
Bluetooth

Part 2: Baseband and Java ME

Kjell Jørgen Hole

UiB



Last updated 22.02.09

Mail: Kjell.Hole@ii.uib.no

URL: www.kjhole.com

Outline

KJhole.com

- Definition of baseband
- System timing
- Logical transport: ACL, SCO and eSCO
- Packet structure
- Channel coding
- Overview of Java ME (Java Platform, Micro Edition)

Baseband Part of a device which controls the radio (see Figure 2-1). It is responsible for low level timing, error control, and management of the link within the domain of a single data packet transfer

2.3

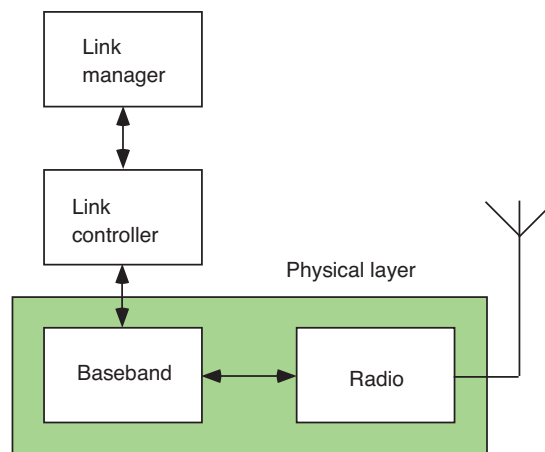


Figure 2-1 Link control and the baseband

2.4

CLKN Real time clock. Implemented by 28 bit counter, reset to 0 at power up. Incremented every half slot, or $312.5 \mu\text{s}$ ($\mu = 10^{-6}$)

All Bluetooth devices use CLKN to:

- synchronize Tx-Rx data exchanges between devices
- differentiate between lost and re-sent packets
- generate predictable and reproducible sequence of hop channel numbers

2.5

- Each Slave in a piconet adds an offset value onto its CLKN (see Figure 2-2). New value—denoted CLK and called **piconet clock**—is an estimate of Master's CLKN
- Master adds another offset to its CLKN to obtain an estimate, CLKE, of the CLK in a Slave device. CLKE is used to connect to Slave before Slave has synchronized to Master

2.6

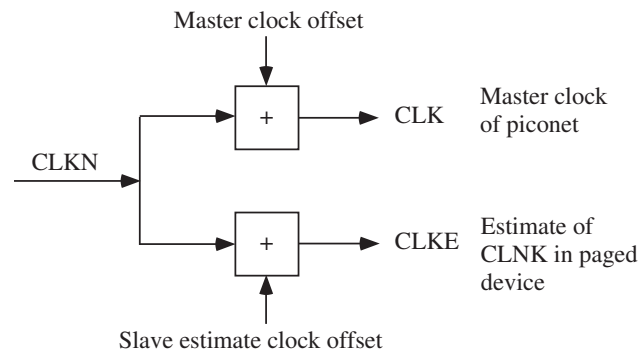


Figure 2-2 Conceptual Bluetooth CLK offset application

2.7

ACL Logical Transport

KJhole.com

ACL (*Asynchronous Connection-Oriented*) logical transports provide packet-switched connections:

- A Master may have a number of ACL logical transports to different Slaves
- Only *one* ACL transport can exist between a Master and a Slave
- A Master need not transmit to a Slave in a regular fashion
- Master determines which Slave to transmit to and receive from on a slot by slot basis

2.8

More on ACL Logical Transports

- Most ACL packets facilitate error checking and re-transmission
- ACL logical transports carry data to and from L2CAP and Link Manager (LM) layers
- Data carried in **DH** (Data High rate) packets and **DM** (Data Medium rate) packets. DM packets carry less data, but provide extra error protection
- Broadcast packets are ACL packets that are not addressed to a specific Slave

2.9

SCO Logical Transports

KJhole.com

SCO (*Synchronous Connection Oriented*) logical transports provide circuit-switched connections that carry 64 kbps of information:

- Symmetric connections between Master and Slave with reserved bandwidth in the form of reserved slots
- Intended for use with time-bounded information such as audio
- Master can support up to three SCO transports to the same Slave or to different Slaves

2.10

More on SCO Logical Transports

- SCO packets are not retransmitted
- A SCO transport is set up by a LM command from the Master to the Slave. Message contains timing parameters to specify the reserved slots
- *Extended SCO (eSCO)* logical transports:
 - support a range of synchronous bit rates
 - offer limited retransmission of packets

2.11

Packet Structure

KJhole.com

A standard basic rate packet may consist of (see Figure 2-3):

Access code Used to detect the presence of a packet. Identifies the packet as being from or to a specific Master. Always included

