8 TRANSMIT/RECEIVE ROUTINES

This section describes the way to use the packets as defined in Section 4 in order to support the traffic on the ACL and SCO links. Both single-slave and multi-slave configurations are considered. In addition, the use of buffers for the TX and RX routines are described.

The TX and RX routines described in sections 8.1 and 8.2 are of an informative character only. The final implementation may be carried out differently.

8.1 TX ROUTINE

The TX routine is carried out separately for each ACL link and each SCO link. Figure 8.1 on page 81 shows the ACL and SCO buffers as used in the TX routine. In this figure, only a single TX ACL buffer and a single TX SCO buffer are shown. In the master, there is a separate TX ACL buffer for each slave. In addition there may be one or more TX SCO buffers for each SCO slave (different SCO links may either reuse the same TX SCO buffer, or each have their own TX SCO buffer). Each TX buffer consists of two FIFO registers: one **current** register which can be accessed and read by the Bluetooth controller in order to compose the packets, and one **next** register that can be accessed by the Bluetooth Link Manager to load new information. The positions of the switches S1 and S2 determine which register is current and which register is next; the switches are controlled by the Bluetooth Link Controller. The switches at the input and the output of the FIFO registers can never be connected to the same register simultaneously.

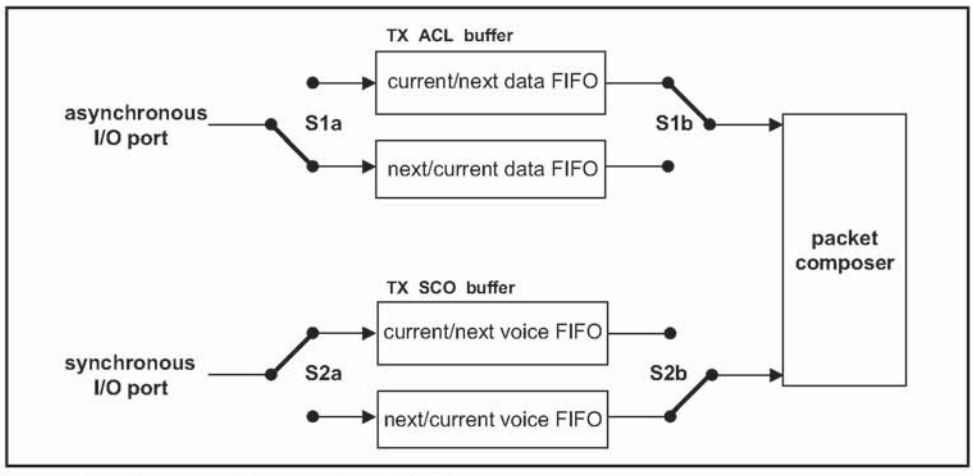

Figure 8.1: Functional diagram of TX buffering.

Of the packets common on the ACL and SCO links (**ID**, **NULL**, **POLL**, **FHS**, **DM1**) only the **DM1** packet carries a payload that is exchanged between the Link Controller and the Link Manager; this common packet makes use of the

ACL buffer. All ACL packets make use of the ACL buffer. All SCO packets make use of the SCO buffer except for the **DV** packet where the voice part is handled by the SCO buffer and the data part is handled by the ACL buffer. In the next sections, the operation for ACL traffic, SCO traffic, and combined data-voice traffic on the SCO link will be considered.

8.1.1 ACL traffic

In the case of pure (asynchronous) data, only the TX ACL buffer in Figure 8.1 on page 81 has to be considered. In this case, only packet types **DM** or **DH** are used, and these can have different lengths. The length is indicated in the payload header. The selection of high-rate data or medium-rate data shall depend on the quality of the link. When the quality is good, the FEC in the data payload can be omitted, resulting in a **DH** packet. Otherwise, **DM** packets must be used.

The default TYPE in pure data traffic is **NULL**. This means that, if there is no data to be sent (the data traffic is asynchronous, and therefore pauses occur in which no data is available) or no slaves need to be polled, **NULL** packets are sent instead – in order to send link control information to the other Bluetooth unit (e.g. ACK/STOP information for received data). When no link control information is available either (no need to acknowledge and/or no need to stop the RX flow) no packet is sent at all.

The TX routine works as follows. The Bluetooth Link Manager loads new data information in the register to which the switch S1a points. Next, it gives a **flush** command to the Bluetooth Link Controller, which forces the switch S1 to change (both S1a and S1b switch in synchrony). When the payload needs to be sent, the packet composer reads the current register and, depending on the packet TYPE, builds a payload which is appended to the channel access code and the header and is subsequently transmitted. In the response packet (which arrives in the following RX slot if it concerned a master transmission, or may be postponed until some later RX slot if it concerned a slave transmission), the result of the transmission is reported back. In case of an ACK, the switch S1 changes position; if a NAK (explicit or implicit) is received instead, the switch S1 will not change position. In that case, the same payload is retransmitted at the next TX occasion.

As long as the Link Manager keeps loading the registers with new information, the Bluetooth Link Controller will automatically transmit the payload; in addition, retransmissions are performed automatically in case of errors. The Link Controller will send **NULL** or nothing when no new data is loaded. If no new data has been loaded in the **next** register, during the last transmission, the packet composer will be pointing to an empty register after the last transmission has been acknowledged and the **next** register becomes the **current** register. If new data is loaded in the **next** register, a **flush** command is required to switch the S1 switch to the proper register. As long as the Link Manager keeps loading the data and type registers before each TX slot, the data is automatically processed by the Link Controller since the S1 switch is controlled by the

ACK information received in response. However, if the traffic from the Link Manager is interrupted once and a default packet is sent instead, a **flush** command is required to continue the flow in the Link Controller.

