碩士學位論文

블루투스 베이스밴드의 ASIC 설계 및 HCI에서의 최적 패킷 전송

濟州大學校 **大學院**

通信工學科

金 宰 必

블루투스 베이스밴드의 ASIC 설계 및 HCI에서의 최적 패킷 전송

指導敎授　林 載 允

金 宰 必

이 論文을 工學 碩士學位 論文으로 提出함

2001 年　12 月

金宰必의 工學 碩士學位 論文을 認准함

審査委員長　金 興 洙　印
委　　員　康 鎭 植　印
委　　員　林 載 允　印

濟州大學校 大學院

2001年　12 月

ASIC Design Of Bluetooth Baseband and Optimal Packet Transmission In HCI

Jae-Pil Kim
(Supervised by professor Jea-Yun Lim)

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Science

2001. 12.
This thesis has been examined and approved.

Thesis director, Heung-Soo Kim, Prof. of Telecom. Eng.

(Name and signature)

Date

DEPARTMENT OF TELECOMMUNICATION
ENGINEERING GRADUATE SCHOOL
CHEJU NATIONAL UNIVERSITY

목 차

Abstract

The baseband is responsible for channel coding, decoding, low level control of the timing and management of the link within the domain of a single data packet transfer. It adds addressing and link control fields to the raw payload data and provides error detection and correction method.

This thesis presented the SAFER+ algorithm for efficient authentication and method to solve problem of the frequency hop selector.

The proposed SAFER+ algorithm can reduce the step of the authentication process, and it can also reduce the whole circuit size by using two MUX and counter. Proposed method could get delay improvement effect by excluding divider in the frequency hop selector.

To prove validity of this method, simulation environment was established by connecting the host and the module with RS232C. And the process to connect two modules was confirmed through the HCI event packet occurring according to each command. And then, Bluetooth system performance was analyzed by comparing with the result of transmission achievement according to data size and distance gradation.

This thesis is intended to have Bluetooth's characteristic that is inexpensive, reliable, small, and low power.

Ⅰ. 서 론

근거리 무선 통신의 새로운 통신방식으로 부각되는 블루투스는 작고 저렴한 가격, 모든 디지털 제품들에 저가의 블루투스 칩을 넣어서 사용자가 장치를 연결할 필요 없이 적은 전력소모로 근접시 장치들이 자동으로 동기를 맞추게 하는 것을 목표로 하고 있다.

블루투스 칩은 크게 RF부와 베이스밴드로 구성되어있다. RF부는 무선망을 구성하며 이 구성은 79개의 교신 채널을 보유하고 있고, 베이스밴드는 송수신할 내용과 제품군에 따라 필요한 정보를 제공하도록 한다. 즉 동일지역에 여러 개의 블루투스 기기가 위치하여 있을 때 각각의 블루투스 기기는 자신과 동일한 성격(동일 제품군)의 블루투스 기기를 찾아 교신을 시도하도록 구성되어야 한다. 그리고, 블루투스 기기는 동일 성격의 블루투스 기기가 한 지역에 있으면 블루투스 기기는 스스로 프로그램에 의해 최고 7개가 하나의 군을 이룬다. 그 중 하나가 마스터(Master) 기기가 되고 나머지 6개는 슬레이브(Slave) 기기가 된다. 마스터와 슬레이브로 구성된 블루투스 기기군은 마스터를 중심으로 교신을 이룬다. 마스터는 필요시 슬레이브를 호출하여 서로의 정보를 교환하며 이때 차후 교신할 호핑 시퀀스(Hopping Sequence)를 설정한다. 이와 같은 순서로 마스터는 6개의 슬레이브와 순차적으로 송수신을 실시하여, 어느 나라 어느 곳에서 제조되어도 동일제품간의 교신이 이루어질 수 있는 것이다(Nathan J. MULLER, 2000)(Miller and Brent A, 2000)(Held and Gilbert, 2000)(동역메카트로닉스연구소 기술정보실, 2000). 이런 많은 장점을 가진 블루투스지만 케이블에 기반하여 기본적인 보안이 제공되는 유선 통신과는 다르게 블루투스는 무선 통신이므로 데이터의 정확한 전송과 보안이 중요한 문제라고 할 수 있겠다. 그래서 블루투스 스택의 가장 하위 계층인 베이스밴드에서 기기간의 배치에 제한이 없고 동기식/비동기식 응용프로그램이 모두 지원되는 채널접근코드와 전송된 패킷의 에러를 체크하고 보안을 제공하는 암호화기가 필요하다(Bluetooth Special Interest Group, 1999). 또한, 사용자 데이터의 보호와 서비스 이용에 대한 완벽한 상호 호환성을 위한 사용자 인증, 데이터의 기밀성 그리고 데이터의 무결성이 요구되어지므로 블루투스 인증 부분과 인증키의 종류에 따른 생성 방법도 요구된다.

한편, 블루투스 시스템의 상위계층은 호스트장치 프로세서에서 작동되고, 하위계층은 블루투스 장치에서 작동되는 시스템에서 상위계층과 하위계층간에 인터페이스가 필요하며, 이를 위한 블루투스 표준은 HCI(Host Controller Interface)로 규정한다. 이것을 표준 인터페이스로 하면, 상위계층과 하위계층의 혼합이 가능하다. 이처럼, 블루투스 장치를 상위계층과 하위계층으로 분리하여 HCI를 사용하게 되면, 블루투스 장치의 메모리 용량을 적게 하여 프로세서나 DSP를 적게 요하므로 가격을 낮출 수 있고, 블루투스 장치의 시험과 형식승인에 유용하고, 블루투스 접속요구에 따라 호스트 장치를 개·폐시킬 수 있는 장점을 갖게 된다(Jennifer Bray and Charles F Sturmam, 2001).

본 논문에서는 블루투스의 베이스밴드 설계를 채널접근코드 알고리즘과 암호화 알고리즘에 따라 구현하고, 효율적으로 인증을 수행하기 위한 SAFER+ 알고리즘의 개선 방안 및 구현 기법을 제안한다. 그리고, 교신할 호핑 시퀀스를 설계시 제산기를 사용하지 않음으로써 지연을 개선하는 기법을 제시한다. 이상의 베이스밴드 설계를 ASIC(Application Specific Integrated Circuit)으로 구현하고, RF단과 기본 프로토콜을 상호 연결한 시스템을 설계하여 두 모듈간의 연결까지의 동작을 HCI 이벤트를 실제로 수행하는 과정을 보인다. 그리고, 데이터의 크기별에 따라 실내 환경에서 발생 가능한 상황에 따라 실제적으로 파일전송을 수행함으로써 전송시간과 전송속도를 통해 성능을 분석한다.

본 논문의 구성은 Ⅰ장에서는 블루투스의 전반적인 동향과 베이스밴드와 HCI의 개요, 연구 방향을 제시하고, Ⅱ장에서는 블루투스 스택의 구성과 기능에 대해 간단히 살펴본다. Ⅲ장에서는 베이스밴드의 각 파트별 역할과 구현 알고리즘에 대해 설명하고, Ⅳ장에서는 두 모듈간의 연결 동작시 발생되는 HCI의 이벤트에 대해 실지적으로 수행한 결과와 비교·확인한다. 그리고, Ⅴ장에서는 Ⅲ장에서 기술한 베이스밴드 각 파트별 구현 결과와 인증시 SAFER+ 알고리즘 개선 방안, 주파수 홉 선택시 지연시간 개선 및 시뮬레이션을 수행하고, 이중 주파수 홉 선택기를 직접 칩으로 구현한다. Ⅵ장에서는 HCI 이벤트 과정을 확인하기 위한 모듈 시스템 연결 회로구성에 대해 살펴보고, 데이터 크기에 따라 거리별 차등을 주어 실제적으로 전송을 수행한 결과인 전송시간과 전송 속도를 통해 시스템 성능을 분석한다. 마지막으로 Ⅶ장에서 본 논문의 결론을 맺는다.

Ⅱ. Bluetooth 프로토콜 구성 및 기능

블루투스 망 위의 통신 시스템을 지원하기 위한 디바이스의 프로토콜 스택은 Fig.1과 같다.

블루투스 코어 프로토콜과 블루투스 라디오는 블루투스 디바이스에 의해 요구되는 사항이고 나머지 프로토콜은 필요할 때만 사용 가능하다.

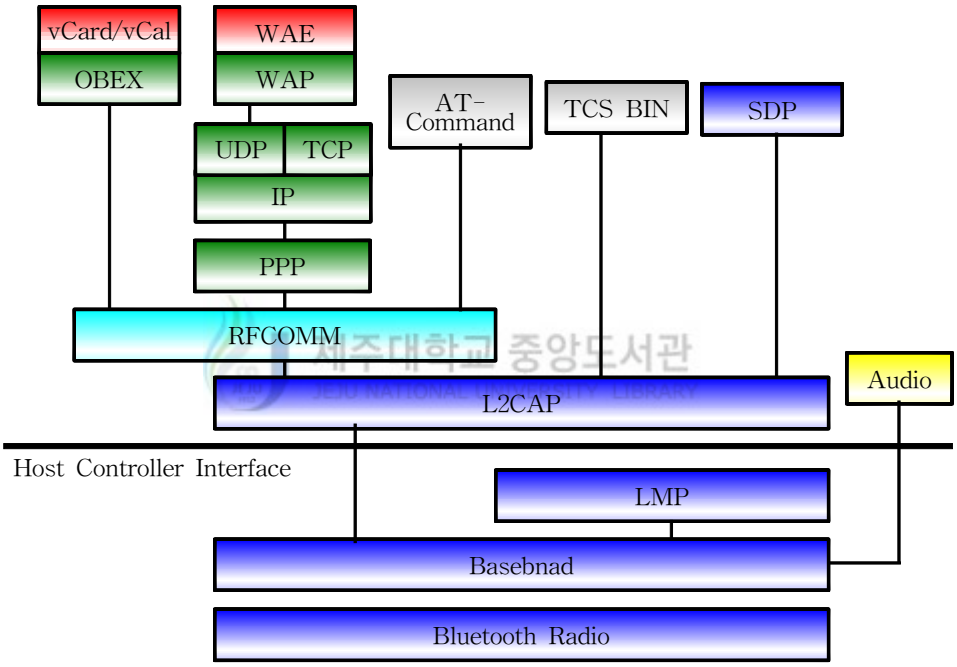


Fig.1 Bluetooth protocol stack

베이스밴드 프로토콜은 호스트에서 내려오는 명령을 받아 내부적으로 처리할 수 있으면 처리하고 외부 RF로 보낼 필요가 있을 때는 LM에 의해 상대 디바이스와 명령을 송수신한 후 결과를 호스트로 올려주는 역할을 한다. 여기에 다시 인증, 암호화 같은 security 측면도 포함된다.

LMP는 블루투스 디바이스들 사이의 연결을 셋업(Set-up)하는 역할을 한다.

HCI는 소프트웨어 사양에 관련된 것으로 블루투스 하드웨어 모듈과 서로 주고

받는 패킷의 포맷과 절차를 정의하고 있다. 즉, 블루투스 모듈이 이해할 수 있는 표준 포맷으로 데이터를 만들어 보내주고, 블루투스 모듈은 그 결과를 표준 패킷으로 만들어 호스트로 보내는 방법을 정의하고 있다. 예를 들어 일반적인 TCP/IP 네트워크에서 IP 패킷 포맷은 통신을 하는 상대편 컴퓨터가 어떤 종류이고 LAN 카드가 어떤 종류인지 몰라도 서로 데이터를 주고받을 수 있듯이 블루투스의 HCI도 누구나 이해할 수 있는 HCI 패킷을 정의함으로써 하드웨어 독립성을 강화했다.

L2CAP(Logical Link Control and Adaptation Protocol) 레이어는 인터넷 프로토콜의 TCP 레이어와 거의 비슷한 역할을 한다. L2CAP는 HCI 레이어 바로 위에 위치하며 상위 프로토콜이나 애플리케이션에게 64 KB까지의 데이터 패킷을 보내고 받을 수 있도록 해준다. 그리고 프로토콜 멀티플렉싱, SAR(Segmentation and reassembly), QoS(Quality of Service), 그룹 전송의 기능을 수행한다. 프로토콜 멀티플렉싱이란 HCI를 통해 데이터가 들어올 때 어느 프로토콜 데이터인지를 구분, 데이터에 적합한 프로토콜로 분배해주는 기능을 말한다. SAR은 L2CAP 상위 레이어에서 전달된 큰 데이터를 HCI 레이어에 전달할 때 작은 조각으로 잘라 전달하고, 반대로 HCI 레이어를 통해 들어오는 작은 데이터의 조각을 원래의 큰 데이터로 복구해 상위 레이어로 전달해 주는 기능이다. QoS는 신뢰성있는 데이터 전송을 말하는 것으로 L2CAP는 커넥션을 확립할 때 QoS정보를 블루투스 디바이스 사이의 L2CAP에서 서로 주고 받는다.

SDP(Service Discovery Protocol)는 블루투스 디바이스가 제공하는 서비스 찾기 기능을 제공하는 프로토콜이다. SDP는 크게 SDP 클라이언트와 서버로 나뉜다. 모든 블루투스 디바이스는 기능에 따라 SDP 서버나 클라이언트 또는 두 기능 모두 포함하고 있어야 한다. SDP가 이렇게 클라이언트와 서버 구조를 가지고 있기 때문에 L2CAP와 마찬가지로 요구(Request)와 응답(Response)구조로 되어있다. SDP 클라이언트에서 어떤 요구를 보내면 서버가 응답하는 구조다. SDP 서버는 내부적으로 소속된 블루투스 디바이스가 제공하는 모든 서비스에 관한 정보를 포함하고 있는 데이터베이스 테이블을 갖고 있기 때문에 클라이언트가 어떤 요구를 하면 이 데이터베이스를 찾아서 답변을 해준다.