Header Contains all control information associated with the packet and link, such as address to intended Slave. Not used in some packets

Payload User data and control information from higher layers. Not always included

2.12

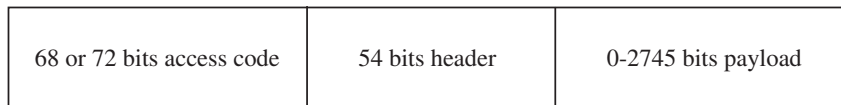


Figure 2-3 Bluetooth packet structure

2.13

Access Code

KJhole.com

Access code consists of (see Figure 2-4):

- 4 bits preamble used to detect edges of received data
- 64 bits synchronization word
- 4 bits trailer, only used when access code is followed by a packet header

2.14

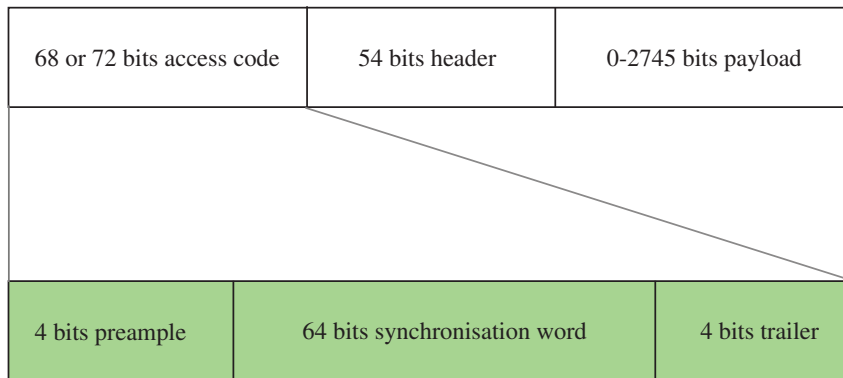


Figure 2-4 Access code structure

2.15

Synchronization Word Algorithm

KJhole.com

1. Determine 24-bit Lower Address Part (LAP) of Bluetooth device address (48 bit IEEE MAC address)
 - a device address or reserved inquiry address is used
2. Append 6-bit Barker sequence to LAP to improve auto-correlation properties
3. XOR new sequence with bits 34 to 63 of full length, 64-bit Pseudorandom Noise (PN) sequence
4. Encode resulting 30-bit sequence with (64,30) BCH (Bose-Chaudhuri-Hocquenghem) block code to obtain 34 parity bits

2.16

Algorithm Continued

KJhole.com

5. 34-bit parity word XOR'd with the remaining bits, 0 to 33 of PN sequence to remove cyclic properties of block code

Remark: 34 bits BCH parity word exhibits very high auto-correlation and very low co-correlation properties. Thus, a correlator can be used to obtain a match between the received and expected (reference) synch word. The BCH code also ensures large Hamming distance (≥ 14) between sync words based on different LAPs

2.17

Types of Access Codes (1)

KJhole.com

Channel Access Code (CAC)—derived from Master's LAP and is used by all devices in apiconet during data exchange

Device Access Code (DAC)—derived from a specific device's LAP. It is used when paging a specific device and by that device in Page Scan while listening for paging messages to itself

General Inquiry Access Code (GIAC)—used by all devices during the inquiry procedures

2.18

Types of Access Codes (2)

KJhole.com

Dedicated Inquiry Access Code (DIAC)—range of reserved codes for inquiry procedures between specific devices

2.19

Packet Header

KJhole.com

- 54 bits packet header (see Figure 2-5) contains 18 bits of information encoded with a rate 1/3 repetition code, i.e., each information bit is transmitted 3 times
- The large amount of overhead is included because each header field is crucial to the correct operation of the link

2.20

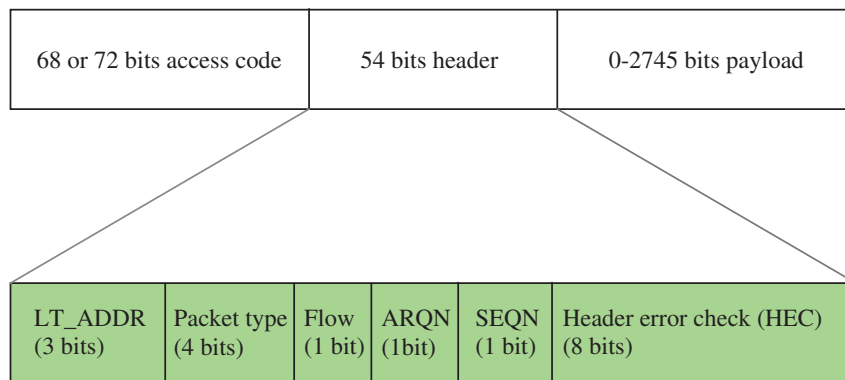


Figure 2-5 Packet header structure

2.21

Header Fields

KJhole.com

LT_ADDR Master assigns Logical Transport ADDRESS (LT_ADDR) to Slave. 3-bit field sufficient for 7 Slaves. Broadcast packet has address zero