The **flush** command can also be used in case of time-bounded (isochronous) data. In case of a bad link, many retransmission are necessary. In certain applications, the data is time-bounded: if a payload is retransmitted all the time because of link errors, it may become outdated, and the system might decide to continue with more recent data instead and skip the payload that does not come through. This is accomplished by the **flush** command as well. With the **flush**, the switch S1 is forced to change and the Link Controller is forced to consider the next data payload and overrules the ACK control.

8.1.2 SCO traffic

In case of an SCO link, we only use **HV** packet types. The synchronous port continuously loads the **next** register in the SCO buffer. The S2 switches are changed according to the T_{sco} interval. This T_{sco} interval is negotiated between the master and the slave at the time the SCO link is established.

For each new SCO slot, the packet composer reads the **current** register after which the S2 switch is changed. If the SCO slot has to be used to send control information with high priority concerning a control packet between the master and the considered SCO slave, or a control packet between the master and any other slave, the packet composer will discard the SCO information and use the control information instead. This control information must be sent in a DM1 packet. Data or link control information can also be exchanged between the master and the SCO slave by using the **DV** or **DM1** packets. Any ACL type of packet can be used to sent data or link control information to any other ACL slave. This is discussed next.

8.1.3 Mixed data/voice traffic

In Section 4.4.2 on page 58, a **DV** packet has been defined that can support both data and voice simultaneously on a single SCO link. When the TYPE is **DV**, the Link Controller reads the data register to fill the data field and the voice register to fill the voice field. Thereafter, the switch S2 is changed. However, the position of S1 depends on the result of the transmission like on the ACL link: only if an ACK has been received will the S1 switch change its position. In each **DV** packet, the voice information is new, but the data information might be retransmitted if the previous transmission failed. If there is no data to be sent, the SCO link will automatically change from **DV** packet type to the current **HV** packet type used before the mixed data/voice transmission. Note that a **flush** command is required when the data stream has been interrupted and new data has arrived.

Combined data-voice transmission can also be accomplished by using separate ACL links in addition to the SCO link(s) if channel capacity permits this.

8.1.4 Default packet types

On the ACL links, the default type is always **NULL** both for the master and the slave. This means that if no user information needs to be send, either a **NULL** packet is sent if there is **ACK** or **STOP** information, or no packet is sent at all. The **NULL** packet can be used by the master to allocate the next slave-to-master slot to a certain slave (namely the one addressed). However, the slave is not forced to respond to the **NULL** packet from the master. If the master requires a response, it has to send a **POLL** packet.

The SCO packet type is negotiated at the LM level when the SCO link is established. The agreed packet type is also the default packet type for the SCO slots.

8.2 RX ROUTINE

The RX routine is carried out separately for the ACL link and the SCO link. However, in contrast to the master TX ACL buffer, a single RX buffer is shared among all slaves. For the SCO buffer, it depends how the different SCO links are distinguished whether extra SCO buffers are required or not. Figure 8.2 on page 84 shows the ACL and SCO buffers as used in the RX routine. The RX ACL buffer consists of two FIFO registers: one register that can be accessed and loaded by the Bluetooth Link Controller with the payload of the latest RX packet, and one register that can be accessed by the Bluetooth Link Manager to read the previous payload. The RX SCO buffer also consists of two FIFO registers: one register which is filled with newly arrived voice information, and one register which can be read by the voice processing unit.

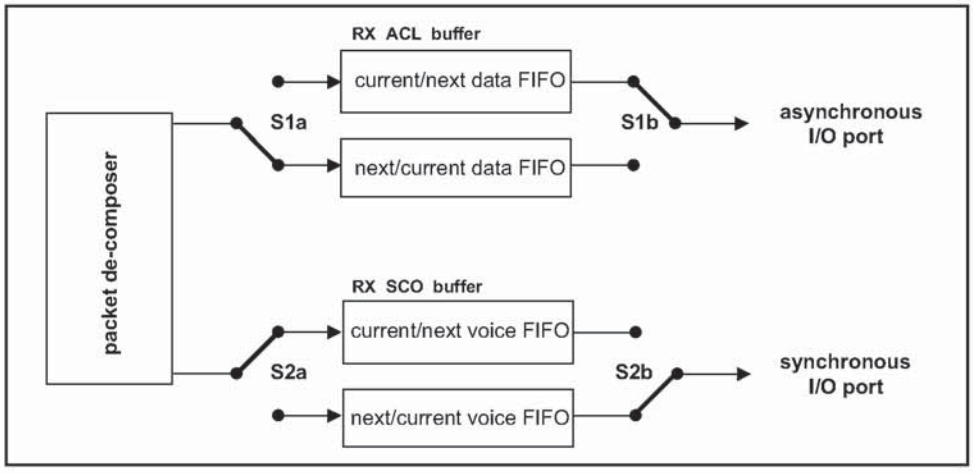

Figure 8.2: Functional diagram of RX buffering

Since the TYPE indication in the header of the received packet indicates whether the payload contains data and/or voice, the packet de-composer can automatically direct the traffic to the proper buffers. The switch S1 changes

every time the Link Manager has read the old register. If the next payload arrives before the RX register is emptied, a STOP indication must be included in the packet header of the next TX packet that is returned. The STOP indication is removed again as soon as the RX register is emptied. The SEQN field is checked before a new ACL payload is stored into the ACL register (flush indication in L_CH and broadcast messages influence the interpretation of the SEQN field see Section 5.3 on page 68).