Ⅲ. Baseband

베이스밴드는 채널 부호화와 복호화, 단일 데이터 패킷 전송 영역 내에서의 링크 관리 등의 기능을 수행한다. 그것은 순수한 페이로드 데이터에 주소 필드와 링크 제어 필드를 추가하고 오류 검출 및 정정 기능을 제공한다.

장치들은 두 가지 기본 동작모드인 슬레이브나 마스터로 존재하며 피코넷인 소형 네트워크에서 서로간에 통신한다. 피코넷은 마스터가 제어하는 많은 슬레이브들로 구성된다. 장치들 사이에 존재하는 데이터 링크는 오디오와 같은 시간 종속 데이터의 경우 SCO(Synchronous Connection-Oriented)로, 그리고 패킷 기반 데이터의 경우 ACL(Asynchronous ConnectionLess)로 분류된다.

많은 다른 패킷 형태가 존재하며 신뢰성과 데이터 대역폭 사이의 절충이 필요하다. 블루투스는 장치들이 마스터의 전송에 반복적으로 재동기화함으로써 시 동기를 유지하도록 보장한다. 주파수 호핑 알고리즘은 장치 클럭에 기초를 두기 때문에 이것은 또한 주파수 호핑이 보조를 맞추도록 보장한다(Bluetooth Special Interest Group, 1999).

1. Channel Access Code

각 패킷은 채널접근코드로 시작된다. 만약 패킷 헤더가 다음에 오면 채널접근코드는 72bit이고, 그렇지 않으면 채널접근코드는 68bit이다. 이 채널접근코드는 패킷의 도착을 수신자에게 지시하기 위하여 사용되고, 타이밍 동기화와 DC 옵셋 보상과 식별에 사용된다. 채널접근코드는 피코넷(piconet)의 채널로 교환되는 모든 패킷을 식별한다. 즉, 동일한 피코넷에 전송되는 모든 패킷은 동일 채널에 의해 행해진다. CAC(Channel Access Code)는 피코넷의 채널을 특성짓고 채널에서 교환되는 모든 패킷의 preamble을 구성한다.

CAC는 전체 72bit로 preamble, sync word, trailer로 구성된다. 다음 Fig.2는 개체 각각의 비트를 표시한 그림이다.

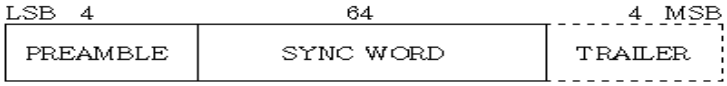


Fig.2 Channel access code format

1) Sync word 구조

Sync word는 24bit 주소(LAP:Low Address Part)에서 얻어진 64bit 코드 워드이다. 즉, 마스터 인자의 LAP에 의해 sync word가 결정된다. 이 구조는 다른 LAP에 기반을 둔 sync word사이의 커다란 헤밍(hamming) 거리를 보장한다. 게다가 sync word의 좋은 자기상관 특성은 타이밍 동기화 과정을 개선시킨다. sync word는 총길이 64bit 임의의 가상 Noise(PN)-열의 중첩인 수정된 블록 코드 (64,30)에 기반을 둔다. 이 PN열은 채널접근코드의 자기상관 특성을 개선시킨다. PN열 생성에 대해서 살펴보면, PN열은 기본 다항식 $h(D)=1+D+D^3+D^4+D^6$을 사용한다. LFSR(Linear Feedback Shift Register)과 그것의 작동 상태는 Fig.3에 나타나 있다.

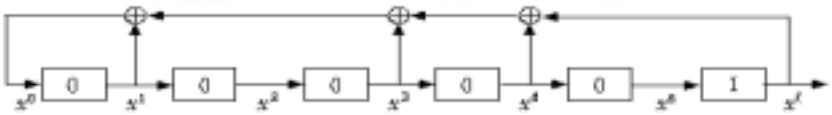


Fig.3 Operation state of LFSR

DC-free 4bit 열 0101과 1010은 식(1)과 같이 쓸 수 있다. 여기서 D^i는 i번째 기기의 지연을 뜻한다.

$$\begin{cases} F_0(D)=D+D^3, \\ F_1(D)=1+D^2, \end{cases} \tag{1}$$

또한, Barker 열은

$$\begin{cases} B_0(D)=D^2+D^3+D^5, \\ B_1(D)=1+D+D^4, \end{cases} \tag{2}$$

식(2)와 같이 정의 내린다. 이것은 길이 7-Barker 열을 생성하는데 사용된다.

2) 채널접근코드 생성 알고리즘

Fig.4는 채널접근코드 생성 단계를 알고리즘으로 간략히 표현한 그림이다.

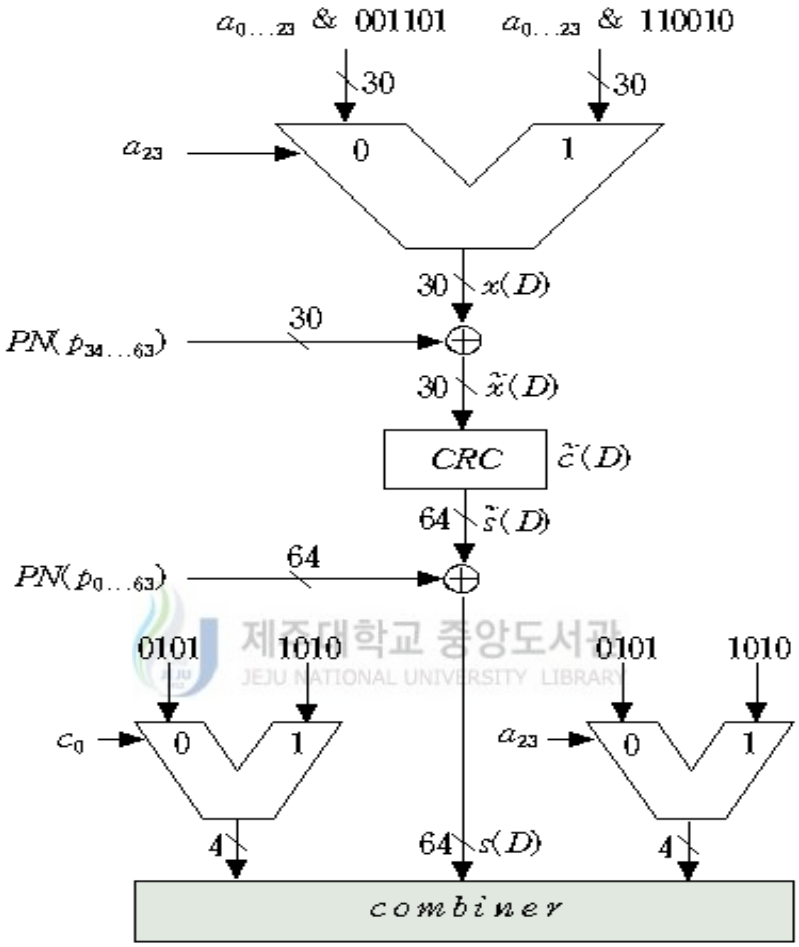


Fig.4 Algorithm of channel access code

채널접근코드를 생성시키기 위해 먼저, 부호화하기 위한 30개의 정보 비트 $x(D)$를 초기화한 후 이 $x(D)$에 PN열을 중첩시켜 $\tilde{x}(D)$를 구한다. (64,30) 수정 블록 코드의 parity 비트 $\tilde{c}(D)$를 생성하여, $\tilde{x}(D)$와 합쳐서 code word $\tilde{s}(D)$를 생성한다. 그리고, 생성된 $\tilde{s}(D)$에 PN열을 중첩시켜 $s(D)$를 생성한다.

최종적으로 preamble과 trailer를 추가하여 채널접근코드를 완성할 수 있다.

2. 오류 검출

블루투스에서는 채널접근코드, 헤더의 HEC(Header Error Check), 정보의 CRC(Cyclic Redundancy)를 단계별로 검사함으로써 수신 패킷의 에러 또는 잘못된 전송을 체크할 수가 있다. 패킷을 수신했을 때 가장 먼저 채널접근코드가 체크되는데, 이의 수행은 LAP가 정확한지를 체크하여 다른 피코넷으로부터의 패킷 수신을 예방하게된다.

HEC와 CRC는 데이터의 에러와 잘못된 주소를 체크하기 위해 사용되는데 UAP(Upper Address Part)가 HEC와 CRC를 구하는데 초기값으로 사용된다. HEC와 CRC는 일반적으로 쓰이는 제산회로를 사용하여 구현할 수 있다. HEC의 생성은 헤더의 HEC를 제외한 데이터 부분이 10비트이고 HEC가 8비트이므로 식(3)과 같은 8비트 다항식을 사용한다. Fig.5는 이에 따른 CRC 생성회로를 보여주고 있다.

$$g(D) = (D+1)(D^7 + D^4 + D^3 + D^2 + 1) = D^8 + D^7 + D^5 + D^2 + D + 1 \qquad (3)$$

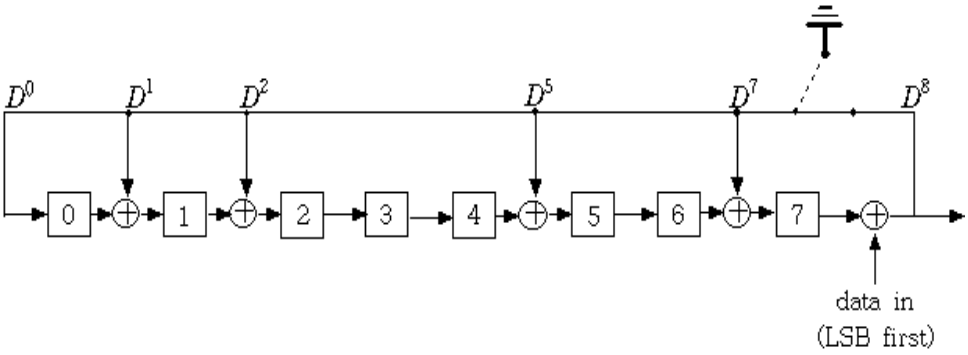


Fig.5 The LFSR circuit generating the HEC

CRC의 생성에서는 일반적으로 16비트 다항식이나 32비트 다항식이 사용되며, 식(4)는 16비트 다항식을 나타내며 Fig.6 은 이에 따른 CRC 생성회로를 보여주고

있다.

$$g(D) = D^{16} + D^{12} + D^5 + 1 \tag{4}$$

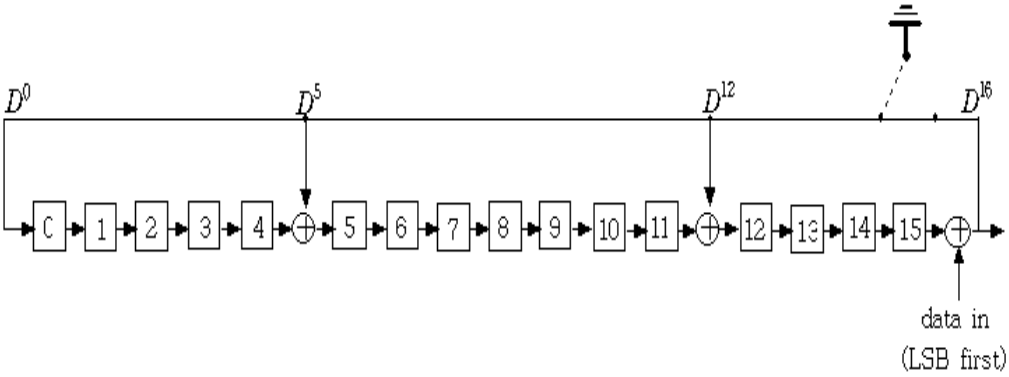


Fig.6 The LFSR circuit generating the CRC

3. 보안

케이블에 기반하여 기본적인 보안이 제공되는 유선통신과는 다르게 블루투스는 무선통신이므로 손쉽게 도청이 가능하고, 도청시 인지가 불가능하다는 단점이 있다. 따라서 다른 곳으로부터의 침입을 방지하는 것이 중요한 문제라고 할 수 있다.

원하는 디바이스와의 연결만을 보장하고 전송하고자 하는 정보를 보호하기 위해 블루투스에서는 암호화(Encryption)와 인증(Authentication)을 제공하고 있다. 블루투스는 128비트 키까지를 사용하는 SAFER+(Secure And Fast Encryption Routine) 암호화기를 갖는 강력한 보안 형태를 갖는다.

그래서, 암호화기는 기본적으로 SAFER+ 알고리즘을 사용하고 있고, 이를 위해선 4개의 입력값이 들어가게 되는데, 48비트의 디바이스 주소, 인증 과정에서 사용될 인증키, 암호화 과정에서 사용되는 암호화키, 그리고 블루투스 자체에서 생성되는 128비트의 RAND(random number)가 그것이다.

Fig.7은 블루투스에서의 전체적인 보안 시스템을 보여주고 있다.

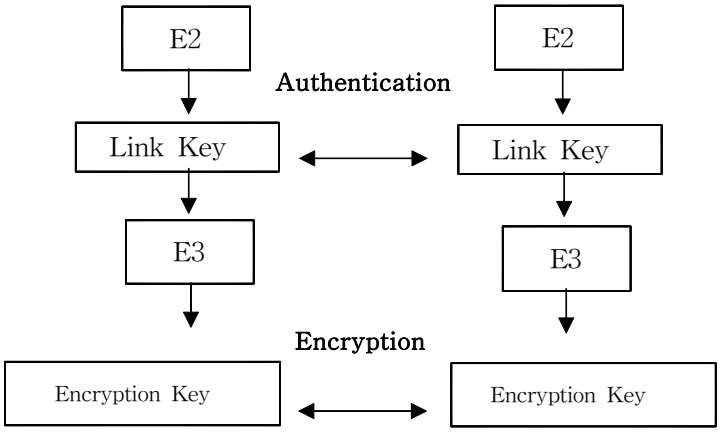


Fig.7 Key control

1) 인증

블루투스 인증은 본질적으로 challenge - response 방법을 사용한다. 여기서 상대편이 공유되는 비밀키를 알고 있는지 체크하기 위해 2-move protocol이 사용되는데, 기본적으로 이 프로토콜은 두 디바이스가 같은 키를 갖고 있는지, 그리고 그들이 인증 과정을 성공적으로 수행했는지를 체크하게 된다.

