is ready」。第 1 步驟：從rx_pre_hw Ring中取出一個RX Frame Buffer當新的rx_hw2。第 2 步驟：而裝滿或時間超過的rx_hw2 Frame Buffer則被加入rx_pre_sw Ring中等待著封包被分析。然後在圖 20當中，因爲DEV收到Beacon封包，需要馬上被處理，因此產生Interrupt事件「Beacon reached」。第 1 步驟：硬體讓目前接收的RX Frame Buffer是爲timeout，因此發生如圖 18或圖 19的情況。

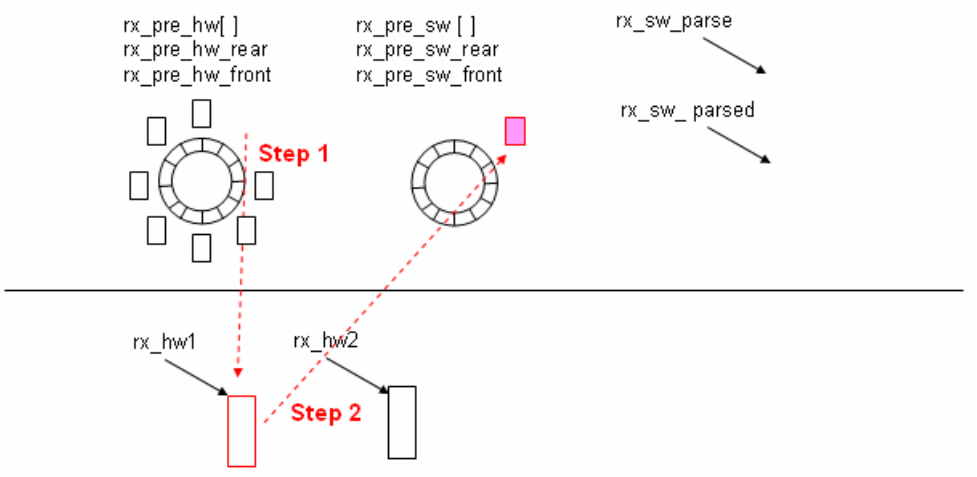

圖 18　Interrupt 發生：Buffer1 is ready

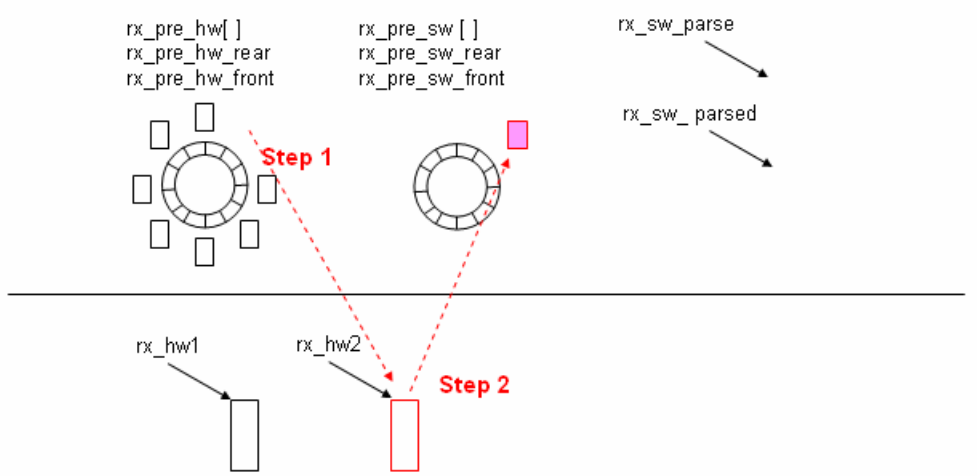

圖 19　Interrupt 發生：Buffer 2 is ready

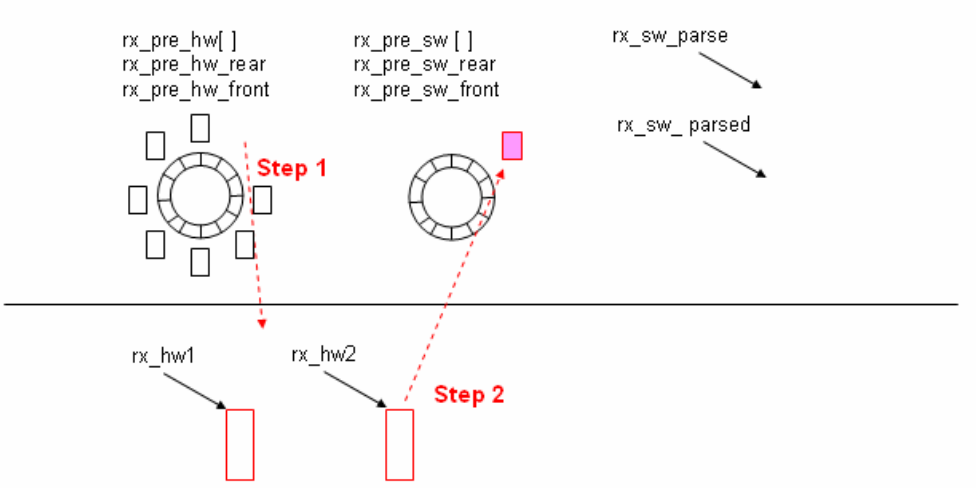

圖 20　Interrupt 發生：Beacon reached

在圖 21當中，我們將說明封包何時被解析。第 3 步驟：我們初始RX Parser Thread，內爲無窮迴圈，會判斷rx_per_sw Ring是否有Frame Buffer要解析。一旦有Frame Buffer被加入之後，先把所有在rx_pre_sw Ring內的Frame Buffers加入到rx_sw_parse那等待被解析。第 4 步驟：接著RX Parser Thread會對rx_sw_parse的Frame Buffers解析其收到的封包。解析完畢的Frame Buffer接著會被移到rx_sw_parsed，其目的是資源重複使用，以避免要求記憶體所帶來的延遲。第 5 步驟：在Tasklet中，若rx_sw_parsed有Frame Buffer，我們會移到rx_pre_hw Ring中。