The S2 switch is changed every T_{SCO}. If – due to errors in the header – no new voice payload arrives, the switch still changes. The voice processing unit then has to process the voice signal to account for the missing speech parts.

8.3 FLOW CONTROL

Since the RX ACL buffer can be full while a new payload arrives, flow control is required. As was mentioned earlier, the header field FLOW in the return TX packet can use STOP or GO in order to control the transmission of new data.

8.3.1 Destination control

As long as data cannot be received, a STOP indication is transmitted which is automatically inserted by the Link Controller into the header of the return packet. STOP is returned as long as the RX ACL buffer is not emptied by the Link Manager. When new data can be accepted again, the GO indication is returned. GO is the default value. Note that all packet types not including data can still be received. Voice communication for example is not affected by the flow control. Also note that although a Bluetooth unit cannot receive new information, it can still continue to transmit information: the flow control is separate for each direction.

8.3.2 Source control

On the reception of a STOP signal, the Link Controller will automatically switch to the default packet type. The current TX ACL buffer status is frozen. Default packets are sent as long as the STOP indication is received. When no packet is received, GO is assumed implicitly. Note that the default packets contain link control information (in the header) for the receive direction (which may still be open) and may contain voice (**HV** packets). When a GO indication is received, the Link Controller resumes to transmit the data as is present in the TX ACL buffers.

In a multi-slave configuration, only the transmission to the slave that issued the STOP signal is stalled. This means that the previously described routine implemented in the master only concerns the TX ACL buffer that corresponds to the slave that cannot accept data momentarily.

8.4 BITSTREAM PROCESSES

Before the user information is sent over the air interface, several bit manipulations are performed in the transmitter to increase reliability and security. To the packet header, an HEC is added, the header bits are scrambled with a whitening word, and FEC coding is applied. In the receiver, the inverse processes are carried out. Figure 8.3 on page 86 shows the processes carried out for the packet header both at the transmit and the receive side. All header bit processes are mandatory.

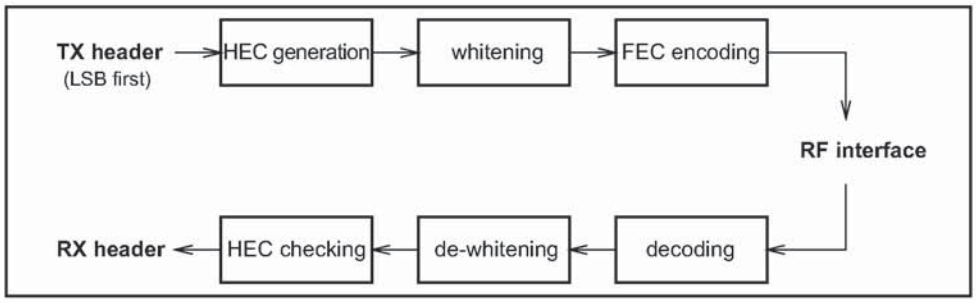

Figure 8.3: Header bit processes.

For the payload, similar processes are performed. It depends on the packet type, which processes are carried out. Figure 8.4 on page 86 shows the processes that may be carried out on the payload. In addition to the processes defined for the packet header, encryption can be applied on the payload. Only whitening and de-whitening, as explained in Section 7 on page 79, are mandatory for every payload; all other processes are optional and depend on the packet type and the mode enabled. In Figure 8.4 on page 86, optional processes are indicated by dashed blocks.

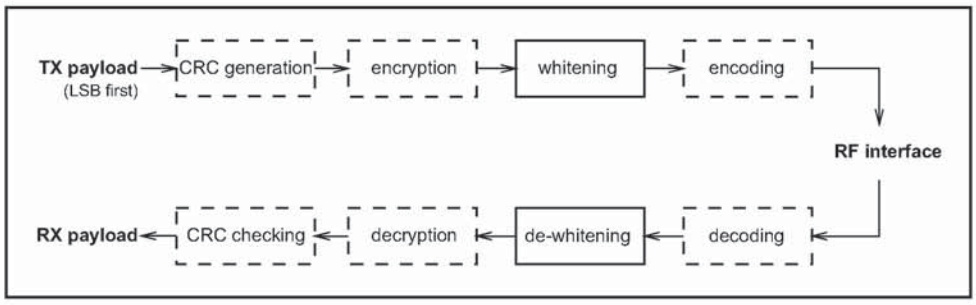

Figure 8.4: Payload bit processes.

9 TRANSMIT/RECEIVE TIMING

The Bluetooth transceiver applies a time-division duplex (TDD) scheme. This means that it alternately transmits and receives in a synchronous manner. It depends on the mode of the Bluetooth unit what the exact timing of the TDD scheme is. In the normal connection mode, *the master transmission shall always start at even numbered time slots (master CLK1=0) and the slave transmission shall always start at odd numbered time slots (master CLK1=1).* Due to packet types that cover more than a single slot, master transmission may continue in odd numbered slots and slave transmission may continue in even numbered slots.