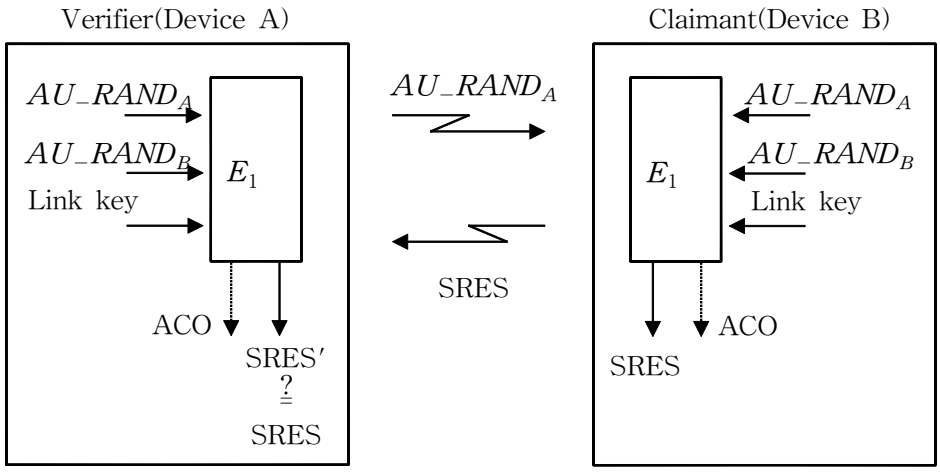


Fig.8 Description of the authentication process

이 인증 과정에서 생성된 ACO(Authenticated Ciphering Offset)값은 양쪽 디바

이스에 저장되고 나중에 암호화키를 만드는데 사용된다. Fig.8은 두 디바이스 사이에서의 인증 과정을 보여주고 있다.

2) 링크키의 종류에 따른 생성

블루투스 디바이스간의 인증을 위해서 사용되는 링크키(Link key)에는 다양한 유형이 있고 각 키는 다른 방법으로 생성된다. 링크키는 SAFER+ 알고리즘의 "E" 구현을 통해 생성된 128비트의 수로서, 반영구적으로 쓰이거나 일시적으로 사용된다. 그 종류는 다음과 같다.

① 단위키(Unit key) : 반영구적인 키로서 비휘발성 메모리에 저장되며, 공장 출하시 처음 설정되고 특별한 이유가 없을 경우 거의 변하지 않는다.

② 조합키(Combination key) : 블루투스 장치간의 단위키를 서로 교환하고 두 개의 단위키를 조합하여 링크키로 사용하는 경우이다. 이 키는 두 블루투스 장치간에 더 많은 보안을 요구할 때 사용한다.

③ 마스터키(Master key) : 이 키는 다중 연결 설정시 사용하며 현재의 링크키를 대체하는 임시키이다.

④ 초기화키(Initialization key) : 128비트로 하나의 세션에 대하여 사용되는 링크키이고 unit이 초기화될 때 사용된다. 이 키는 사용자의 PIN(Personal Identification Number) 입력값을 이용하여 생성되는데 보통 4자리 숫자로 입력받게 된다.

⑤ 암호화키(Encryption key) : 마지막으로 현재의 링크키로부터 유도되는 암호화키가 있다. 이는 암호화기, E_0에서 암호 열을 생성하기 위하여 이 키(K_c)를 사용한다.

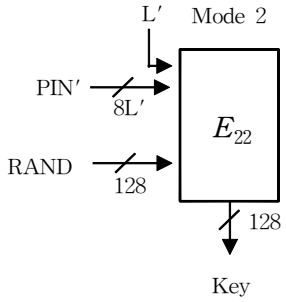
이렇게 여러 가지의 키를 사용하는 이유는 각 디바이스마다 메모리 저장 공간의 많고 적음 여부, PIN을 입력받아 사용할 것인가 아니면 고정 PIN을 사용할 것인가, 일대일 통신인지 멀티캐스팅을 사용하는지에 따라 유동적으로 키를 사용하기 위해서이다.

마스터키와 초기화키를 생성하기 위해서는 E_{22} 알고리즘이 사용된다. E_{22} 알고리즘은 PIN 코드값을 사용하는데, 고정적인 값을 사용할 수도 있고 임의적으로 입력받아 사용할 수도 있다. 이외에도 요청자 디바이스의 디바이스 주소, 확인자 디바이스에 의해 생성되는 128비트의 RAND가 입력값으로 들어가고, 128비트의 출력키값이 나오게 된다.

$$E_{22}:(\Pi N', RAND, L') \rightarrow Ar'(X, Y)$$

$$X = \bigcup \Pi N'[i(\bmod L')]$$

$$Y = RAND[0 \ldots 14] \bigcup (RAND[15] \oplus L')$$

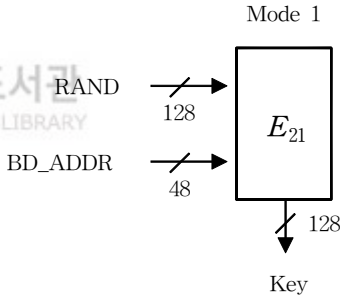


단위키와 조합키는 E_{21} 알고리즘을 이용하여 생성된다. 이때는 디바이스 주소와 RAND의 적절한 조합을 통해 키의 생성이 이루어진다.

$$E_{21}:(RAND, address) \rightarrow Ar'(X, Y)$$

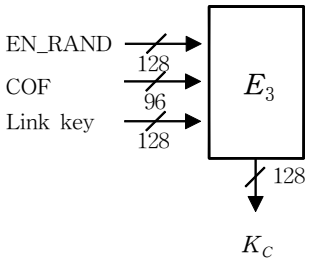
$$X = RAND[0 \ldots 14] \bigcup (RAND[15] \oplus 6)$$

$$Y = \bigcup address[i(\bmod 6)]$$



마지막으로 암호화키를 생성하는 E_3 알고리즘이 있다. 암호화키는 현재의 링크키와 96비트의 COF(Ciphering Offset Number), 그리고 128비트의 RAND를 통해 만들어진다. 이때 COF는 인증 과정에서 생성되는 ACO에 기초를 두고 있다.

$$E_3: (K, RAND, COF) \rightarrow Hash(K, RAND, COF, 12)$$



3) 인증을 위한 E1 알고리즘 구현

인증을 위한 함수 E1은 Fig.9에 나타나있듯이, SAFER+라고 불리워지는 Ar과 Ar의 변형 형태인 Ar′으로 구성되어 있다. E1의 입력으로는 링크키, RAND 그리고 디바이스 주소값이 들어가고 최종적으로 SRES(Signed Response)와 ACO가 만들어지게 된다.

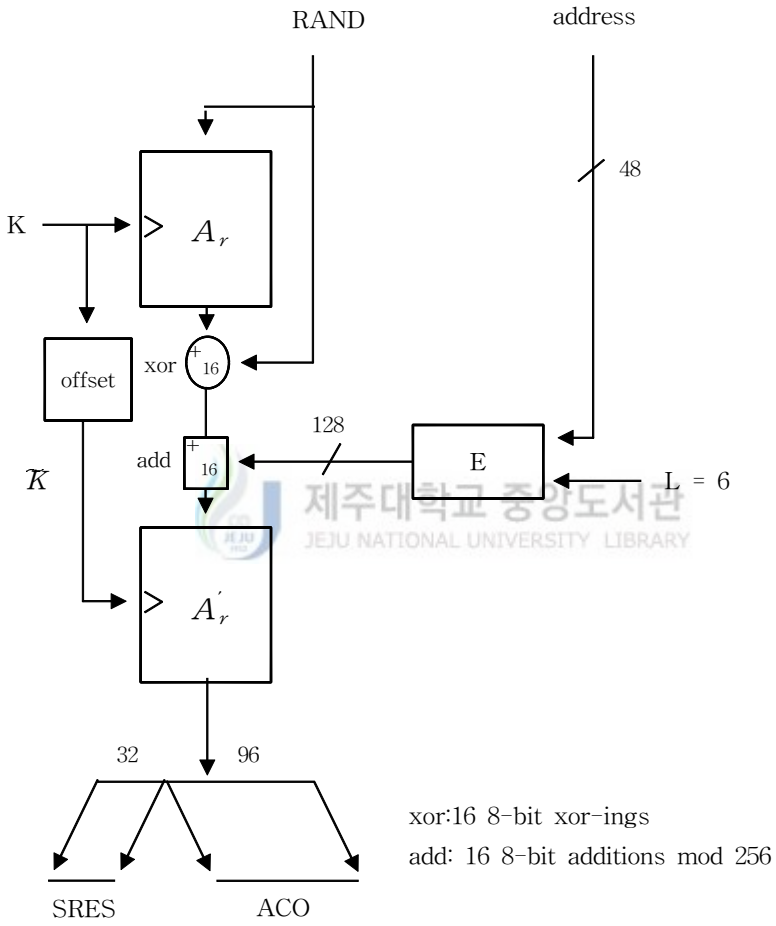


Fig.9 Flow of Data for the Computation of E1

Ar로 표현되어지는 SAFER+는 1998년을 기점으로 표준 기한이 만료된 DES(the Data Encryption Standard)의 대안으로 개발 중인 AES(Advanced Encryption Standard)의 후보 알고리즘 중 하나로서 그 구조는 Fig.10과 같다.

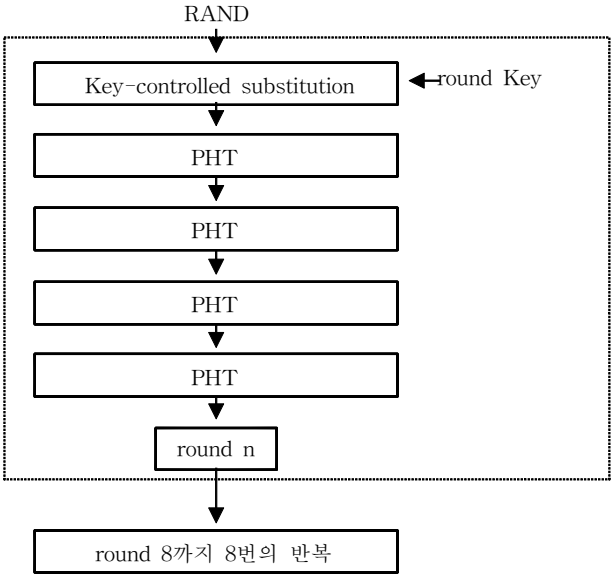


Fig.10 SAFER+ Algorithm

4. 암호화

케이블에 기초한 유선통신과 달리 블루투스에서 보안성 보장은 링크레벨에서 암호화기를 구현함으로써 가능하다. 블루투스 정보의 암호화는 모든 정보를 재동기화하는 E_0라 불리는 열 암호화기(stream cipher)에 의해 수행된다.

Fig.11은 이 과정을 수행하는 E_0에 대한 그림이다.

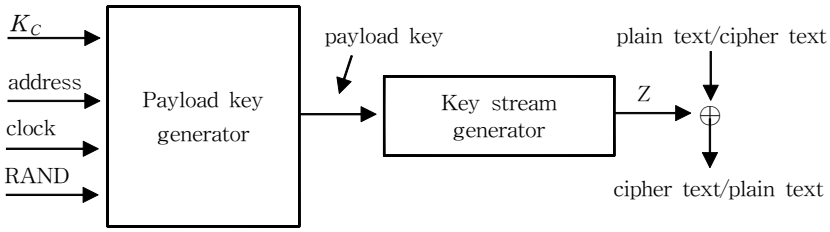


Fig.11 Stream ciphering for Bluetooth with E_0

열 암호화기는 세 부분으로 구성되는데 첫 번째 부분에서 초기화를 형성하고, 두 번째 부분에서 키 열 비트(key stream bit)를 형성하여 마지막 부분에서 암호화와 복호화를 수행한다.

초기화한 후 첫 번째 부분에서, 암호화기는 다음 4종류의 입력이 필요하다.

① 암호화 또는 암호 해석을 위한 수(즉, 장치들간의 보내지는 데이터. RAND)
② 마스터의 블루투스 장치 주소(address)
③ 마스터의 블루투스 슬롯 클럭(clock)
④ 양쪽 장치에 의하여 분배되는 비밀 키(K_c)

그리고, 키 열 비트를 형성하기 위한 암호화기에 쓰인 각각의 LFSR은 식(5)와 같다.

$$\begin{aligned} LFSR_1 &= t^{25}+t^{20}+t^{12}+t^{8}+1 \\ LFSR_2 &= t^{31}+t^{24}+t^{16}+t^{12}+1 \\ LFSR_3 &= t^{33}+t^{28}+t^{24}+t^{4}+1 \\ LFSR_4 &= t^{39}+t^{36}+t^{28}+t^{4}+1 \end{aligned} \tag{5}$$

위 4개의 LFSR에서 사용된 레지스터들의 전체길이는 128이고, 각 단계에서의 구현은 식(6)을 이용한다. 이때 x_t^i는 $LFSR_i$의 t^{th}심벌을 의미한다.

$$\begin{aligned} y_t &= \sum_{i=1}^{4} x_t^i \\ z_t &= x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus c_t^0 \in \{0,1\} \\ z_t &= x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus c_t^0 \in \{0,1\} \\ c_{t+1} &= (c_{t+1}^1, c_{t+1}^0) = s_{t+1} \oplus T_1[c_t] \oplus T_2[c_{t-1}] \end{aligned} \tag{6}$$

여기서, $T_1 : (x_1, x_0) \rightarrow (x_1, x_0)$
$T_2 : (x_1, x_0) \rightarrow (x_0, x_1 \oplus x_0)$ 이다.