- Master does not have an LT_ADDR

Flow Flag asserted when device is unable to receive any more data due to full receiver buffer

ARQN and SEQN SEQN toggled each time new packet with CRC is transmitted, ACK represented by ARQN=1 and NAK by ARQN=0

Header Error Check (HEC) 8-bit CRC

2.22

Packet type Defines which type of traffic is carried by packet:

- SCO, ACL
- **ID** packet consists of access code, used during “pre-connection”
- **NULL** packet consists of access code and header, used for flow control or to pass ARQ
- **POLL** packet same structure as NULL packet, must be acknowledged. Not part of the ARQ scheme
- FHS (Frequency Hop Synchronization) contains device address and clock of sender

2.23

ACL Payload

The ACL payload is divided into three fields (see Figure 2-6):

- payload header with fields:
 - LLID (Logical Link ID)** Field indicates whether payload is start or continuation of an L2CAP message or an LMP message
 - Flow** Controls data flow at L2CAP level
 - Length** Number of data bytes in payload
- payload data
- CRC, calculated over both payload header and payload

2.24

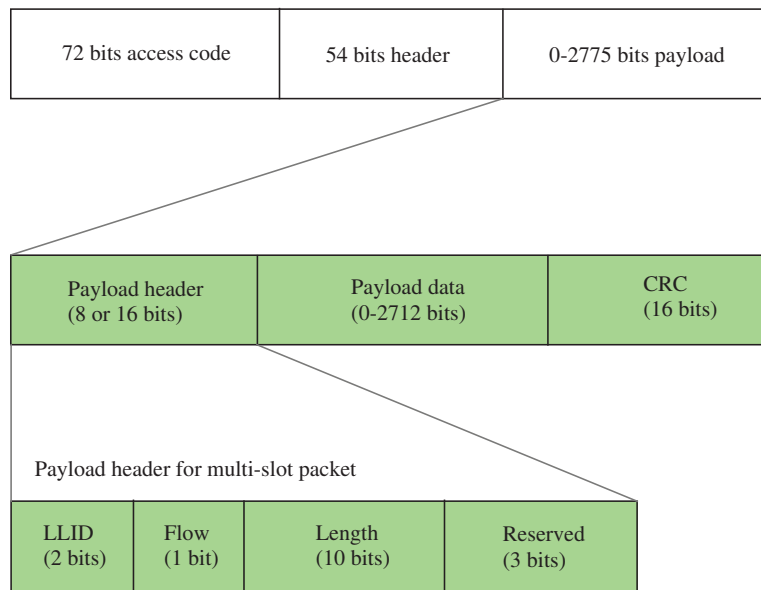


Figure 2-6 ACL payload and payload header structure

2.25

SCO Packet Structure

KJhole.com

- Same access code and header as ACL packets
- ARQ and SEQ fields in header redundant since flow control and retransmission are not used
- CRC field in header not used
- Payload size fixed at 30 bytes (240 bits), error control code with rate 1/3, 2/3, or 1 (no code) used for source data size of 10, 20, or 30 bytes

2.26

- The payload length
 - is negotiated during LMP eSCO setup
 - remains constant unless re-negotiated
 - may be different for each direction

2.27

Bitstream Processing

CRC Performed on all packet headers and ACL payload data. Also applied to the payload data in some SCO packets

Encryption Chiper stream produced by encryption engine XOR'd into bitstream data path

Whitening or bit randomization Pseudo random bit sequence mixed with data bitstream. Reduces DC bias, thus avoiding channel drift

FEC (Forward Error Correction) Non, rate 1/3 repetition code, and rate 2/3 shortened (15,10) Hamming code

2.28

Java ME

2.29

Java ME Overview

KJhole.com

- Java ME is the Java platform for consumer devices such as mobile phones, PDAs, TV set-top boxes, and in-vehicle telematics systems, as well as for a broad range of embedded devices
- Like its enterprise (Java EE), desktop (Java SE) and smart card (Java Card) counterparts, Java ME is a set of standard Java APIs defined through the *Java Community Process* (JCP) program
- Java ME brings the power and benefits of Java technology to consumer and embedded devices (code portability, object-oriented programming, user interface, security model, and network protocols)