All timing diagrams shown in this chapter are based on the signals as present at the antenna. The term “exact” when used to describe timing refers to an ideal transmission or reception and neglects timing jitter and clock frequency imperfections.

The average timing of master packet transmission must not drift faster than 20 ppm relative to the ideal slot timing of 625 µs. The instantaneous timing must not deviate more than 1 µs from the average timing. Thus, the absolute packet transmission timing t_k of slot boundary k must fulfill the equation:

$$t_k = \left(\sum_{i=1}^{k} (1 + d_i) T_N \right) + j_k + \text{offset}, \qquad \text{(EQ 1)}$$

where T_N is the nominal slot length (625 µs), j_k denotes jitter ($|j_k| \leq 1$ µs) at slot boundary k, and, d_k, denotes the drift ($|d_k| \leq 20$ ppm) within slot k. The jitter and drift may vary arbitrarily within the given limits for every slot, while “offset” is an arbitrary but fixed constant. For hold, park and sniff mode the drift and jitter parameters as described in Link Manager Protocol Section 3.9 on page 203 apply.

9.1 MASTER/SLAVE TIMING SYNCHRONIZATION

The piconet is synchronized by the system clock of the master. The master never adjusts its system clock during the existence of the piconet: it keeps an exact interval of Mx625 µs (where M is an even, positive integer larger than 0) between consecutive transmissions. The slaves adapt their native clocks with a timing offset in order to match the master clock. This offset is updated each time a packet is received from the master: by comparing the exact RX timing of the received packet with the estimated RX timing, the slaves correct the offset for any timing misalignments. Note that the slave RX timing can be corrected with any packet sent in the master-to-slave slot, since only the channel access code is required to synchronize the slave.

The slave TX timing shall be based on the most recent slave RX timing. The RX timing is based on the latest successful trigger during a master-to-slave slot. For ACL links, this trigger must have occurred in the master-to-slave slot directly pre-

ceding the current slave transmission; for SCO links, the trigger may have occurred several master-to-slave slots before since a slave is allowed to send an SCO packet even if no packet was received in the preceding master-to-slave slot. The slave shall be able to receive the packets and adjust the RX timing as long as the timing mismatch remains within the ±10 µs uncertainty window.

The master TX timing is strictly related to the master clock.The master shall keep an exact interval of Mx1250 µs (where M is a positive integer larger than 0) between the start of successive transmissions; the RX timing is based on this TX timing with a shift of exactly Nx625 µs (where N is an odd, positive integer larger than 0). During the master RX cycle, the master will also use the ±10µ uncertainty window to allow for slave misalignments. The master will adjust the RX processing of the considered packet accordingly, but will **not** adjust its RX/TX timing for the following TX and RX cycles.

Timing behaviour may differ slightly depending on the current state of the unit. The different states are described in the next sections.

9.2 CONNECTION STATE

In the connection mode, the Bluetooth transceiver transmits and receives alternately, see Figure 9.1 on page 88 and Figure 9.2 on page 89. In the figures, only single-slot packets are shown as an example. Depending on the type and the payload length, the packet size can be up to 366 µs. Each RX and TX transmission is at a different hop frequency. For multi-slot packets, several slots are covered by the same packet, and the hop frequency used in the first slot will be used throughout the transmission.

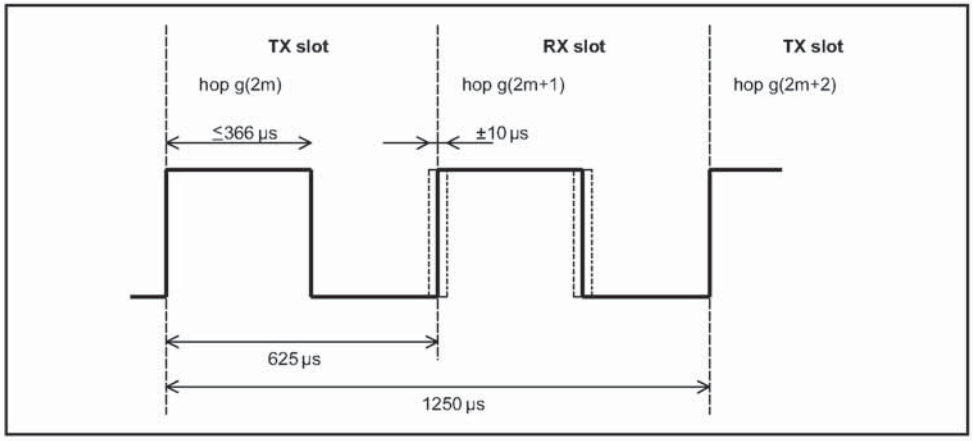

Figure 9.1: RX/TX cycle of Bluetooth master transceiver in normal mode for single-slot packets.