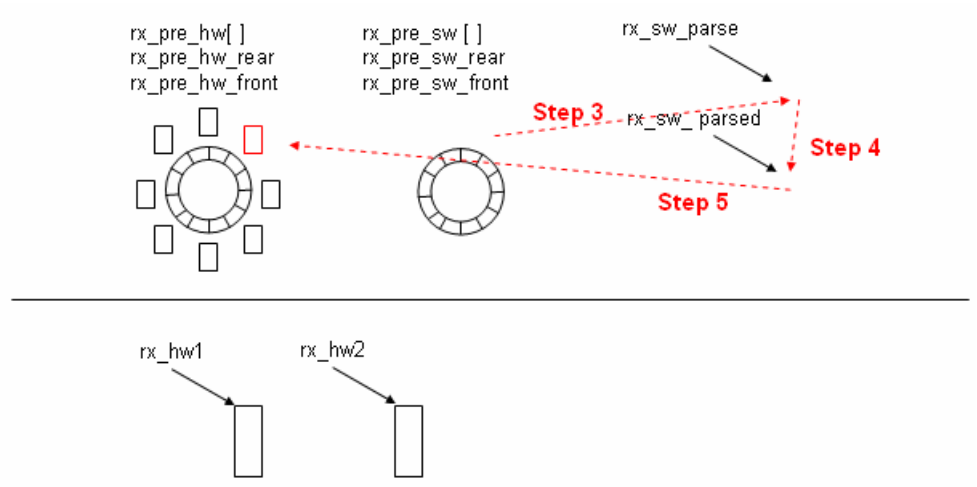

圖 21　RX Parser Thread

4.4.2 TX Buffer

我們根據不同類型封包的製作給予不同的 Frame Buffer 暫存，以加快 UWB 系統在傳送封包時的速率。因此這一小節我們將對 Beacon Frame Buffer、Command Frame Buffer、Data Frame Buffer 和 Delayed ACK Frame Buffer 的運作機制加以說明之。

圖 22爲Beacon Frame Buffer初始化的示意圖。我們使用兩個指標 tx_hw_b和tx_pre_hw_b來輪流暫存Beacon封包的製作。圖 23中，在製作Beacon封包的時後，我們使用tx_pre_hw_b來暫存之，要傳送Beacon封包之前，我們會調換tx_hw_b與tx_pre_hw_b，並讓TX_BUF_START等於tx_hw_b內的Beacon封包起始位址。而調換後的tx_pre_hw_b又可以繼續製作Beacon封包。因此，我們使用兩個Frame Buffer來輪流暫存Beacon封包，目的是要避免要求記憶體所帶來的延遲。