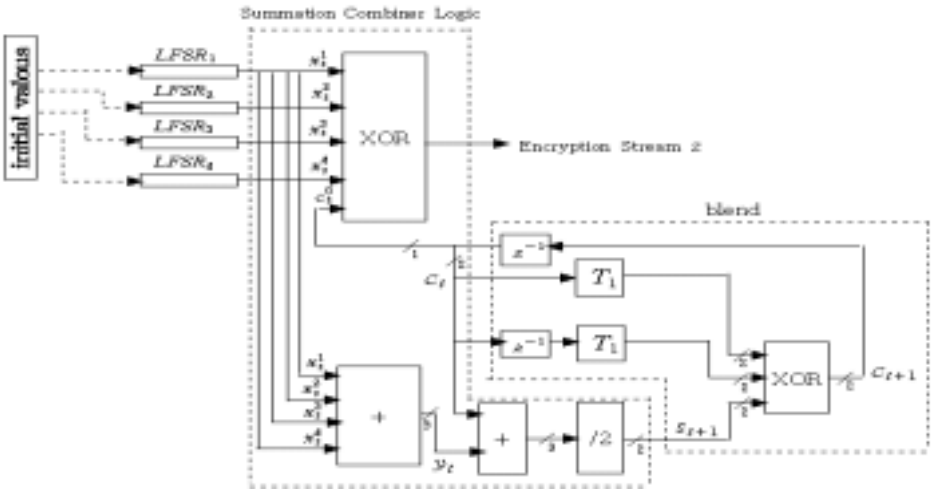


Fig.12 Concept of the encryption engine

Fig.12는 키 열 생성기로 사용되는 암호화기의 개념도이다.

5. 주파수 호핑 선택

마스터, 슬레이브 기기간의 간섭 현상을 줄이기 위해서는 스팩트럼 확산 방식을 이용할 수 있다. 통신중에 무선 송수신기는 의사 랜덤(pseudo random)방법으로 한 채널에서 다른 채널로 도약을 하는데, 순간적인 홉의 대역폭은 작지만 전체 주파수 대역에 대하여 확산이 이루어진다. 이런 스팩트럼 확산에 따라 피코넷에 있는 모든 기기는 호핑 채널을 따르기 위하여 마스터 기기의 주소와 클럭을 사용한다. 통신장치가 피코넷에 참여하지 않을 때는 대기상태가 되며 주기적으로 호출 메시지를 체크한다. 수신된 비트의 대부분이 접근코드와 일치하면, 장치는 자신을 활성화시키고 연결설정 절차를 취한다. 호출장치와 수신장치가 같은 통화대기 캐리어를 선택하면, 수신장치는 접근코드를 수신하고 승인신호를 되돌려 보낸다. 그런 후에 호출장치는 그 장치의 주소와 현재 클럭을 포함하고 있는 패킷을 전송한다. 수신장치는 이 패킷을 승인한 후에, 각각의 장치는 홉 선택을 위하여 호출장치

의 파라미터를 상용하여 호출장치가 주로 동작하는 피코넷을 형성한다.

Fig.13은 79 hop 시스템에서의 홉 선택 블록 다이어그램이다.

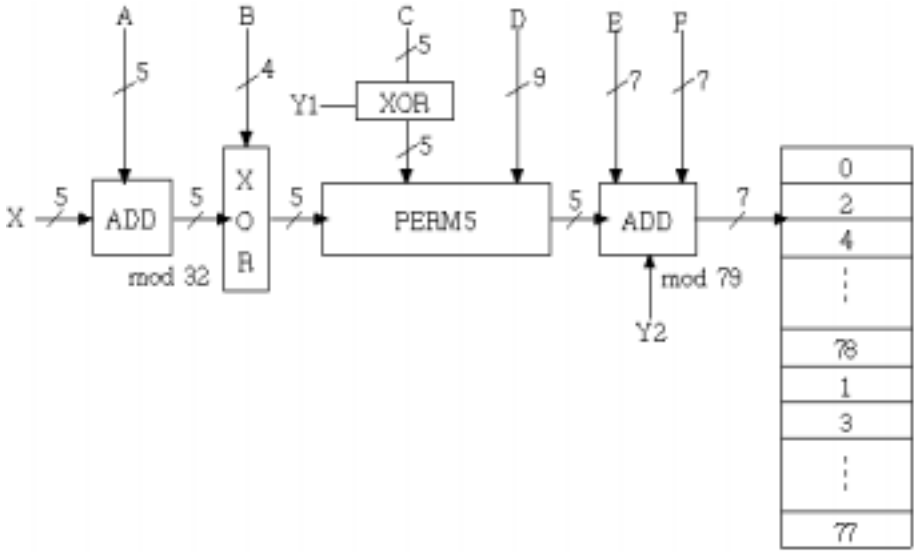


Fig.13 Block diagram of hop selection for the 79-hop system

홉 선택을 위해 32홉 세그먼트에서 입력 X가 정해지고 Y1, Y2는 마스터에서 슬레이브, 슬레이브에서 마스터로의 전송사이에서 선택된다. A에서 D까지는 세그먼트내의 명령으로 정해지고, E와 F는 호핑 주파수의 사상으로 결정된다. 커널은 호핑 주파수에 포함된 레지스터에 신청한다. 이런 목록은 먼저 모든 짝수 호핑 주파수를 작성하고난 다음 모든 홀수 호핑 주파수를 작성한다. 홉 시퀀스는 다음의 세 가지 종류의 클럭사이에서 판별된다.

① CLK_{27-0} : 현재의 피코넷의 마스터 클럭

② $CLKN_{27-0}$: 기기의 원래 클럭

③ $CLKE_{27-0}$: 호출된 기기의 원래 클럭에서 측정된 클럭

CLKE는 수신의 근사화를 위해 옵셋과 호출자의 CLKN을 합한 것이다.

79-hop 시스템에서 입력 X는 다음의 경우에 따라 결정된다.

Table 1. Control for 79-hop system

	Page scan/ Inquiry scan	Page/Inquiry	Page response (master/slave)and Inquiry response	Connection state
X	$CLKN_{16-12}$	$X^{(79)}_{P_{4-0}}$ / $X^{(79)}_{i_{4-0}}$	$X^{(79)}_{prm_{4-0}}$ / $X^{(79)}_{prs_{4-0}}$	CLK_{6-2}
Y1	0	$CLKE_1$ / $CLKN_1$	$CLKE_1$ / $CLKN_1$	CLK_1
Y2	0	$32\times CLKE_1$ / $32\times CLKN_1$	$32\times CLKE_1$ / $32\times CLKN_1$	$32\times CLK_1$
A	A_{27-23}	A_{27-23}	A_{27-23}	$A_{27-23}\oplus CLK_{25-21}$
B	A_{22-19}	A_{22-19}	A_{22-19}	A_{22-19}
C	$A_{8,6,4,2,0}$	$A_{8,6,4,2,0}$	$A_{8,6,4,2,0}$	$A_{8,6,4,2,0}\oplus CLK_{20-16}$
D	A_{18-10}	A_{18-10}	A_{18-10}	$A_{18-10}\oplus CLK_{15-7}$
E	$A_{13,11,9,7,5,3,1}$	$A_{13,11,9,7,5,3,1}$	$A_{13,11,9,7,5,3,1}$	$A_{13,11,9,7,5,3,1}$
F	0	0	0	$16\times CLK_{27-7} \mod 79$

Ⅳ. Host Controller Interface

HCI는 블루투스 프로토콜 소프트웨어 중에서 가장 중요하고, 블루투스의 특성과 가장 밀접하게 관련된 부분이다.

블루투스 장치는 두 부분으로 즉, 스택의 상위계층(L2CAP와 그 이상)을 구현하는 호스트와 하위 계층(LMP와 그 이하)을 구현하는 모듈로 구성할 수 있다. HCI는 블루투스 호스트와 그의 모듈간에 균일한 인터페이스를 제공한다. HCI는 표준화되었기 때문에, 호스트 소프트웨어는 다양한 제조 업체의 블루투스 장치에 공통으로 사용할 수 있다(Bluetooth Special Interest Group, 1999)(Jennifer Bray and Charles F Sturmam, 2001).

1. HCI 패킷 형태

HCI에 관한 블루투스 표준은 다음과 같이 정의내릴 수 있다.

◀ 모듈을 제어하기 위해 호스트가 사용하는 명령어 패킷
◀ 호스트에게 하위계층에서의 변화를 알리기 위해 모듈이 사용하는 이벤트 패킷
◀ 호스트와 모듈간에 음성과 데이터를 전송하는 데이터 패킷
◀ HCI 패킷을 전송하는 트랜스포트 계층패킷

HCI는 세 가지 형태의 패킷을 사용한다.

호스트에서 모듈로 가는 명령어, 모듈에서 호스트로 가는 이벤트, 양방향으로 이동하는 데이터 패킷이다. 호스트와 모듈사이에 이들 세 가지 형태의 패킷은 블루투스 모듈을 완벽하게 제어하고 요구된 모든 데이터를 전달하는데 사용할 수 있다. HCI 패킷 흐름의 방향을 그림으로 표현하면 Fig.14와 같다.

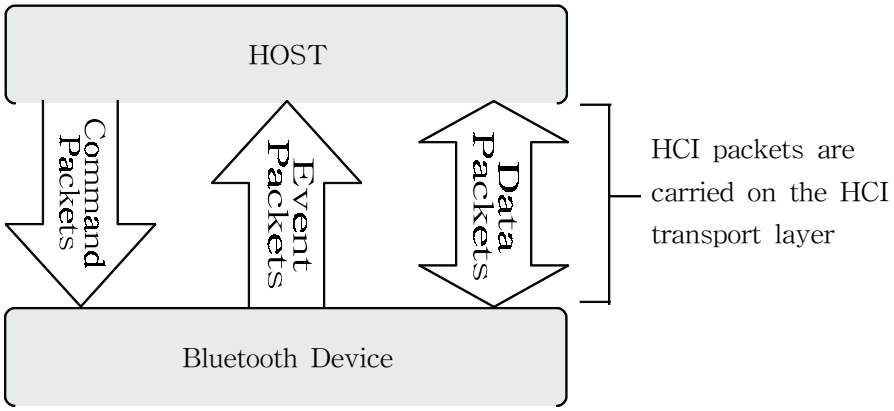


Fig.14 The three types of HCI packets

2. HCI 명령어

블루투스 모듈을 제어하고 블루투스 모듈상태를 감시하기 위해 호스트는 HCI 명령어를 사용한다. 명령어는 HCI 명령어 패킷을 사용하여 전송한다. 호스트에서 호스트 컨트롤러 방향으로 전달되는 패킷으로, 실제 블루투스 모듈에 어떤 명령을 전달하면 모듈이 이 패킷을 해석해 수행하게 된다.

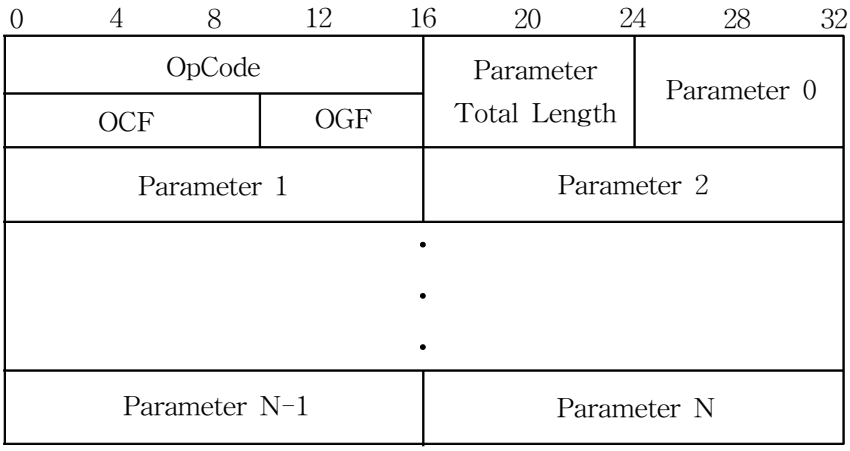


Fig.15 HCI command packet

HCI 명령어 패킷구조는 Fig.15와 같다. HCI 명령어 패킷은 다른 명령어와 구별하기 위해 명령어 종류를 나타내는 두 개의 바이트 OpCode로부터 시작한다.

OpCode의 6 비트는 OGF(Opcode Group Field)로 명령어 그룹을 나타낸다. OpCode의 10 비트는 OCF(Opcode Command Field)를 차지하며 그룹내의 명령어를 나타낸다.

OpCode 다음에 바이트 단위의 후속 파라미터 길이를 나타내는 단일 바이트 필드가 있다. 파라미터들은 단일 바이트 필드 다음에 오며, 각 파라미터는 정수개의 바이트를 차지하는데 모두 단일 바이트가 아니므로, 파라미터 총 길이는 파라미터 총수와 반드시 같지는 않다.

명령어 패킷을 C 구조로 나타내면 다음과 같다.

```
typedef suruct{
union{
struct{
unsigned short int OCF : 10 ;     //Opcode Command Field
unsigned short int OGF : 6 ;      //Opcode Group Field
};
unsigned short int OpCode;
};
unsigned char ParameterTotalLength;
추가 파라미터 정의들.....;
}COMMAND_PACKET;
```


3. HCI 이벤트 패킷

호스트에서 보내는 명령어에 대한 결과나 모듈 상태 변화를 호스트에 알리기 위

해 블루투스 모듈인 호스트 컨트롤러에서 호스토로 보내는 패킷이다.