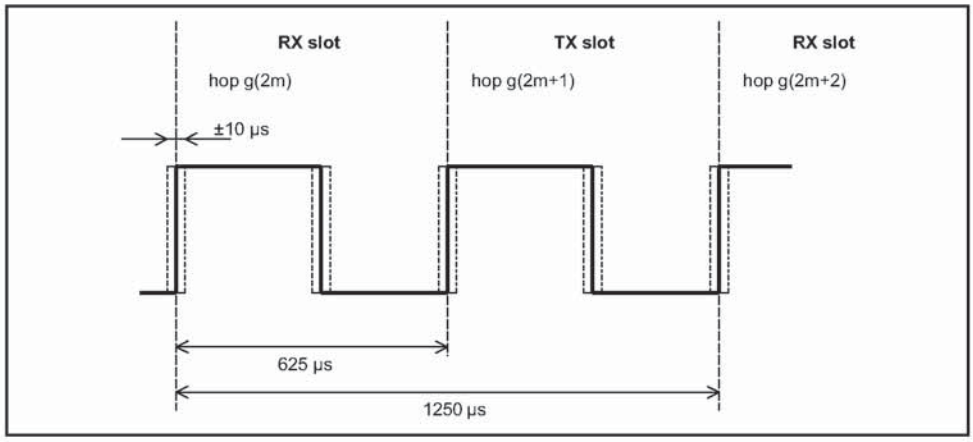

Figure 9.2: RX/TX cycle of Bluetooth slave transceiver in normal mode for single-slot packets.

The master TX/RX timing is shown in Figure 9.1 on page 88. In figures 9.1 through 9.6, f(k) is used for the frequencies of the page hopping sequence and f'(k) denotes the corresponding page response sequence frequencies. The channel hopping frequencies are indicated by g(m). After transmission, a return packet is expected Nx625 µs after the start of the TX burst where N is an odd, positive integer. N depends on the type of the transmitted packet. To allow for some time slipping, an uncertainty window is defined around the exact receive timing. During normal operation, the window length is 20 µs, which allows the RX burst to arrive up to 10 µs too early or 10 µs too late. During the beginning of the RX cycle, the access correlator searches for the correct channel access code over the uncertainty window. If no trigger event occurs, the receiver goes to sleep until the next RX event. If in the course of the search, it becomes apparent that the correlation output will never exceed the final threshold, the receiver may go to sleep earlier. If a trigger event does occur, the receiver remains open to receive the rest of the packet.

The current master transmission is based on the previous master transmission: it is scheduled Mx1250µs after the start of the previous master TX burst where M depends on the transmitted and received packet type. Note that the master TX timing is not affected by time drifts in the slave(s). If no transmission takes place during a number of consecutive slots, the master will take the TX timing of the latest TX burst as reference.

The slave's transmission is scheduled Nx625µs after the start of the slave's RX burst. If the slave's RX timing drifts, so will its TX timing. If no reception takes place during a number of consecutive slots, the slave will take the RX timing of the latest RX burst as reference.

9.3 RETURN FROM HOLD MODE

In the connection state, the Bluetooth unit can be placed in a **hold** mode, see Section 10.8 on page 112. In the **hold** mode, a Bluetooth transceiver neither transmits nor receives information. When returning to the normal operation after a **hold** mode in a slave Bluetooth unit, the slave must listen for the master before it may send information. In that case, the search window in the slave unit may be increased from ±10 µs to a larger value X µs as illustrated in Figure 9.3 on page 90. Note that only RX hop frequencies are used: the hop frequency used in the master-to-slave (RX) slot is also used in the uncertainty window extended into the preceding time interval normally used for the slave-to-master (TX) slot.

If the search window exceeds 625 µs, consecutive windows shall not be centered at the start of RX hops g(2m), g(2m+2), ... g(2m+2i) (where 'i' is an integer), but at g(2m), g(2m+4), ... g(2m+4i), or even at g(2m), g(2m+6), ...g(2m+6i) etc. to avoid overlapping search windows. The RX hop frequencies used shall correspond to the RX slot numbers.

It is recommended that single slot packets are used upon return from hold to minimize the synchronization time, especially after long hold periods that require search windows exceeding 625 µs.

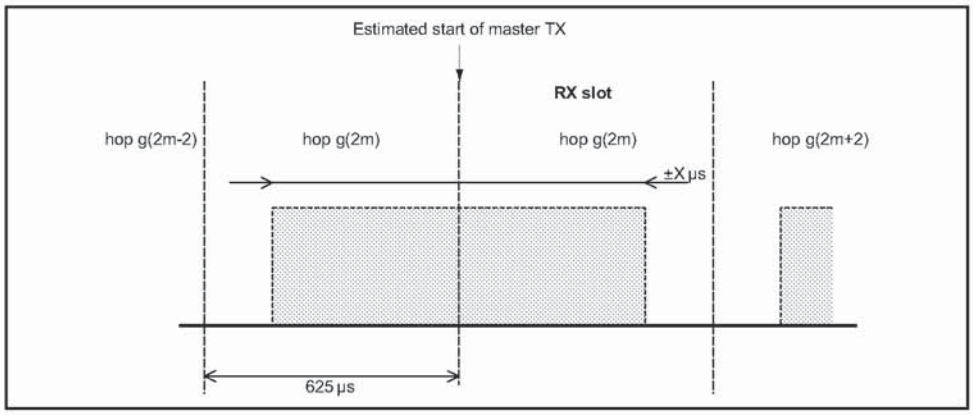

Figure 9.3: RX timing of slave returning from hold state.

9.4 PARK MODE WAKE-UP

The **park** mode is similar to the **hold** mode. A parked slave periodically wakes up to listen to beacons from the master and to re-synchronize its clock offset. As in the return from hold mode, a parked slave when waking up may increase the search window from ±10 µs to a larger value X µs as illustrated in Figure 9.3 on page 90.