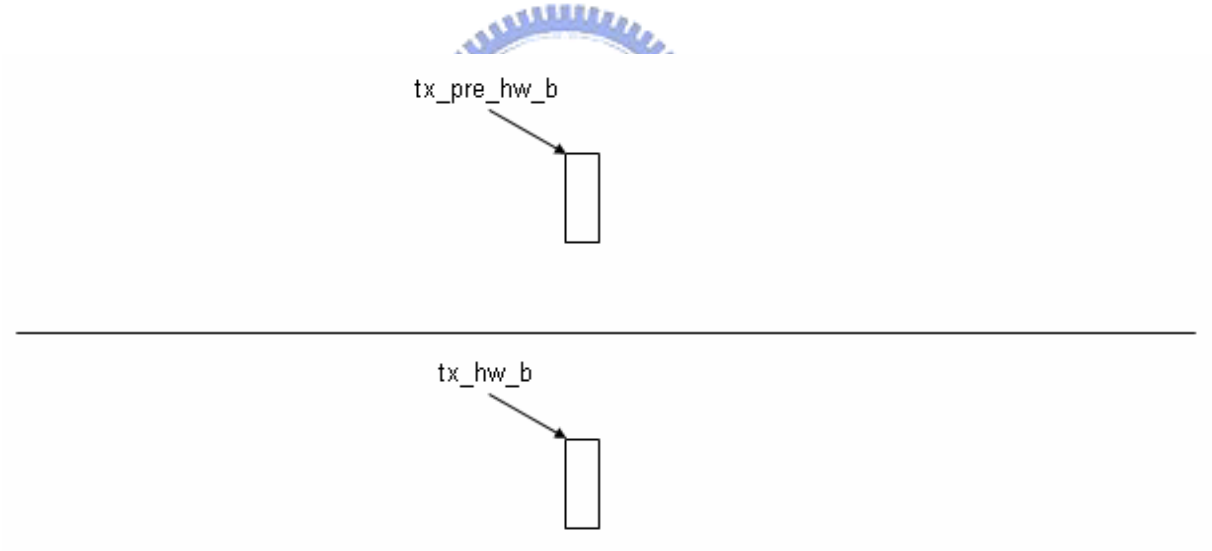

圖 22　初始 Beacon Frame Buffer

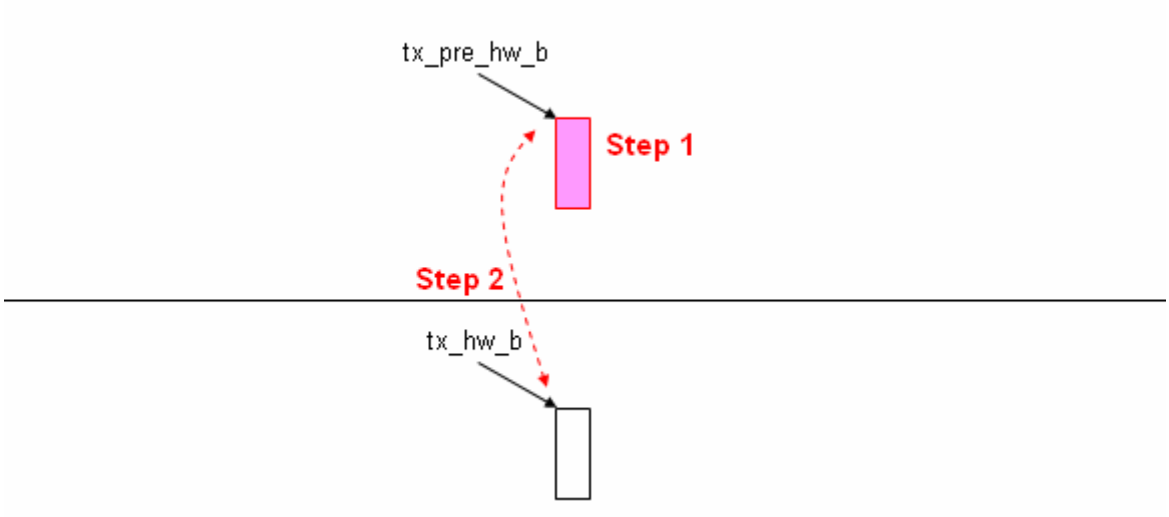

圖 23　Beacon Frame Buffer 輪流暫存

圖 24爲Command Frame Buffer初始化的示意圖。我們使用五個指標來處理製作Command封包的動作。後面我們會解釋這五個指標如何運作。

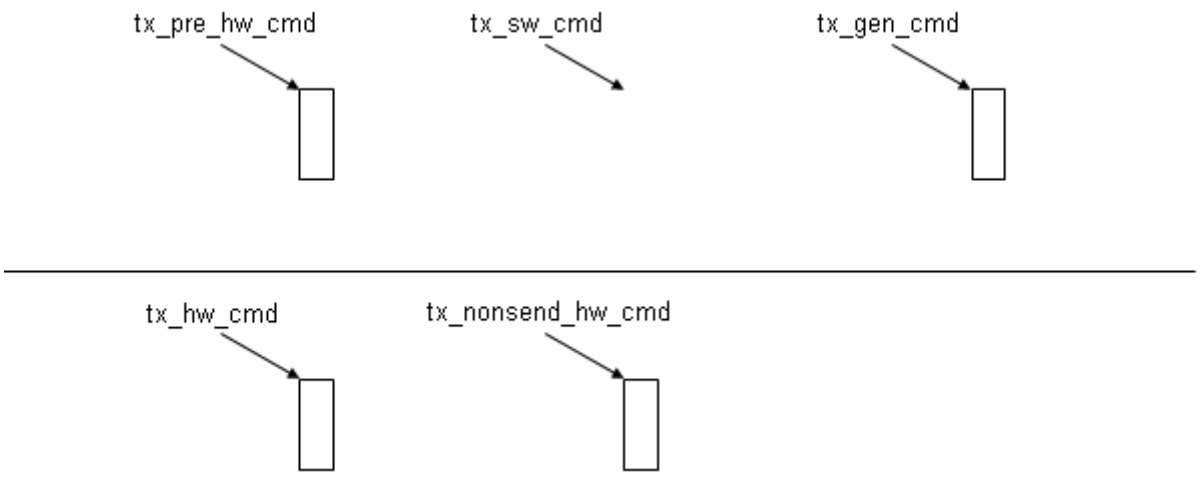

圖 24　初始 Command Frame Buffer

圖 25和圖 26爲製作Command封包的流程圖。第 1 步驟：使用tx_gen_cmd Buffer來

製作Command封包，一旦製作完畢以後，第 2 步驟：tx_pre_hw_cmd與tx_sw_cmd調換，第 3 步驟：然後製作好的Command封包以複製的方式複製到tx_sw_cmd。複製完畢之後，第 4 步驟：tx_pre_hw_cmd與tx_sw_cmd再調換回來。

若不用 tx_pre_hw_cmd、tx_sw_cmd 和 tx_gen_cmd 三個指標，而只用 tx_pre_hw_cmd 一個指標來製作 Command 封包，有可能會發生以下的情況。若製作 Command 封包時使用 tx_pre_hw_cmd Frame Buffer，一旦進入 CAP 且 Channel 閒置一段時間，可以開始傳送 Command 封包，但是目前 tx_pre_hw_cmd 卻被軟體所 Block 住，導致無法把之前製作好的 Command 封包傳送出去。加上 Interrupt ISR 內無法使用 Blocking 的機制，故我們使用了 tx_pre_hw_cmd、tx_sw_cmd 和 tx_gen_cmd 三個 Frame Buffer 來處理以上的情況。