2.30

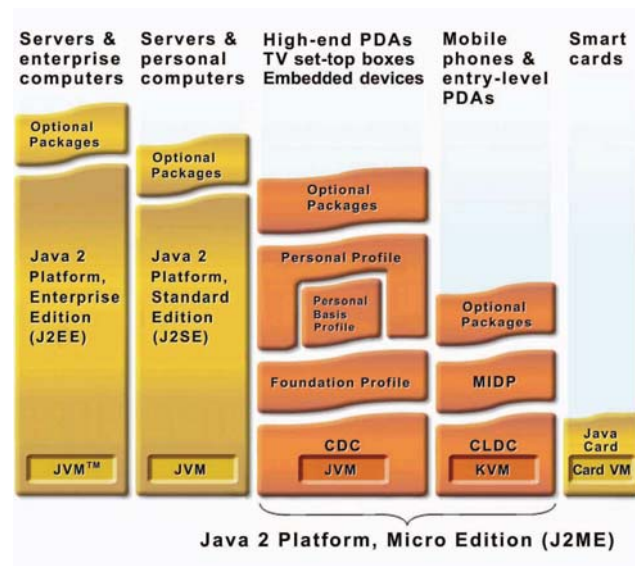


Figure 2-7 Java platforms

2.31

The Java ME Architecture

KJhole.com

- The Java ME architecture defines **configurations**, **profiles** and **optional packages** as elements for building complete Java runtime environments
 - each combination is optimized for the memory, processing power, and I/O capabilities of a related category of devices
- See <http://java.sun.com/javame/>

2.32

Configuration Composed of a virtual machine and a minimal set of class libraries. Configurations provide the base functionality for different types of devices

- Currently, there are two Java ME configurations: the *Connected, Limited Device Configuration* (**CLDC**), and the *Connected Device Configuration* (**CDC**)

2.33

- CLDC is the smaller of the two configurations, designed for devices with intermittent network connections, slow processors and limited memory (e.g., mobile phones and PDAs)
- These devices typically have either 16- or 32-bit CPUs, and a minimum of 192 KB memory available for the Java platform implementation and associated applications
- A CLDC implementation generally includes a *Kilobyte Virtual Machine* (KVM), specially designed for memory-constrained devices

2.34

- CDC is designed for high-end consumer devices that have more memory, faster processors, and greater network bandwidth, such as TV set-top boxes, residential gateways, in-vehicle telematics systems, and high-end PDAs
- CDC includes a full-featured Java virtual machine, and a much larger subset of the Java SE platform than CLDC
- Most CDC-targeted devices have 32-bit CPUs and a minimum of 2MB of memory available for the Java platform and associated applications

2.35

Profile In order to provide a complete runtime environment targeted at specific device categories, configurations must be combined with a set of higher level APIs, or profiles, that further define the application life cycle model, the user interface, and access to device specific properties

- It is possible for a single device to support several profiles
- Important profiles are: *Mobile Information Device Profile (MIDP)*, *Foundation Profile (FP)*, and *Personal Profile (PP)*

2.36

- MIDP is designed for mobile phones and entry-level PDAs
- MIDP combined with CLDC offers core application functionality, such as a user interface, network connectivity, and local data storage
- MIDP applications are called **MIDlets**. MIDlet is a class defined in MIDP and is the superclass for all MIDP applications

2.37

- CDC profiles are layered so that profiles can be added as needed to provide application functionality for different types of devices
- FP is the lowest level profile for CDC. It provides a network-capable implementation of CDC that can be used for embedded implementations without a user interface
- PP is the CDC profile aimed at devices that require full GUI or Internet applet support, such as high-end PDAs, communicator-type devices, and game consoles. Includes the full Java Abstract Window Toolkit (AWT) libraries

2.38

- The Java ME platform can be further extended by combining various optional packages with CLDC, CDC, and their corresponding profiles
- Created to address very specific market requirements, optional packages offer standard APIs for using both existing and emerging technologies such as *Bluetooth*, Web services, wireless messaging, multimedia, and database connectivity
- Because optional packages are modular, device manufacturers can include them as needed to fully leverage the features of each device

2.39

- The JSR-82 expert group focused on Java ME devices
- The Java Bluetooth API rely heavily on one set of CLDC APIs known as the *Generic Connection Framework* (GCF)
- GCF is included in Java SE under JSR-197 (Generic Connection Framework Optional Package),
see <http://www.jcp.org/en/jsr/detail?id=197>

2.40

- Baseband responsible for coding, timing, and link management within the domain of a single data packet transfer
- Devices exist in two modes of operation, namely Slave and Master
- SCO data links for time bounded data and ACL links for packet based data
- The Java ME platform makes it possible for Java programmers to develop applications (MIDlets) for handheld devices