9.5 PAGE STATE

In the page state, the master transmits the device access code (ID packet) corresponding to the slave to be connected, rapidly on a large number of different hop frequencies. Since the ID packet is a very short packet, the hop rate can be increased from 1600 hops/s to 3200 hops/s. In a single TX slot interval, the paging master transmits on two different hop frequencies. In a single RX slot interval, the paging transceiver listens on two different hop frequencies; see Figure 9.4 on page 91. During the TX slot, the paging unit sends an ID packet at the TX hop frequencies *f(k)* and *f(k+1)*. In the RX slot, it listens for a response on the corresponding RX hop frequencies *f'(k)* and *f'(k+1)*. The listening periods are exactly timed 625 µs after the corresponding paging packets, and include a ±10 µs uncertainty window.

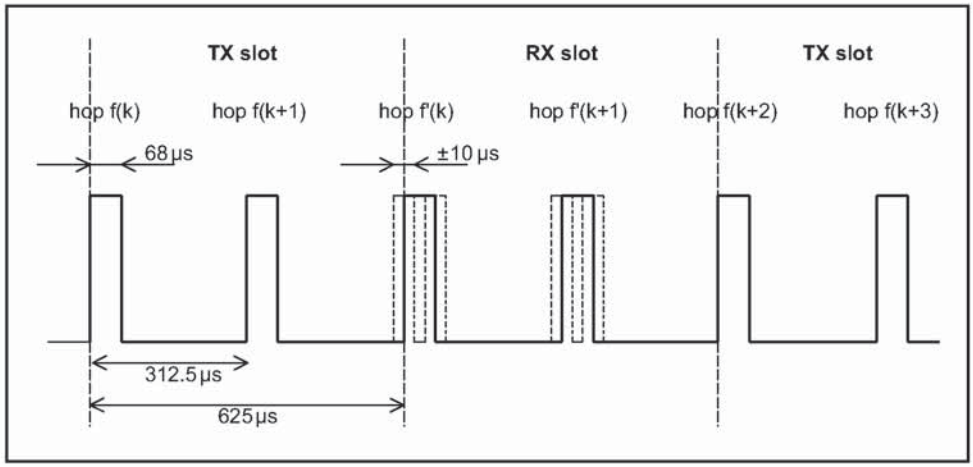

Figure 9.4: RX/TX cycle of Bluetooth transceiver in PAGE mode.

9.6 FHS PACKET

At connection setup and during a master-slave switch, an **FHS** packet is transferred from the master to the slave. This packet will establish the timing and frequency synchronization (see also Section 4.4.1.4 on page 56). After the slave unit has received the page message, it will return a response message which again consists of the ID packet and follows exactly 625 µs after the receipt of the page message. The master will send the FHS packet in the TX slot following the RX slot in which it received the slave response, according the RX/TX timing of the master. The time difference between the response and **FHS** message will depend on the timing of the page message the slave received. In Figure 9.5 on page 92, the slave receives the paging message sent **first** in the master-to-slave slot. It will then respond with an ID packet in the first half of the slave-to-master slot. The timing of the **FHS** packet is based on the timing of the page message sent first in the preceding master-to-slave slot: there is an exact 1250 µs delay between the first page message and the **FHS** packet. The packet is sent at the hop frequency *f(k+1)* which is the hop frequency following the hop frequency *f(k)* the page message was received in. In Figure 9.6 on page 92, the slave receives the paging message sent **secondly** in the master-to-slave slot. It will then respond with an ID packet in the

second half of the slave-to-master slot exactly 625 µs after the receipt of the page message. The timing of the **FHS** packet is still based on the timing of the page message sent **first** in the preceding master-to-slave slot: there is an exact 1250 µs delay between the **first** page message and the **FHS** packet. The packet is sent at the hop frequency *f(k+2)* which is the hop frequency following the hop frequency *f(k+1)* the page message was received in.

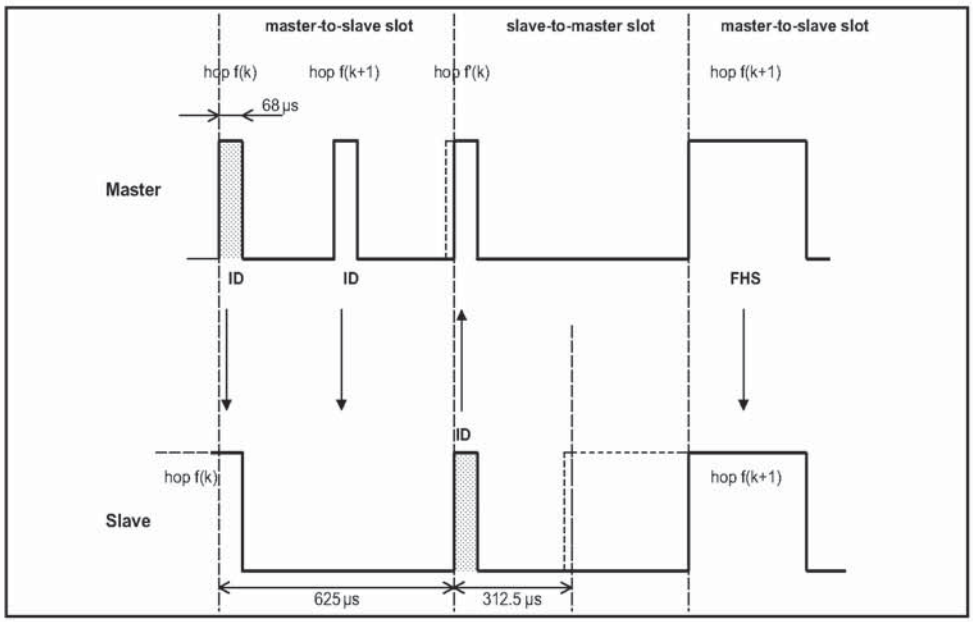

Figure 9.5: Timing of FHS packet on successful page in first half slot.