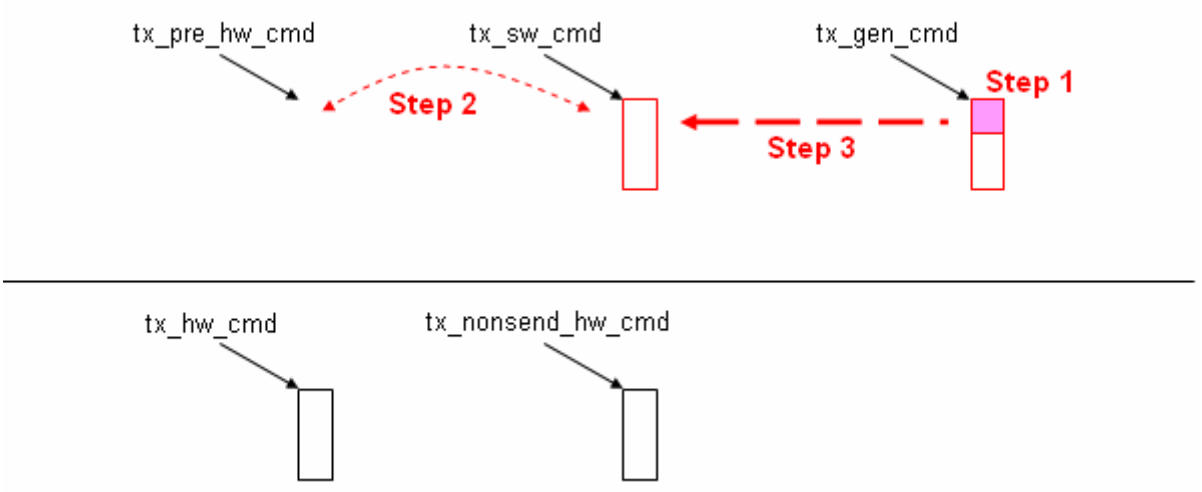

圖 25　製作 Command 封包(1)

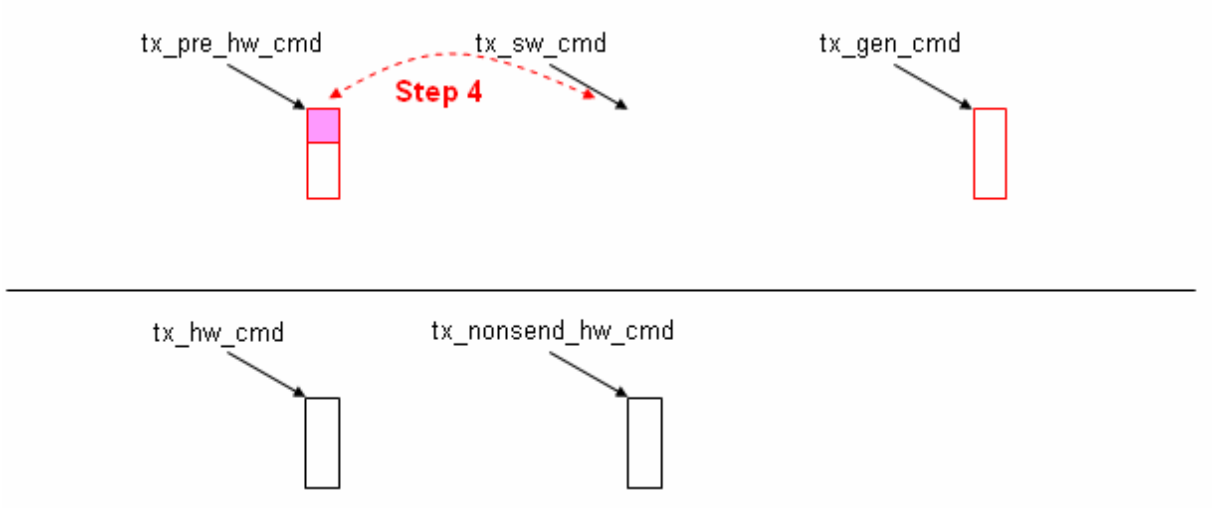

圖 26　製作 Command 封包(2)

圖 27和圖 28傳送Command封包的流程圖。第 1 步驟：一旦進入CAP區間，便開始啓動Slotted Aloha機制開始倒數，若Channel持續閒置某段時間，就開始傳送Command封包。首先，若有Command封包已被製作好，則我們交換tx_hw_cmd和tx_pre_hw_cmd指標，並設定TX_BUF_START等於tx_hw_cmd內Command封包的起始位址。第 2 步驟：Channel持續閒置某段時間，開始傳送tx_hw_cmd內Command封包出去。

第 3 步驟：一旦 CAP 區間結束，而 tx_hw_cmd 內的 Command 封包因爲競爭模式可能無法傳送完畢，若 tx_hw_cmd 內還有 Command 封包，則交換 tx_hw_cmd 和 tx_nonsend_cmd 指標。第 4 步驟：交換 tx_pre_hw_cmd 和 tx_sw_cmd。第 5 步驟：把 tx_nonsend_cmd 內未送出的 Command 封包複製到 tx_sw_cmd 中。複製完畢之後，再交換 tx_pre_hw_cmd 和 tx_sw_cmd，等待下一次 CAP 區間可以傳送 Command 封包。