Fig.16과 같이 HCI 이벤트 패킷 구조는 HCI 명령어 패킷과 비슷하다. 이벤트 코드는 HCI 명령어를 나타내는 OpCode 필드와 비슷한 기능을 한다.

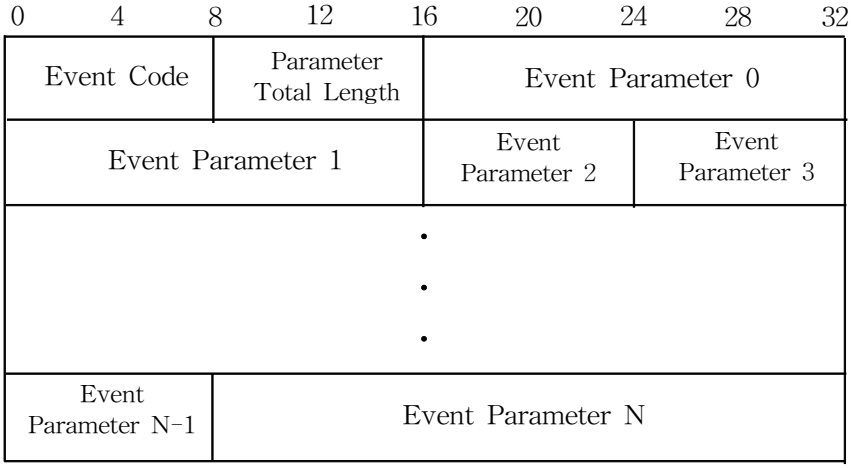


Fig.16 Structure of an HCI event packet

이벤트 패킷을 명령어 패킷과 연관지어 살펴보면, 호스트 컨트롤러는 호스트가 보낸 커맨드의 수행을 완료했을 때 완료 사실을 호스트에 알리기 위해 약 40여개의 정의된 이벤트 중에서 Command Complete Event를 보낸다. 이렇게 단순하게 한 명령어 완료에 대한 응답으로 하나의 이벤트가 발생하는 경우는 대부분 블루투스 모듈의 내부적인 설정이나 상태를 바꾸기 위한 명령어를 사용할 때다.

이와 달리 하나의 명령어에 의해 두 개 이상의 이벤트가 발생하는 경우도 있다. 이런 명령어는 대부분 RF에서 데이터가 상대방의 블루투스 디바이스와 교환되는 경우에 발생한다. 호스트가 명령어를 호스트 컨트롤러에 보낸 후 바로 Command Status Event가 호스트로 전송돼 명령어를 호스트 컨트롤러가 정상적으로 접수했다는 것을 알려주고 실제 명령 수행으로 들어간다.

명령어가 정상적으로 완료된 후, 블루투스 모듈은 완료 상태를 호스트에 알리기 위해 명령어에 적합한 complete event를 호스트로 전송해 명령어의 처리가 완전히 끝났음을 알린다.

4. HCI 데이터 패킷

HCI 데이터 패킷은 실제 상위 프로토콜의 데이터를 실어 나르는 역할을 하는 패킷으로 접속이 이루어진 후에는 데이터 패킷이 가장 중요한 역할을 하게 된다.

HCI를 통하여 데이터(ACL)와 음성(SCO)을 전송하는데 각각 다른 HCI 데이터 패킷을 사용한다.

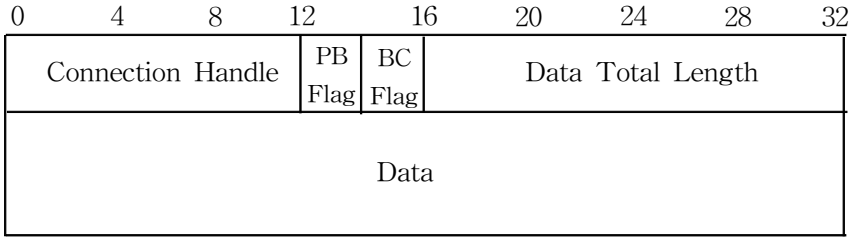


Fig.17 HCI ACL data packet

Fig.17은 ACL 데이터 전송에 사용되는 HCI 데이터 패킷을 나타낸 것이다. 이 패킷에서 접속 핸들(Connection Handle)은 ACL 접속을 구별하기 위한 ID로 사용된다. PB 플랙(Packet Boundary flag)은 패킷 데이터가 상위계층 L2CAP 패킷의 시작을 전송하는지 아니면 L2CAP 패킷의 일부인지를 나타낸다.

예를 들어, 2KB 의 데이터를 HCI 데이터 패킷으로 전송할 때 이것을 한번에 보내기에는 너무 크기 때문에 여러 개의 조각으로 잘라 보내야 한다. 만약, 8개의 256 바이트로 잘라 HCI 데이터 패킷으로 보낼 경우, 8개의 조각중에서 첫 번째 조각을 포함하는 HCI 데이터 패킷만이 이 PB 플랙이 1로 설정되고 나머지는 0으로 설정된다. 이렇게 함으로써 데이터를 받아들이는 상대편 블루투스 디바이스가 이 비트를 체크해 이 플랙이 설정돼 있다면 패킷의 시작으로 인식한 다음, 이 플랙이 설정될 때까지 데이터를 받아 하나로 묶어 처리하게 된다. BC 플랙(Broadcast flag)은 방송데이터의 각 점에서의 데이터를 나타내고, 액티브 방송과 피코넷 방송을 구별한다.

Fig.18은 SCO 데이터를 전송하는데 사용되는 HCI 패킷을 나타낸 것으로 그 구조는 ACL 패킷과 아주 비슷하다.

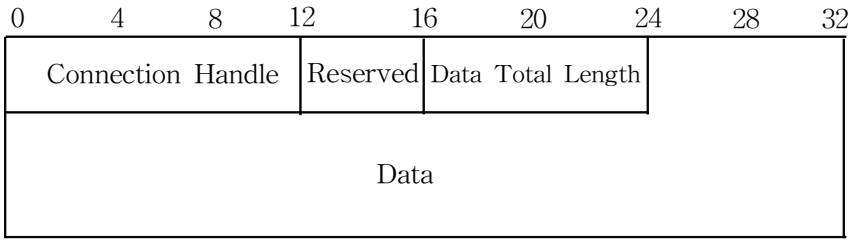


Fig.18 HCI SCO data packet

SCO 데이터 패킷은 ACL 패킷과 비교해 보면, PB와 BC 플랙이 없고 데이터 총 길이 필드는 한 바이트이고 데이터 필드를 255 바이트로 제한하는 것에서 차이가 난다.

이와 같이 데이터 길이 필드를 작게 하는 이유는 SCO 링크가 양방향 오디오용이기 때문이다. 과다한 데이터가 한 패킷 내에서 전송되면, SCO 링크의 끝과 끝 사이에 시간지연이 증가된다. SCO 데이터 패킷은 블루투스에서 대부분 오디오 데이터를 전송하는 목적으로 사용되는데, 이 오디오 처리는 대부분의 블루투스 디바이스가 하드웨어적으로 처리하고 있기 때문에 특별히 소프트웨어에서 다루는 일은 없다.

5. HCI 트랜스포트 계층

트랜스포트 계층은 호스트로부터 HCI 패킷을 받아 블루투스 모듈로 보내는데 필요하다. 블루투스는 세 개의 트랜스포트 계층을 정의한다.

① USB (Universal Serial Bus)
② RS232C : 에러 정정 기능이 있는 직렬 인터페이스

③ UART (Universal Asynchronous Receiver Transmitter)

: 에러 정정 기능이 없는 직렬 인터페이스

RS232C를 살펴보면, RS232C는 직렬 링크이며, Table.2와 같은 패킷 형태를 나타내기 위해 프레임화를 하여 패킷 형태를 구분하는데 용이하다.

Table.2 HCI RS232C Packet Header

HCI packet type	HCI packet indicator
HCI Command Packet	0x01
HCI ACL Data Packet	0x02
HCI SCO Data Packet	0x03
HCI Event Packet	0x04
Error Message Packet	0x05
Negotiation Packet	0x06

RS232C 트랜스포트 계층은 Fig.19와 같이 널-모뎀(null_modem connections)을 사용한다. TXD(Transmit data)는 RXD(Receive data)에 접속되고, CTS(Clear To Send)는 RTS(Ready To Send)에 접속된다.

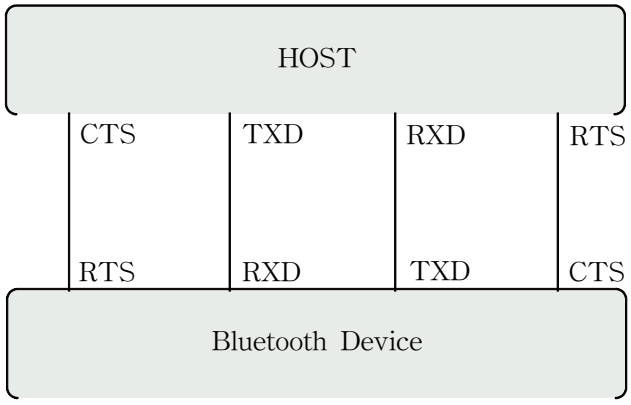


Fig.19 Connections for HCI RS232C Transport layer

6. RS232C 연결시 동작 과정

RF단과 RS232C로 상호 연결한 시스템으로 구현하여, 완성된 두 모듈간의 연결까지 동작을 HCI 이벤트를 실제로 수행하여 그 과정을 하나하나 살펴본다. 각 모듈에서 마스터와 슬레이브의 역할을 규정짓기 위하여 우선 다음과 같이 초기화한다.

Initialising HCI
HCI EMULATOR OFF
Initialising L2CAP
Initialising bt buffer (65536)
Initialising TEST

모듈이 각각 마스터와 슬레이브로 동작하기 위하여, 다음과 같은 이벤트 패킷을 보내어 실행한다.

send_cmd : 0x5 0x10 0x0
data : 0e 0b 01 05 10 00 80 00 40 08 00 08 00
command complete
length = 11, result = 0, num_hci_cmd = 1, ogf = 04, ocf = 05
process_return_param: **READ_BUFFER_SIZE**
process_return_param: acl_len: 128, acl_nbr : 8

HCI 인터페이스를 오가는 데이터 흐름제어에 쓰이는 방법에는, 블루투스 모듈이 가지는 데이터 버퍼 크기(buffer size)를 알려면 HCI_Read_Buffer_Size 명령어를 사용한다. 블루투스 모듈은 그 모듈이 비축할 수 있는 SCO 패킷과 ACL 패킷의 개수, 그리고 HCI SCO 데이터 패킷과 HCI ACL 데이터 패킷의 최대 크기를

호스트에게 알려준다. 이처럼, 블루투스 모듈이 비축할 수 있는 SCO와 ACL 패킷의 수를 알면, 호스트는 블루투스 모듈에게 HCI를 오가는 패킷 숫자만큼 전송할 수 있다. 일단 호스트가 모듈 버퍼를 채울 만큼 충분히 패킷을 전송하고 나면, 호스트는 모듈이 더 이상의 버퍼공간이 있다고 알려줄 때까지 기다려야 한다.

그리고, Fig.20에서 보듯이 명령어가 블루투스 모듈에 전송될 때마다, 모듈은 HCI_Command_Complete 또는 HCI_Command_Status로 응답한다. 명령어가 즉시 실행될 수 있을 때는 HCI_Command_Complete를 사용하고, 그렇지 못한 경우에는 HCI_Command_Status를 사용하고 명령이 끝날 때 HCI_Command_Complete를 전송한다.

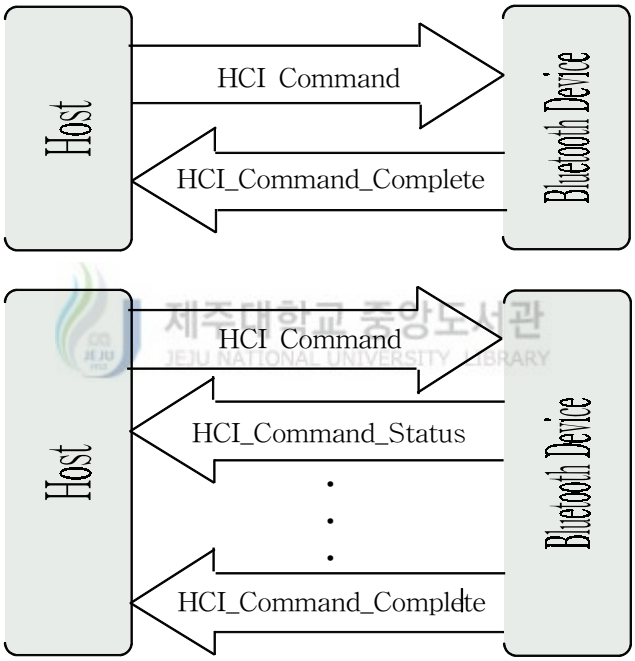


Fig.20 Message sequence charts for HCI command flow control

send_cmd : 0x1e 0xc 0x4 0x40 0x0 0x20 0x0

data : 0e 04 01 1e 0c 00

command complete

length = 4, result = 0, num_hci_cmd = 1, ogf = 03, ocf = 1e

process_return_param: **WRITE_INQUIRYSCAN_ACTIVITY**

호스트는 명령어 HCI_Write_Inquiryscan_Activity를 사용하여 한 스캔의 시작과 다음 스캔의 시작 사이의 간격을 설정한다. 이 명령어는 각 스캔의 지속시간인 윈도우를 설정하는데도 사용한다. 만약 이 명령어가 장치에 보내지지 않으면, 스캔 시작간의 기본 간격은 1.28s이고, 기본 윈도우는 11.25ms이다. 만일 이 간격과 윈도우가 같게 설정되면, 장치는 스캔을 연속적으로 할 것이다. 정상적으로 이것은 이미 설정된 접속이 없는 무 건전지 장치에서만 사용될 수 있다.

send_cmd : 0x1a 0xc 0x1 0x1
write_scan_enable : 0x1a 0xc 0x1 0x1
length = 6
data : 0e 04 01 1a 0c 00
command complete
length = 4, result = 0, num_hci_cmd = 1, ogf = 03, ocf = 1a
process_return_param: WRITE_SCAN_ENABLE

스캐닝은 HCI_Write_Scan_Enable 명령어를 사용하여 수행할 수 있다. 그리고, 명령어 HCI_Read_Scan_Enable은 스캐닝이 가능 또는 불가능한지를 수시로 검사하는데 사용된다. 이 명령어는 조회 스캐닝과 호출 스캐닝 상태를 회송한다. 여기까지는 마스터와 슬레이브 두 모듈간의 공통적인 동작상태를 보여 주고 있다.