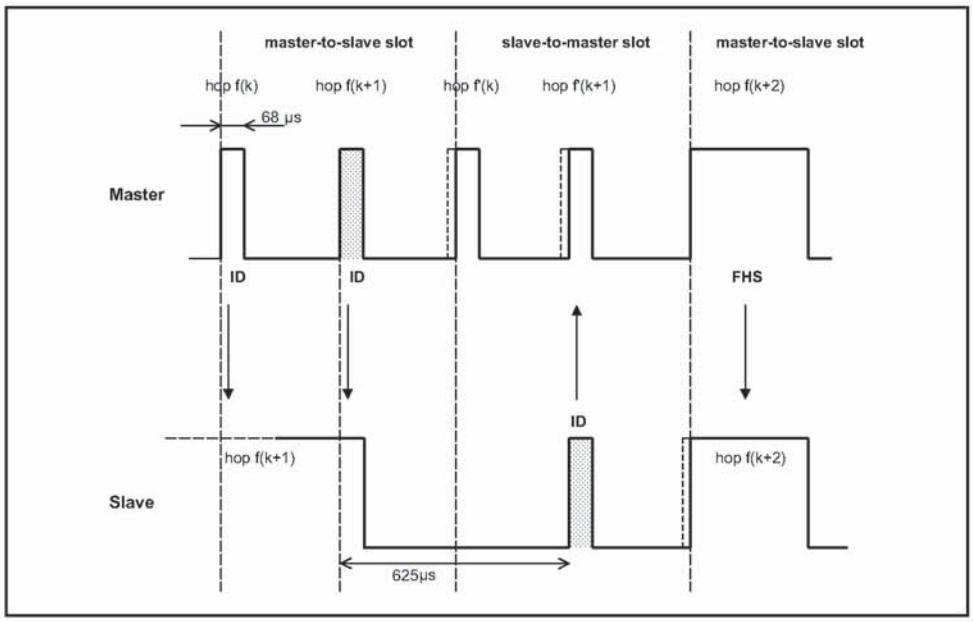

Figure 9.6: Timing of FHS packet on successful page in second half slot.

The slave will adjust its RX/TX timing according to the reception of the **FHS** packet (and not according to the reception of the page message). That is, the second response message that acknowledges the reception of the FHS packet is transmitted 625 µs after the start of the **FHS** packet.

9.7 MULTI-SLAVE OPERATION

As was mentioned in the beginning of this chapter, the master always starts the transmission in the even-numbered slots whereas the slaves start their transmission in the odd-numbered slots. This means that the timing of the master and the slave(s) is shifted by one slot (625 µs), see Figure 9.7 on page 93.

Only the slave that is addressed by its AM_ADDR can return a packet in the next slave-to-master slot. If no valid AM_ADDR is received, the slave may only respond if it concerns its reserved SCO slave-to-master slot. In case of a broadcast message, no slave is allowed to return a packet (an exception is found in the access window for access requests in the park mode, see Section 10.8.4 on page 115).

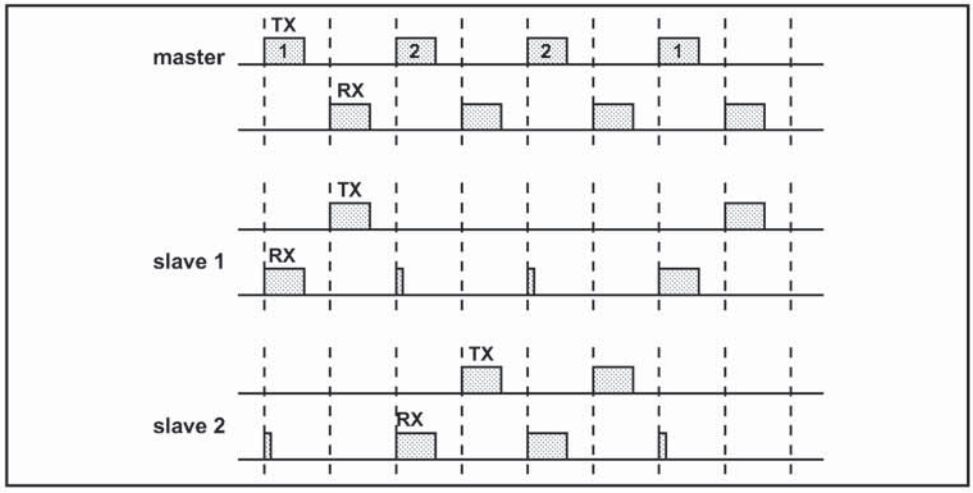

Figure 9.7: RX/TX timing in multi-slave configuration

10 CHANNEL CONTROL

10.1 SCOPE

This section describes how the channel of a piconet is established and how units can be added to and released from the piconet. Several states of operation of the Bluetooth units are defined to support these functions. In addition, the operation of several piconets sharing the same area, the so-called scatternet, is discussed. A special section is attributed to the Bluetooth clock which plays a major role in the FH synchronization.