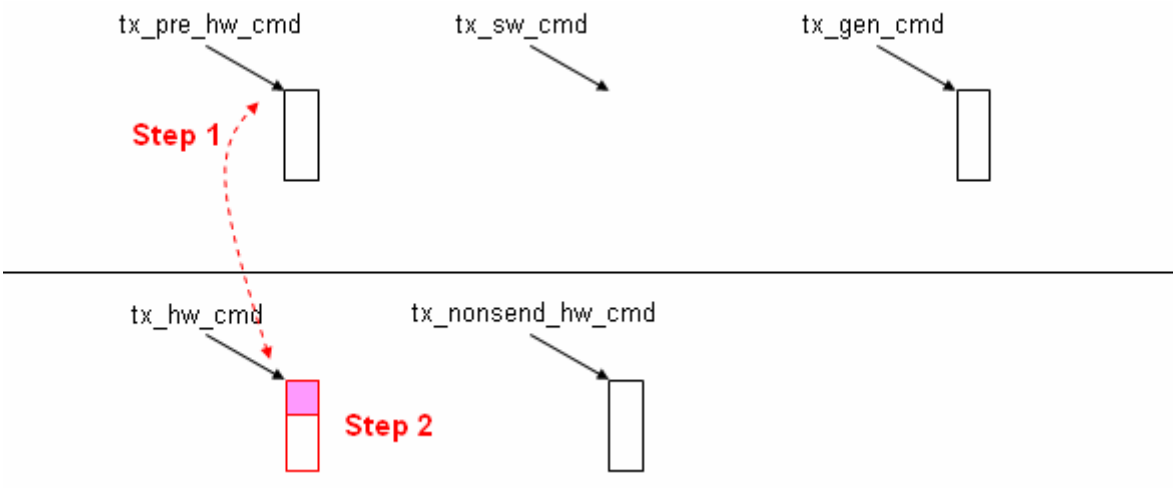

圖 27　傳送 Command 封包(1)

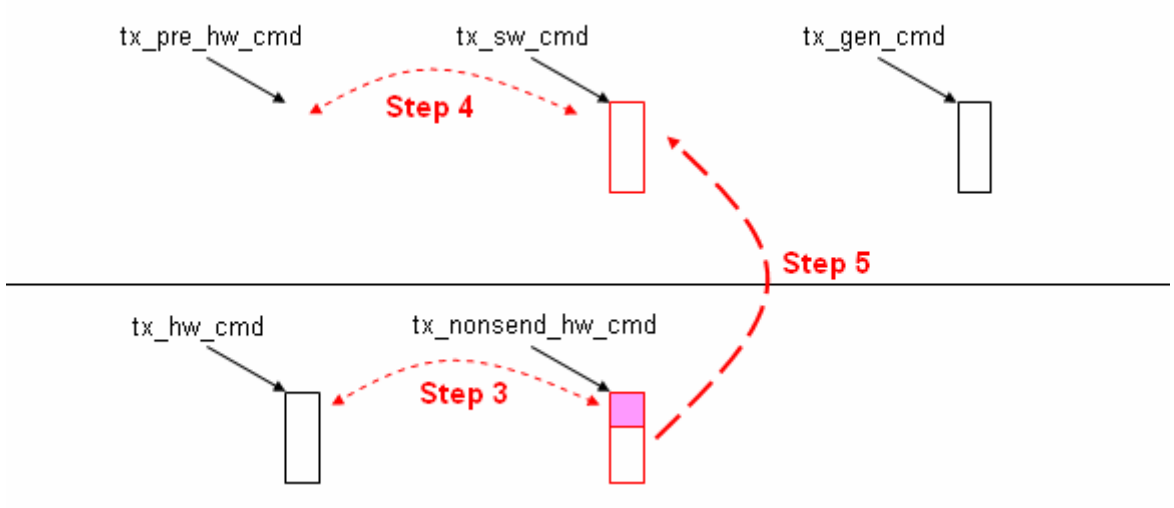

圖 28　傳送 Command 封包(2)

圖 29爲用來紀錄CTAP區間內，所有CTA區塊的資訊。一旦要搜尋某個CTA，我們由cta_head開始線性找尋目標CTA。其搜尋的key爲source id、destination id、stream id。一旦有新的CTA產生或CTA結束，可以使用cta_resort_head來重新調整CTA在CTAP內的順序。

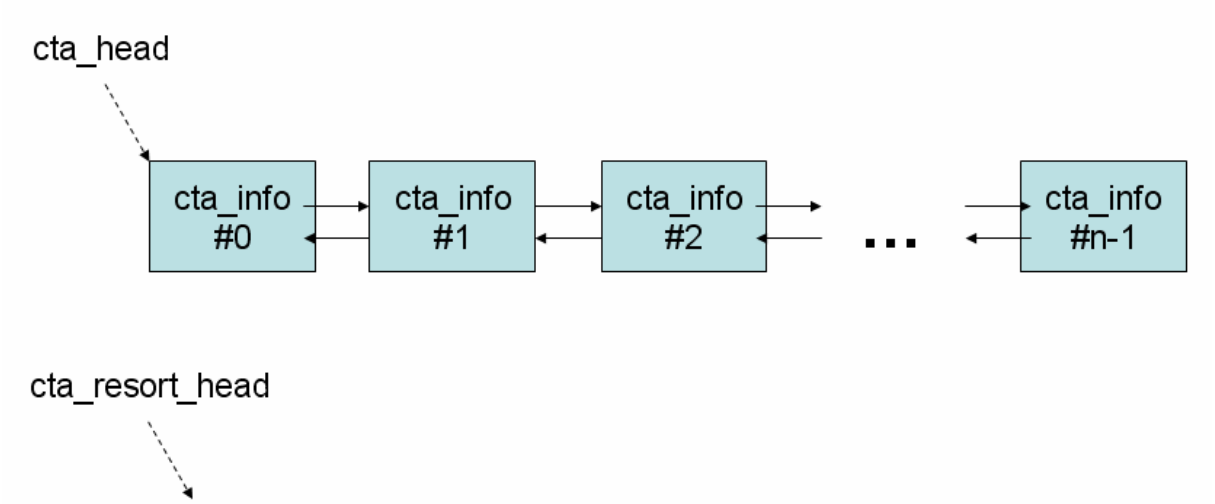

圖 29　CTA List

圖 30爲Data Frame Buffer初始化的示意圖。每一個cta_info結構內皆含圖 30內初始的指標。後面我們會解釋Data Frame Buffer如何運作之。