이렇게 각 모듈을 마스터와 슬레이브로 동작시킨 후, 두 모듈을 연결하기 위해서는 우선 마스터에서는 슬레이브로 동작되는 모듈을 찾기 위하여 inquiry 명령이 더 수행된다.

send_cmd : 0x1 0x4 0x5 0x33 0x8b 0x9e 0xa 0x0
seqcnt = 6
data : 0f 04 00 01 01 04
command status
length = 4, result = 00, num_hci_cmd = 1, ogf = 01, ocf = 01

data : 02 0f 01 (dd 17 00 5b 02 00) 01 00 00 00 00 00 18 4a

process_event_packet: **INQUIRY_RESULT**

각 HCI_Inquiry_Result 이벤트는 응답장치 접속에 필요한 정보를 가지고 있다.

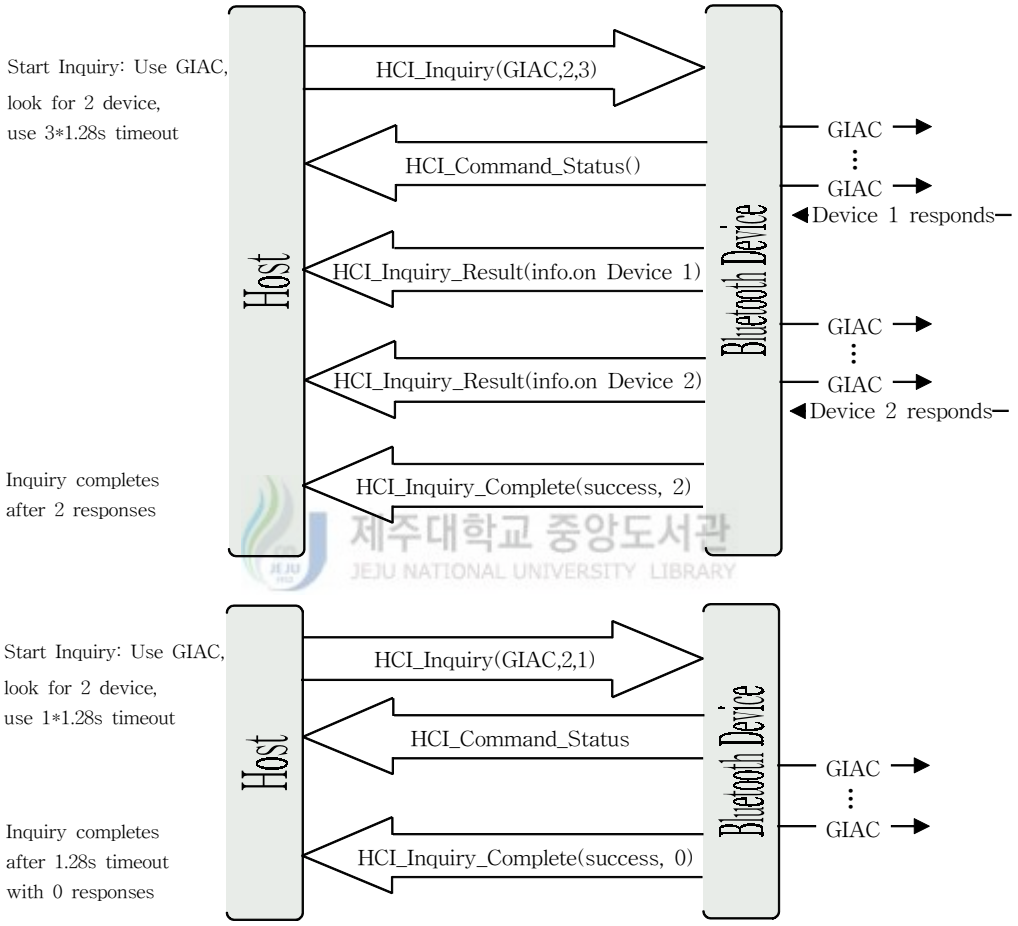


Fig.21 Message sequence charts for inquiries completing

Fig.21은 조회가 성공적으로 마칠 수 있는 두 가지 방법을 나타낸 것이다. 이 그림의 윗 부분의 메시지 순서도에서, 조회는 GIAC(General Inquiry Access Code)를 사용하여 시작하고, 최대로 두 개의 장치를 찾아낸다. 블루투스 장치는 GIAC를 포함하는 ID 패킷 스트림을 전송한다. 두 개의 장치가 응답하며, 각 응답은

HCI_Inquiry_Result를 발생하고, 두 번째 응답 후에, HCI_Inquiry_Complete 이벤트가 발생된다. 아래 그림의 메시지 순서도에서는 아무런 응답이 회송되지 않으므로, 조회가 끝나고, HCI_Inquiry_Complete 이벤트는 아무런 응답이 수신되지 않았음을 나타낸다.

data : 01 01 00
process_event_packet: **INQUIRY_COMPLETE**

마스터가 슬레이브 모듈을 위의 과정을 통해 슬레이브의 BD_ADDR을 알게되면, 두 모듈이 연결되기 위하여 마스터와 슬레이브 모두 아래의 동작을 수행하여 연결을 완성한다.

send_cmd : 0x1c 0xc 0x4 0x40 0x0 0x20 0x0
data : 0f 04 00 01 00 00
command status
length = 4, result = 00, num_hci_cmd = 1, ogf = 00, ocf = 00
data : 0e 04 01 1c 0c 00
command complete
length = 4, result = 0, num_hci_cmd = 1, ogf = 03, ocf = 1c
process_return_param: **WRITE_PAGESCAN_ACTIVITY**

블루투스 장치는 호출 스캔 모드로 들어감으로써 다른 장치들이 그들과 접속되도록 한다. 호출 스캐닝 상태에 있는 장치는 호출 장치로부터 보내지는 패킷 안에 있는 그 자신의 ID를 청취한다. 만약 그 자신의 ID를 발견하면 응답하고 두 장치는 새로운 접속을 설정한다.

send_cmd : 0x5 0x4 0xd 0xdd 0x17 0x0 0x5b 0x2 0x0 0x18 0xcc 0x0 0x0 0x0 0x0 0x0 0x0
seqcnt = 7

data : 0f 04 00 01 05 04

command status

length = 4, result = 00, num_hci_cmd = 1, ogf = 01, ocf = 05

data : 03 0b 00 28 00 dd 17 00 5b 02 00 01 00

process_event_packet: CONNECTION_COMPLETE:

lp_connect_cfm: Success! (hci_handle : 40)

만약, 위의 호출 스캔이 성공하면, 결국 ID 패킷이 그 안에 호출 스캐닝 장치의 블루투스 장치 주소로 수신된다. 그래서 호스트가 접속을 수락하면, 베이스밴드와 LMP 계층은 접속 설정을 완료한다. 일단 접속이 되면, 양쪽 호스트의 모듈들은 양쪽 호스트에게 HCI_Connection_Complete 이벤트를 보냈음을 통보하며, 이때 이 이벤트에는 접속을 구별하는데 사용되는 접속핸들(Connection_Handle)이 포함되어 있다. 그리고 접속이 종료시에는 HCI_Connection_Complete 이벤트에서 호스트로 전달된 접속핸들을 사용하여 접속 해제할 링크를 지정한다.

Ⅴ. 베이스밴드 구현에서의 개선 및 시뮬레이션

본 장에서는 Ⅲ장에서의 베이스밴드 대역을 직접 설계하여 패킷의 정확한 전송 여부와 오류검출과 암호화기, 홉 선택 시스템을 분석하여, 실제 ASIC으로 구현한 결과를 보여주고 있다. 설계 프로그램은 Max-plus, 칩은 FLEX10K를 이용하여 시뮬레이션 하였다.

특히, 베이스밴드 설계시 인증 구현 과정에 사용되는 SAFER+ 알고리즘의 개선 및 구현 방안과 주파수 홉 선택시의 시간지연 개선 방법을 제시하겠다.

1. 접근코드 생성 및 시뮬레이션

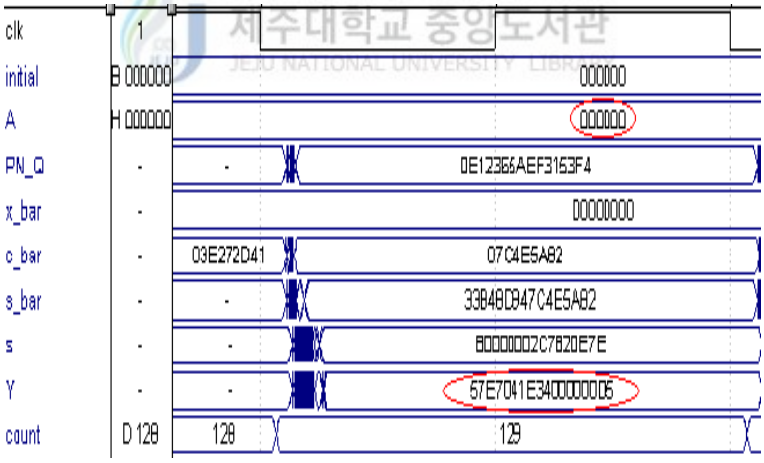


Fig.22 Completion of channel access code

채널접근코드는 LAP에 따라 결정되는데, Fig.22에서 보면 LAP가 000000일 때 채널접근코드가 57E7041E34000000D5임을 보이고 있다. 여기서 5는 Preamble, 7E7041E34000000D는 sync word, 마지막으로 5는 trailer를 나타낸다. 이는 Ⅲ장의

1-2)의 알고리즘으로 구현한 값이 Fig.23의 Bluetooth 1.0b의 데이터와 일치함을 확인할 수 있다.

LAP:	Preamble:	Sync word:	Trailer:
000000	5	7e7041e3 4000000d	5
ffffff	a	e758b522 7fffffff2	a
9e8b33	5	475c58cc 73345e72	a
9e8b34	5	28ed3c34 cb345e72	a
9e8b36	5	62337b64 1b345e72	a

Fig.23 Access Code Sample Data

2. 오류검출 시뮬레이션

베이스밴드에서는 앞서 구현한 채널접근코드를 비롯하여 헤더의 HEC, CRC를 단계별로 검사하여 데이터의 에러와 잘못된 주소를 체크하여 신뢰성을 높일 수 있다. 이를 확인하기 위하여 헤더의 HEC와 CRC에 오류 유무에 따른 결과를 시뮬레이션화하였다.

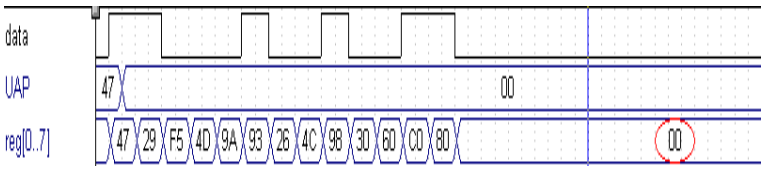


Fig.24 Receive a correct data in HEC

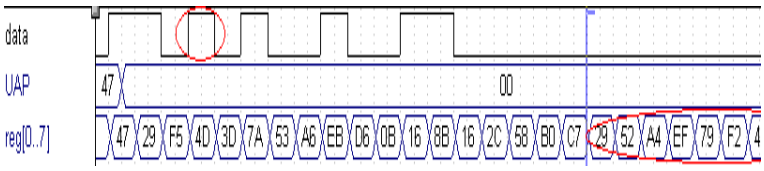


Fig.25 Receive a error data in HEC

Fig.24와 Fig.25는 헤더에서의 에러를 체크한 시뮬레이션 결과를 나타낸 그림이다.

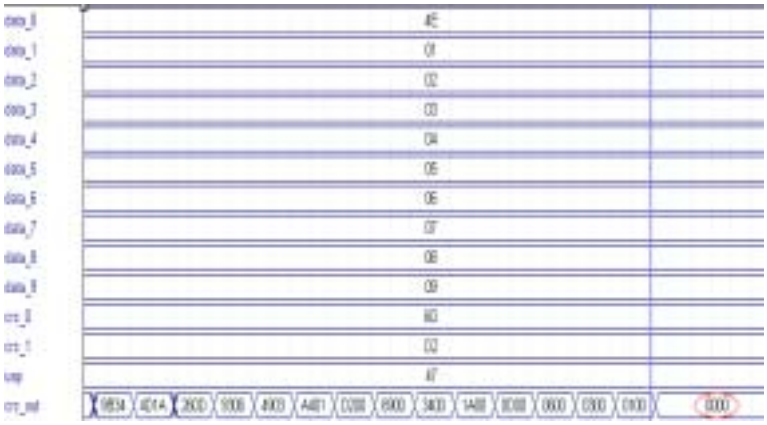


Fig.26 Receive a correct data in CRC

Fig.27 Receive a error data in CRC

Fig.26과 Fig.27은 CRC에서 데이터의 에러를 체크한 시뮬레이션한 결과이다. 이처럼 HEC와 CRC는 데이터의 에러와 잘못된 주소를 체크하기 위해 사용된다.

정상 데이터를 수신 후 에러체크를 했을 때는 0인 값, Fig.24와 Fig.26에서처럼 정상 데이터와 비교하여 오류인 값이 수신되었을 경우에는 0이 아닌 값으로 표시되어 그 결과값을 기준으로 데이터의 오류여부를 판단할 수 있다.

3. 인증을 위한 SAFER+ 알고리즘의 개선 방안

블루투스는 일반적으로 노트북 PC와 이동전화, PDA 같은 휴대장비에 많이 장착되기 때문에, 회로의 축소는 배터리 용량 한계 때문에 소비 전력을 아껴 써야 하는 휴대기기에 있어서 확실한 장점이라고 할 수 있다.

이처럼 블루투스에서 내세우고 있는 저전력의 장점을 강화하기 위해서는 SAFER+ 알고리즘을 좀 더 효율적으로 구현하는 것이 필요하다. SAFER+는 E1에서 뿐만 아니라 인증키를 생성하는데도 Ar′의 변형 형태로 쓰이기 때문에 이를 효과적으로 구현한다면 전력 소비를 최소화 할 수가 있다.

따라서 본 논문에서는 SAFER+ 회로의 축소를 위한 새로운 방안을 제시하겠다.

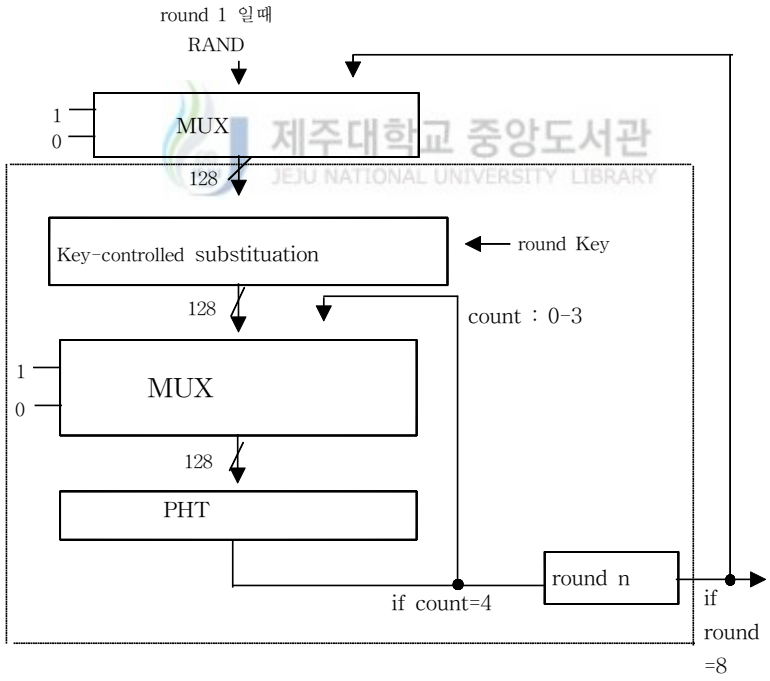


Fig.28 Improvement of SAFER+ Algorithm

이를 위해서 기존의 SAFER+ 알고리즘을 Fig.28과 같이 구현하였는데, 즉 각각

두 개의 MUX와 카운터를 사용함으로써 회로의 크기를 줄일 수 있었다. 이는 결과적으로 PHT단 24개가 줄어드는 효과를 볼 수 있다. 기존의 회로에 비해 추가 회로인 MUX가 들어가기는 하지만 합산기가 사용되는 PHT단의 감소는 전체적인 회로를 크게 줄여주는 역할을 하게 된다.

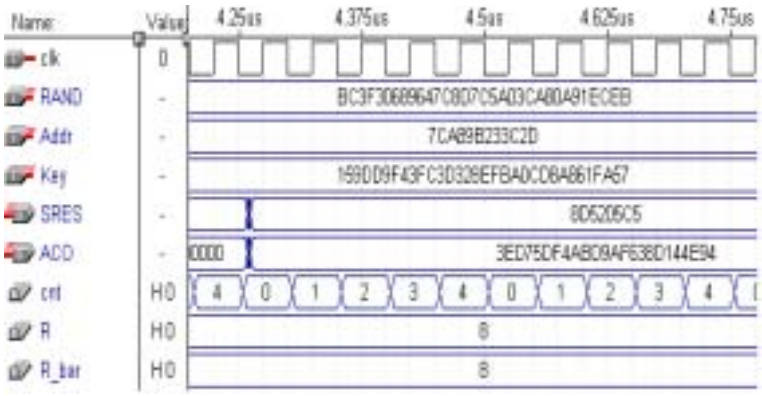


Fig.29 Result of Improvement SAFER+ Algorithm

Fig.29는 개선된 SAFER+을 이용하여 구현한 결과인데, 이는 기존의 SAFER+ 알고리즘으로 생성한 샘플 데이터의 결과와 일치함을 확인할 수 있다.

RAND가 'BC3F30689647C8D7C5A03CA80A91ECEB', 어드레스가 '7CA89B233C-2D' 그리고 키가 '159DD9F43FC3D328EFBA0CD8A861FA57'인 경우를 보여준다.

최종적으로 이 알고리즘을 인증을 위한 함수 E1을 설계시 적용하여 실제로 구현해 보면 Fig.30과 같은 결과를 얻을 수 있다.

Fig.30 Result of E1 apply to Improvement SAFER+ Algorithm

Fig.30을 살펴보면, RAND가 '00000000000000000000000000000000', 어드레스가

‘000000000000’ 그리고 키가 ‘00000000000000000000000000000000’인 경우 SRES가 ‘056C0FE6’, ACO는 ‘48AFCDD4BD40FE F76693B113’임을 보여주고 있다. 이를 블루투스 1.0b의 샘플 데이터와 비교해 보면 구현이 정확함을 알 수 있다.

결과적으로, 두 디바이스는 E1 알고리즘의 구현 결과로서 얻어진 SRES의 비교를 통해 인증의 수행 여부를 결정하게 된다.

4. 암호화 시뮬레이션

Fig.31은 보안성 보장을 위한 암호화기를 설계시 Fig.12에서의 각각의 LFSR에 클럭이 25, 31, 33, 39일 때까지 스위치를 off시키고, 그 후 스위치를 on시켜 원하는 z열을 얻는다.

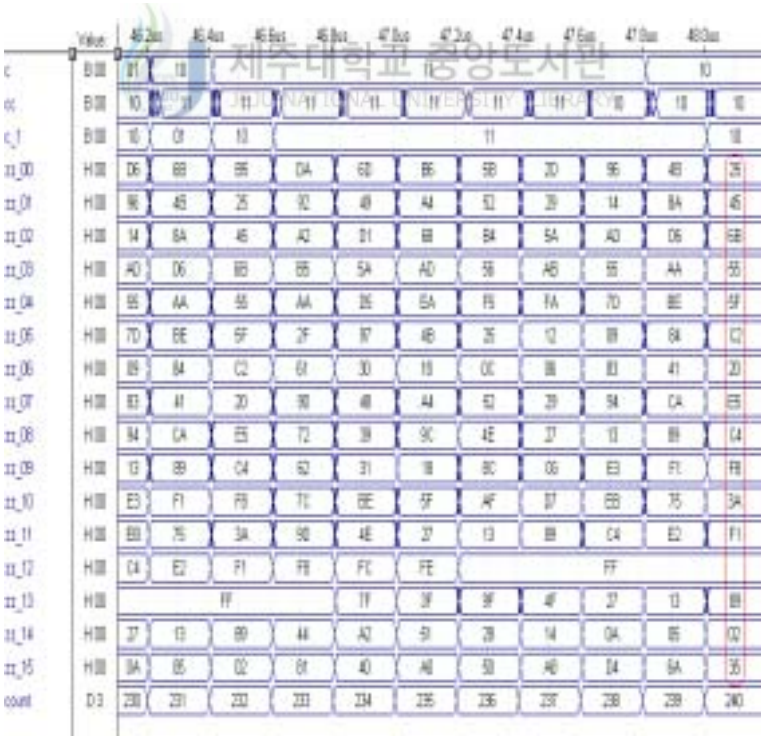


Fig.31 Result of the second part generates the key stream bits

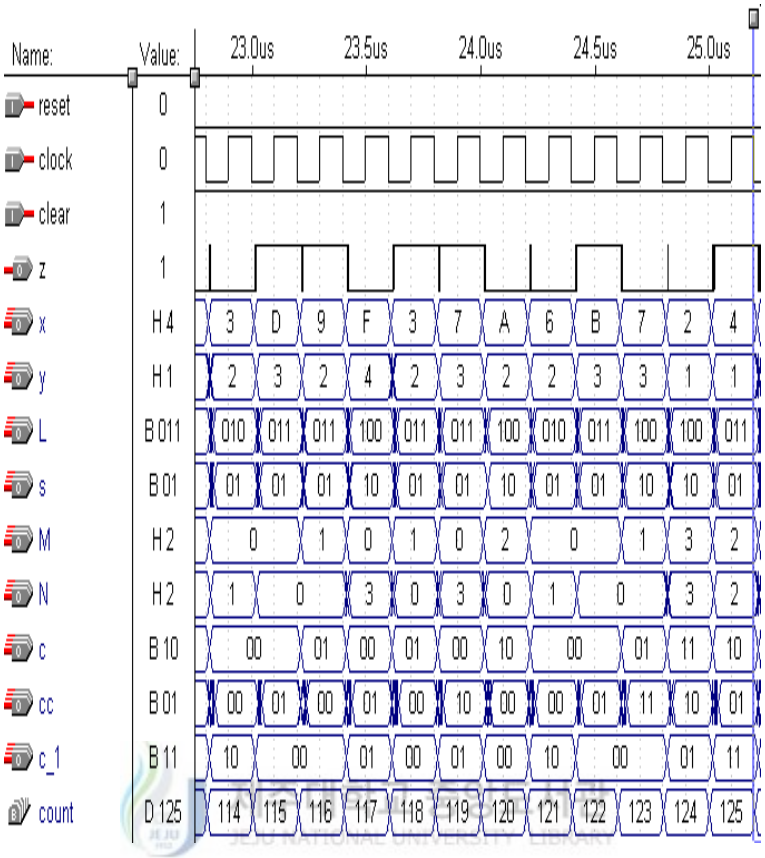


Fig.32 Result of encryption engine

이 z열을 8bit씩 묶어 다시 LFSR에 병렬로 실어 동작시켜, 최종적인 z열을 얻을 수 있는데 이것을 한 비트씩 전송될 데이터와 중첩함으로써 암호화에 사용된다. Fig.32는 이 과정을 시뮬레이션한 결과이다.

5. 주파수 홉 선택시 지연시간 개선 기법

Fig.13에서의 mod부분을 설계시 제산기를 사용하지 않는 방법을 제안하여 기존의 지연을 제거하였다. mod32부분은 최상위비트를 제거하여 나머지값을 얻을 수

있지만, mod79부분은 mod32와 달리 2^k로 정확히 나눌 수 없는 단점이 있다.

그래서, 출력값이 최대 9자리까지 나올 수 있기 때문에 프로그램시 79의 n배후의 값과 출력값을 비교하여 그에 대한 차를 나머지 값으로 하여 지연을 개선할 수 있다.

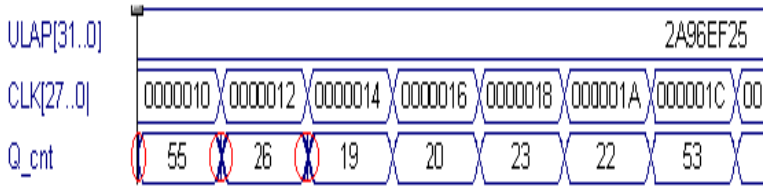


Fig.33 Connection state by divider

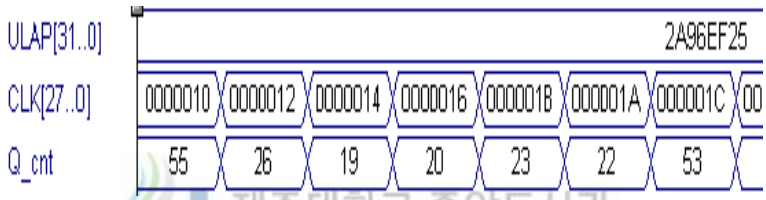


Fig.34 Connection state without divider

Fig.33과 Fig.34는 Table.1에서의 79-hop 시스템의 연결상태시의 주파수가 결정되는 결과를 보여주고 있다. 그리고, 제산기를 사용하는 것보다 사용하지 않는 쪽이 시작초부터 지연이 개선되어, 상당시간 경과 후에는 더욱 더 지연이 개선된다.

6. 주파수 홉 선택기 ASIC 칩 구현

블루투스의 79 호핑 시스템에서의 홉 선택의 수행을 위해 블루투스 기기의 고유주소인 ULAP가 0x9687cba9로 할당되었다는 가정하에 FLEX10K계열 EPF10K50RC240-3칩으로 제작하여 홉 선택 시 정상적인 동작을 확인한다.

1) 칩 테스트 보드와 I/O 핀 설정

Fig.35는 주파수 홉 선택기 ASIC 칩 동작 여부를 테스트하기 위한 보드이다.