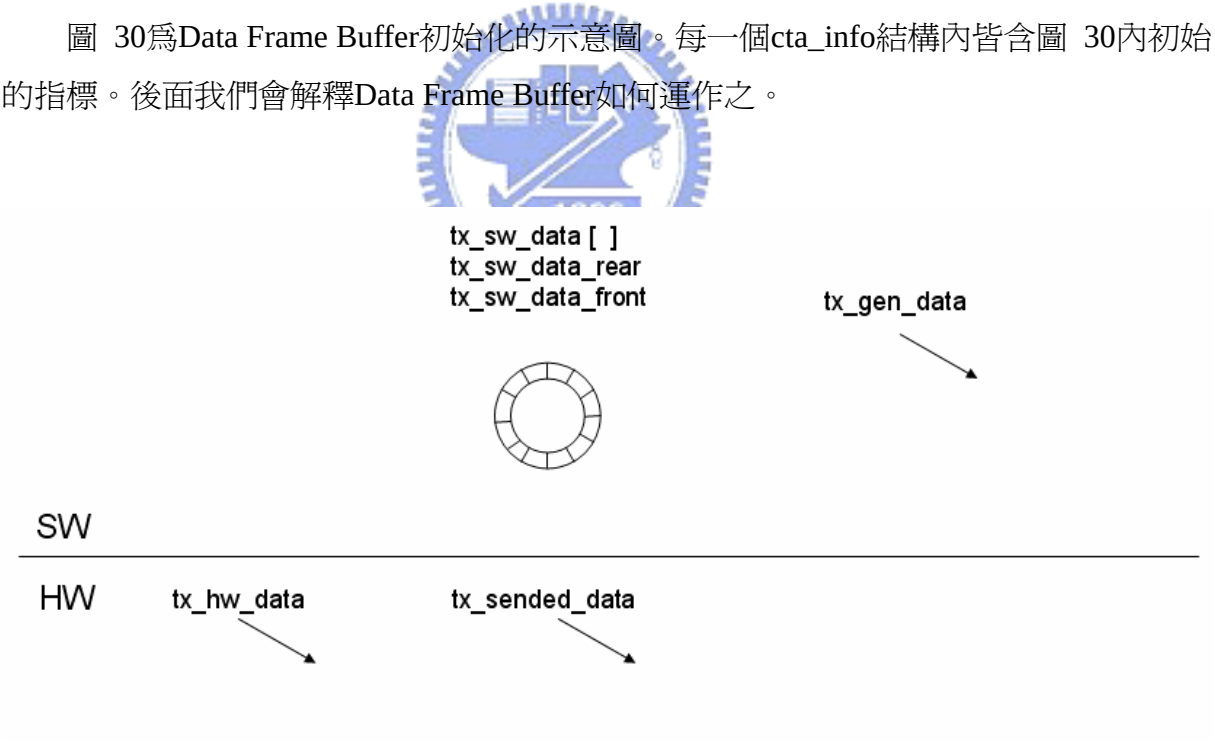

圖 30　初始 Data Frame Buffer

圖 31和圖 32爲製作Data封包的流程圖。第 1 步驟：在製作Data封包之前，會先要求記憶體空間，然後讓tx_gen_data指到該空間，接著製作Data封包。第 2 步驟：一旦tx_gen_data Frame Buffer滿載了或欲傳送的資料被製作完畢，則把tx_gen_data內的Frame Buffer加入到tx_sw_data Ring中。第3步驟:若下一個CTA爲TX CTA,則把tx_sw_data Ring內的一個Data Frame Buffer移至tx_hw_data。當tx_hw_data內的Data封包被傳送完畢，則把Data Frame Buffer移至tx_sended_data，以提供重複再利用。第 5 步驟：製作Data封包之前，若要要求記憶體的話，會先看tx_sended_data是否有空的Data Frame Buffer。

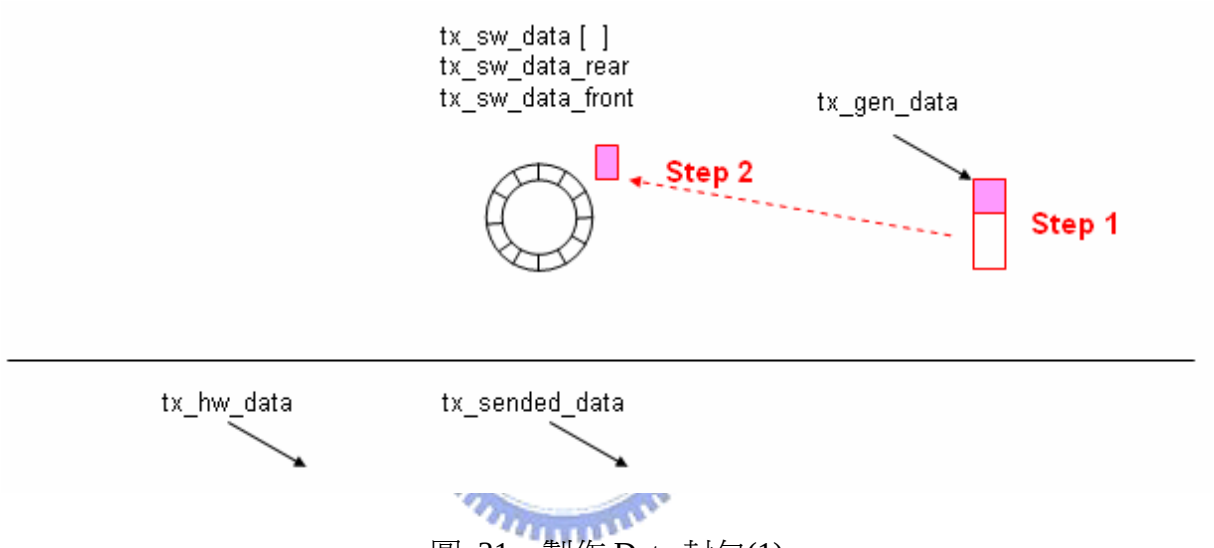

圖 31　製作 Data 封包(1)

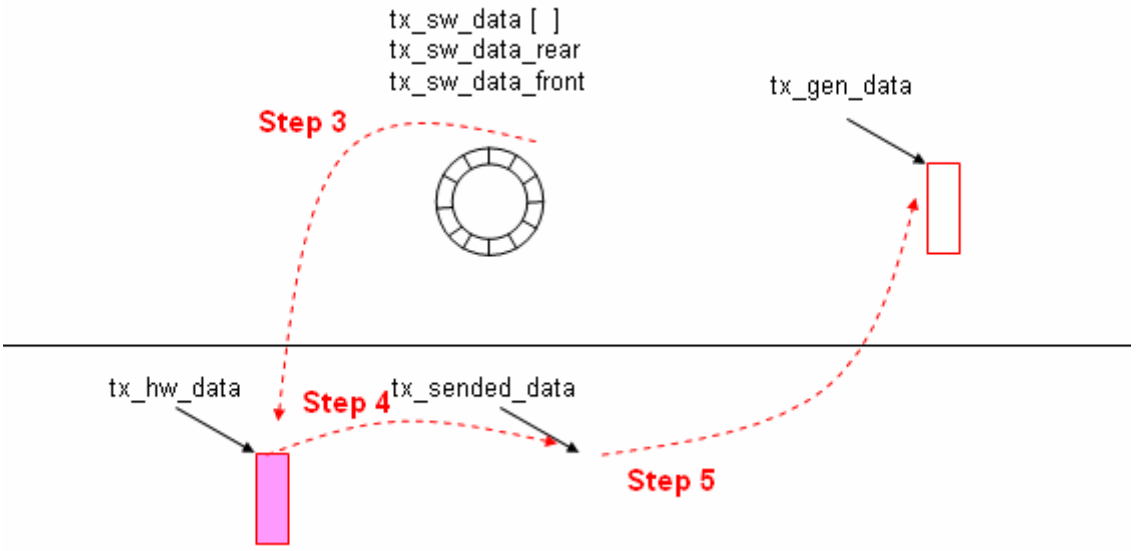

圖 32　製作 Data 封包(2)

圖 33爲Delayed ACK Frame Buffer初始化的示意圖。每一個cta_info結構內皆含圖33內初始的指標。Delayed ACK Frame Buffer運作機制與Data Frame Buffer機制類似，故不再加以說明之。

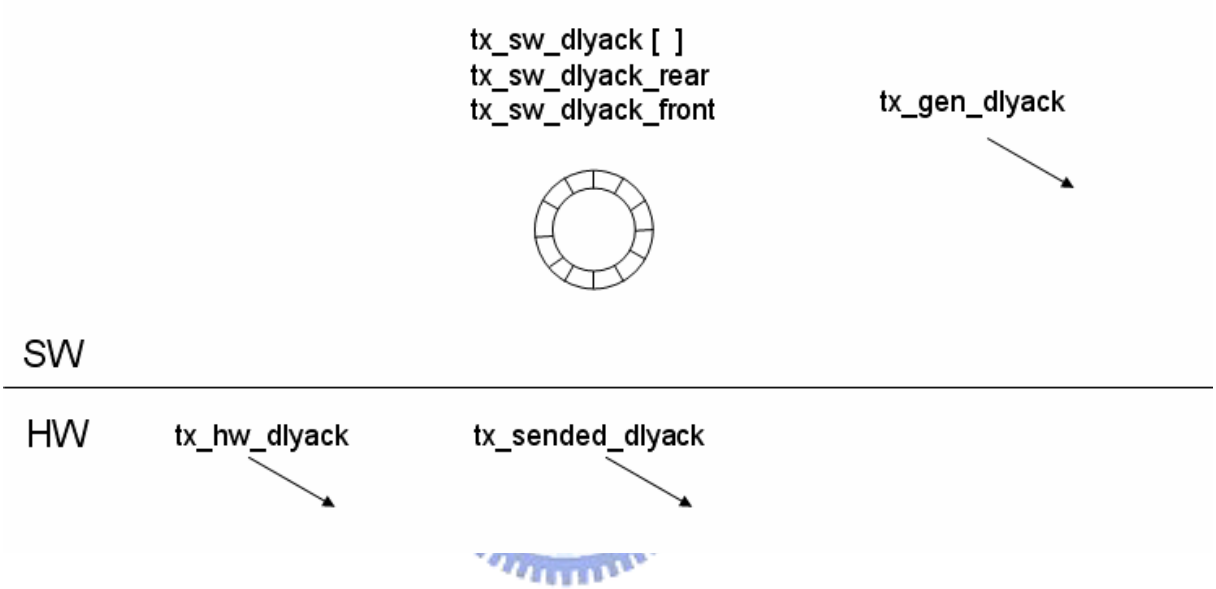

圖 33　初始 Delayed ACK Frame Buffer

第5章　模擬與結果

因爲我們是以軟體爲主來設計 UWB 介質存取控制器，所遭遇到的主要困難是 Superframe Timing 和 TDMA Timing 準確性的問題。因此，在這一章我們將對做 UWB 介質存取控制器做了一些評估。首先，我們針對 Superframe Timing 和 TDMA Timing 的準確性做了一些測量。其次，我們也評估 UWB 介質存取控制器的最大 Throughput 爲何。