Fig.35 HBE-DTK-240 test board

▶ DEVICE = EPF10K50RC240-3

▶ INPUT PIN

→ SIGNAL INPUT : cp = 91

→ KEYPAD INPUT

Sel[2] = 79	Sel[1] = 80	Sel[0] = 81
CLK = 82	Reset = 83	start = 87

➡ Sel = 000 : Page/Inquiry scan

001 : Page/Inquiry state

010 : Page slave

011 : Page master

100 : connection state

▶ OUTPUT PIN

→ LED OUTPUT

Q6 = 116	Q5 = 117	Q4 = 118	Q3 = 119
Q2 = 120	Q1 = 126	Q0 = 127	

2) 시뮬레이션과 테스트 결과

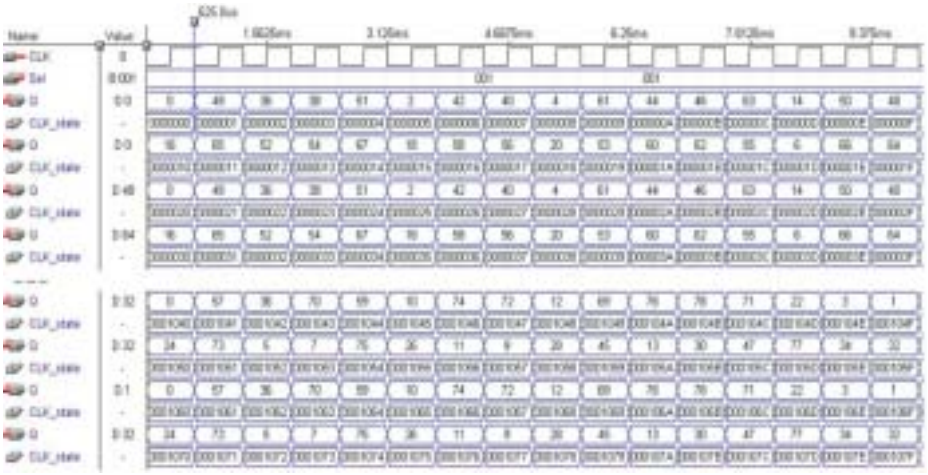


Fig.36 Hop sequence in page or inquiry

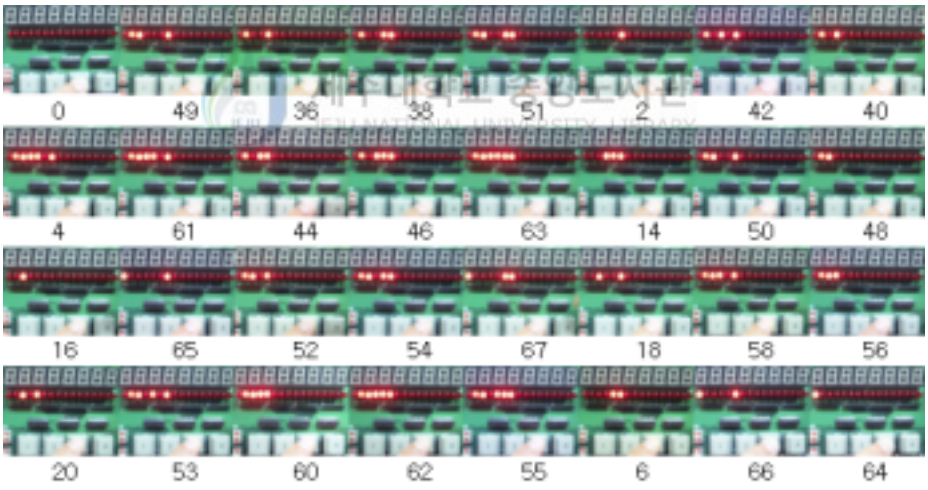


Fig.37 Chip operation of hop sequence in page or inquiry

Fig.36에 보여진 것처럼 CLK_ $state_{12}$ 가 짝수 비트이면 A-train이 적용되어 옵셋 값은 24이고 홀수이면 B-train이 적용되어 옵셋 값이 8이 되도록 설계하였다. Fig.37은 Fig.36의 호출 또는 조회 상태시의 시뮬레이션을 직접 칩으로 구현하여 그 동작이 정상적으로 수행됨을 보여주고 있다.

Ⅵ. 블루투스 시스템 구성도 및 최적 패킷 전송

본 장에서는 Ⅳ장에서 언급한 HCI에서의 패킷 흐름을 파악하기 위하여 구현한 RF단과 RS232C로 상호 연결한 시스템으로 실내 환경에서 발생 가능한 상황에 따라 파일전송을 수행함으로써 전송시간과 전송 속도를 통해 성능을 분석한다.

1. 모의 실험 시스템 구성

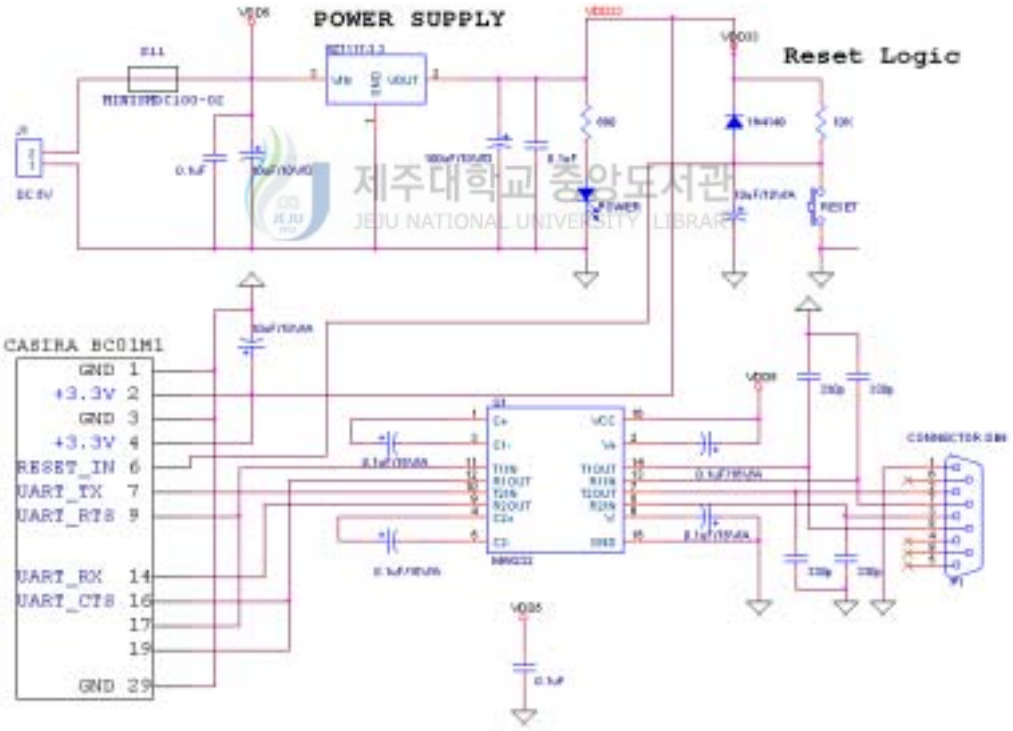


Fig.38 Bluetooth module and RS232C connection circuit

시리얼 통신의 방식에도 여러 가지가 있지만 본 논문에서는 RS232C 방식을 사

용하였는데, 모의실험에서 사용한 모듈의 입출력 신호는 3.3V, 0V로 RS232C 규격에 맞지 않다. 따라서 RS232C 통신을 하기 위해서는 외부에서 신호의 레벨을 변화 시켜 주어야하므로, 시리얼 포트로 값을 읽고 쓸 경우 3.3V를 마크로, 0V를 스페이스로 바꾸어 주었다. 이 과정은 MAX232 칩을 사용하였다.

Fig.38에서, 전원부분에서는 어댑터를 통해 5V 전원인가하면 EZ1117을 거치면서 3.3V 출력전원을 모듈의 2, 4번 핀에 연결한다. 그리고, 모듈과 RS232C 사이에 MAX232칩을 두고 다음과 같이 연결한다.

◀ RS232C의 RXD(2번 핀)과 모듈의 UART_TX(7번 핀)
◀ RS232C의 TXD(3번 핀)과 모듈의 UART_TX(14번 핀)
◀ RS232C의 RTS(7번 핀)과 모듈의 UART_TX(16번 핀)
◀ RS232C의 CTS(8번 핀)과 모듈의 UART_TX(9번 핀)

2. RS232C 연결 모듈과 EVM 보드

Fig.39와 Fig.40은 앞서 설명한 Fig.38의 시스템 구성도를 실지적으로 연결한 모습과 HCI에서의 최적 패킷을 전송하기 위하여 실험환경을 제공한 EVM 보드의 형태를 보여주고 있다.

Fig.39 Connection using RS232C

Fig.40 EVM board

3. 모듈 연결 후 거리별에 따른 파일 전송

데이터의 크기와 환경에 따른 전송 특성을 얻기 위해 데이터 크기가 30,335 Byte, 2,083,272 Byte를 전송한 후 거리에 따른 결과인 전송시간과 전송속도의 결과를 Table.3과 Table.4에 나타내었다.

Table.3 Data size (30,335 Byte)

거 리	2	4	6	8	10
시 간	7	7	6	7	7
전 송 속 도	34,664	34,664	40,440	34,664	34,664

[단위 : 거리(*m*), 시간(sec), 전송속도(*bps*)]

Table.4 Data size (2,083,272 Byte)

거 리	2	4	6	8	10
시 간	460	462	460	462	432
전 송 속 도	36,224	36,072	36,224	36,072	36,072

[단위 : 거리(*m*), 시간(sec), 전송속도(*bps*)]

모의 실험을 위해 모듈은 최대 4 kbps 정도의 전송속도로 제작되었고, Table.3과 Table.4에 나타나있듯이 데이터 크기에 관계없이 거의 일정한 전송속도를 보임을 확인할 수 있다. 이때, 데이터를 전송할 때 거리는 직선 가시거리로 설정하였다.

그리고, 블루투스 피코넷 망의 범위가 10m 내외이므로, 이 범위 내에서 여러 장애물이 있는 환경을 설정하여 파일 전송을 한 결과, Table.3과 Table.4와 같은 결과를 얻어서 이 범위 내에서는 장애물이 파일 전송속도에 영향을 끼치지 않음을 알 수 있다.

Ⅶ. 결 론

본 논문에서는 블루투스에서의 베이스밴드 대역을 블루투스 1.0b를 바탕으로 제시한 알고리즘을 이용하여 설계하였다. 알고리즘 구조는 VHDL로 표현·기술한 후 시뮬레이션을 수행하여, 패킷의 정확한 전송여부와 오류검출과 암호화기, 홉 선택 시스템을 분석하여 그 결과를 비교 검토함으로써 그 효용성을 입증하였으며 이를 ASIC 칩으로 구현하였다.

베이스밴드 설계 과정 중에, 특히 인증부분에서 SAFER+ 알고리즘 회로를 구현시 기존의 방법과 달리 MUX와 카운터를 사용함으로써 단의 수를 간략히 하여 전체 회로 크기를 줄였으며, 이 결과 블루투스에서 내세우고 있는 저전력의 장점을 한층 강화할 수 있을 것으로 기대된다. 그리고, 주파수 홉 선택 시스템을 설계시 기존의 방법인 두 군데 mod 부분을 제산기를 사용하여 설계하는 대신에, 첫 번째 mod 부분인 mod32는 최상위비트를 제거하여 나머지 값을 얻는 방법을 사용하였다. 두 번째 mod 부분인 mod79는 mod32와 달리 2^n로 정확히 나눌 수 없기 때문에 출력값이 최대 9자리까지 나올 수 있는 점을 이용하여 프로그램시 79의 n배후의 값과 출력값을 비교하여 그에 대한 차를 나머지 값으로 얻는 방법을 사용함으로써 지연 개선 효과를 얻을 수 있었다. 이렇게 제시한 방법으로 설계한 주파수 홉 선택기를 직접 칩으로 구현하여 그 동작이 정상적으로 수행됨을 확인하였다.

실제적인 블루투스 모듈간 연결과 전송을 위하여 두 모듈을 호스트와 RS232C로 연결하여 모의 실험 환경을 설정한 후, 두 모듈간을 연결하기 위한 동작은 각 커맨드에 대한 HCI 이벤트 패킷 발생 과정을 통해 확인하였다. 또한, 이 과정에서 설계한 모듈을 이용하여 데이터 전송 모의 실험에 의해 데이터 크기에 따라 거리별 차등을 주어 전송을 수행한 결과인 전송시간과 전송 속도를 비교해봄으로써 시스템 성능을 분석하였다.

참 고 문 헌

- Jennifer Bray and Charles F Sturmam, 2001, "BLUETOOTH Connect without Cables", Prentice-Hall, pp.41-65.
- Nathan J. MULLER, 2000, "BLUETOOTH DEMYSTIFIED", McGraw-Hill Companies, 396pp.
- Miller and Brent A, 2000, "Bluetooth Revealed", Prentice-Hall, 303pp.
- Held and Gilbert, 2000, "Data over Wireless Networks", McGraw-Hill. 230pp.
- Bluetooth Special Interest Group, 1999 , "Specification of the bluetooth system Part B", pp.47-66
- 동역메카트로닉스연구소 기술정보실, 2000, “스팩트럼 확산통신 시스템 설계”, 국제테크노정보연구소, 17-51pp.
- Markus Jakobsson and Susanne Wetzel, 2001, "Security Weakness in Bluetooth", Progress in Cryptotogy CT-RSA
- D. L. Schilling, 1990, "Spread Spectrum goes Commercial", IEEE Spectrum
- J. Zander and G. Malmgren, 1995, "Adaptive frequency hopping in HF communications", IEEE Proc.-Commun, pp.99-105
- J. C. Haartsen, 2000, "The Bluetooth Radio System", IEEE Personal Communications, pp.28-36
- Kris Fleming., 2000, "Architectural Overview of Intel's Bluetooth Software Stack", Intel Technology Journal
- Thomas Barker, 2000, "Bluetooth Security", Bluetooth Developers Conference
- Graham Kirby, 2000, "Integrating Bluetooth Technology into Mobile Products", Intel Technology Journal
- Steve Karty, 2000, "Bluetooth Personal Area Network Technology", NCS Tech. notes