5.1　模擬環境

我們使用 ARM Developer Suite v1.2 來開發 UWB-SW，而 ConvergenSC ver.2004.2.3 (Coware Inc.)則來開發 UWB-HW。我們模擬所用的電腦的環境如下：CPU – Intel P4 2.0 GHz、Memory – 512 MB、OS – Redhat 8.0 (kernel : 2.4.18)。

5.2　模擬結果

首先，我們先測量 Superframe Timing 是否準確，也就是說要測量每一個 Superframe Duration 誤差是否可以被接受。我們從 superframe time=0 時開始計算，到 beacon frame 被完整的送出 UWB-MAC 之間的 clock 數，我們總共測量了 20 次，並取平均值 clock 數值爲 23 clocks。而每個 clock 爲 8ns，因此得到延遲 184ns，相對 Superframe duration 而且，誤差可以被接受。

接著，我們再測量 TDMA Timing 是否準確，也就是說要測量 CTA 與 CTA 之間 guard time 時的中斷延遲是否可以被接受。假設緊接著的 CTA 爲傳送模式，則 guard time 內的中斷需要設定以下 Registers：CTA_END、TX_BUF_START、TX_BUF_LEN、TX_START_TIME、CTA_END。因爲會佔用 AHB 資源，因而導致一些延遲。我們總共測量了 20 個 CTA 與 CTA 之間中斷時所需的 clock 數，並取平均值。平均值爲 406 clocks，而每個 clock 爲 8 ns，因此中斷並更新 Registers 大約需花 3246ns。而 guard time 爲 12ms，

故 TDMA Timing 還算準確。

最後，我們將針對 UWB 介質存取控制器的 Throughput 是多少，也就是要測量 data frames 從 Memory 透過 UWB-HW 送出的最大 Throughput。我們測量一個 DEV 開始要把 32 個 data frames 要送出到另外一個 DEV，到完全傳輸完畢時，所花的 clocks 數，我們也測量了 20 次，並且取其平均值，我們得到 clocks 爲 377376，而每個 clock 爲 8 ns。故傳輸 32 個 data frames 需 3091 us，因此可以算出 Throughput 爲 170.28 Mbps。

第6章　結論與未來工作

本篇論文以軟硬體協同設計(Hardware/Software Codesign)的方式來設計一個以 IEEE 802.15.3 MAC 爲基礎的 UWB 介質存取控制器。我們以軟體爲主來實作 UWB 介質存取控制器的好處是：軟體實作可以縮短系統開發的時間，而且版本更新上也比較容易。

本篇論文的貢獻是如何讓軟體爲主的設計近似於純硬體設計之 Superframe Timing 和 TDMA Timing。除了 Timing 的問題，我們也設計一套了記憶體管理的方式來決定封包所暫存的記憶體起始位址，並且透過 Ring 的結構來加快存取速度，並且能夠重複使用記憶體空間，以降低要求記憶體時所帶來的延遲，以滿足 UWB 之高 Throughput 的特性。

我們實作上採用 CoWare 公司所提供的 ConvergenSC Tools 來模擬我們的 UWB 介質存取控制器。因此，在未來的工作方面，我們可以把 UWB 介質存取控制器真實的實作在兩張 ARM-based Board 上，更能驗證我們 UWB 介質存取控制器的正確性。

參考文獻

[1] Fontana, R. Ameti, A. Richley, E. Beard, L. Guy, D., “Recent Advances in Ultra Wideband Communication Systems,” Proceedings of IEEE Conference on Ultra Wideband Systems and Technologies, pp 129-133, May 2002.

[2] David G. Leeper, “A Long-Term View of Short-Range Wireless,” IEEE Computer Magazine, Vol. 34, No. 6, pp 39- 44, June 2001.

[3] David G. Leeper, “Ultrawideband – The Next Step in Short-Range Wireless,” IEEE MTT-S International Conference on Microwave Symposium Digest, Vol. 1, pp. 357-360, 2003.

[4] Woe Z. Win, Robert A. Scholtz, “Impulse Radio: How It Works,” IEEE Communications Letters, Vol. 2, No. 1, pp. 36-38, February 1998.

[5] Jeff Foerster, Evan Green, Srinivasa Somayazulu, and David Leeper, "Ultra-Wideband Technology for Short- or Medium-Range Wireless Communications," Inter Technology Journal Q2, 2001.

[6] IEEE P802.15.3™ Standard for Information technology – Telecommunications and information exchange between systems – local and metropolitan are networks – Specific requirements - Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs), September, 2003.

[7] Freescale Semiconductor Inc., available at http://www.freescale.com, August 2004.

[8] IEEE 802.15 Working Group for WPAN: http://grouper.ieee.org/groups/802/15.

[9] Domenico Porcino and Walter Hirt, “Ultra-Wideband Radio Technology: Potential and Challenges Ahead,” IEEE Communication Magazine, pp. 66-74, July 2003.

[10] MultiBand OFDM Alliance, available at http://www.multibandofdm.org/

[11] Federal Communications Commission , available at http://www.fcc.gov/

[12] G. J. Foschini, Jr. and M. J. Gans, “On limits of wireless communications in a fading environment when using multiple antennas,” *Wireless Personal Communication*, vol. 6, pp. 311-335, March 1998.