10.2 MASTER-SLAVE DEFINITION

The channel in the piconet is characterized entirely by the master of the piconet. The Bluetooth device address (BD_ADDR) of the master determines the FH hopping sequence and the channel access code; the system clock of the master determines the phase in the hopping sequence and sets the timing. In addition, the master controls the traffic on the channel by a polling scheme.

By definition, the **master** is represented by the Bluetooth unit that initiates the connection (to one or more **slave** units). Note that the names 'master' and 'slave' only refer to the protocol on the channel: the Bluetooth units themselves are identical; that is, any unit can become a master of a piconet. Once a piconet has been established, master-slave roles can be exchanged. This is described in more detail in Section 10.9.3 on page 123.

10.3 BLUETOOTH CLOCK

Every Bluetooth unit has an internal system clock which determines the timing and hopping of the transceiver. The Bluetooth clock is derived from a free running native clock which is never adjusted and is never turned off. For synchronization with other units, only offsets are used that, added to the native clock, provide temporary Bluetooth clocks which are mutually synchronized. It should be noted that the Bluetooth clock has no relation to the time of day; it can therefore be initialized at any value. The Bluetooth clock provides the heart beat of the Bluetooth transceiver. Its resolution is at least half the TX or RX slot length, or 312.5 µs. The clock has a cycle of about a day. If the clock is implemented with a counter, a 28-bit counter is required that wraps around at 2^{28}-1. The LSB ticks in units of 312.5 µs, giving a clock rate of 3.2 kHz.

The timing and the frequency hopping on the channel of a piconet is determined by the Bluetooth clock of the master. When the piconet is established, the master clock is communicated to the slaves. Each slave adds an offset to its native clock to be synchronized to the master clock. Since the clocks are free-running, the offsets have to be updated regularly.

The clock determines critical periods and triggers the events in the Bluetooth receiver. Four periods are important in the Bluetooth system: 312.5 µs, 625 µs, 1.25 ms, and 1.28 s; these periods correspond to the timer bits CLK_0, CLK_1, CLK_2, and CLK_{12}, respectively, see Figure 10.1 on page 96. Master-to-slave transmission starts at the even-numbered slots when CLK_0 and CLK_1 are both zero.

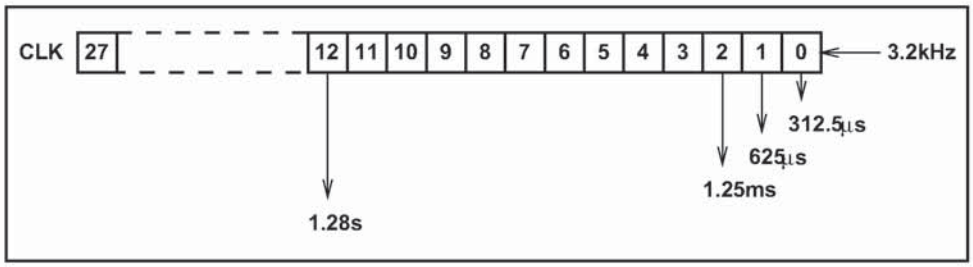

Figure 10.1: Bluetooth clock.

In the different modes and states a Bluetooth unit can reside in, the clock has different appearances:

- CLKN native clock
- CLKE estimated clock
- CLK master clock

CLKN is the free-running native clock and is the reference to all other clock appearances. In states with high activity, the native clock is driven by the reference crystal oscillator with worst case accuracy of +/-20ppm. In the low power states, like **STANDBY, HOLD, PARK**, the native clock may be driven by a low power oscillator (LPO) with relaxed accuracy (+/-250ppm).

CLKE and CLK are derived from the reference CLKN by adding an offset. CLKE is a clock estimate a paging unit makes of the native clock of the recipient; i.e. an offset is added to the CLKN of the pager to approximate the CLKN of the recipient, see Figure 10.2 on page 97. By using the CLKN of the recipient, the pager speeds up the connection establishment.

CLK is the master clock of the piconet. It is used for all timing and scheduling activities in the piconet. All Bluetooth devices use the CLK to schedule their transmission and reception. The CLK is derived from the native clock CLKN by adding an offset, see Figure 10.3 on page 97. The offset is zero for the master since CLK is identical to its own native clock CLKN. Each slave adds an appropriate offset to its CLKN such that the CLK corresponds to the CLKN of the master. Although all CLKNs in the Bluetooth devices run at the same nominal rate, mutual drift causes inaccuracies in CLK. Therefore, the offsets in the slaves must be regularly updated such that CLK is approximately CLKN of the master.

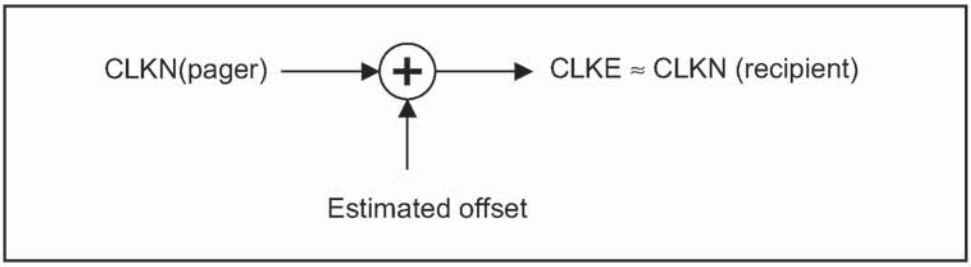

Figure 10.2: Derivation of CLKE

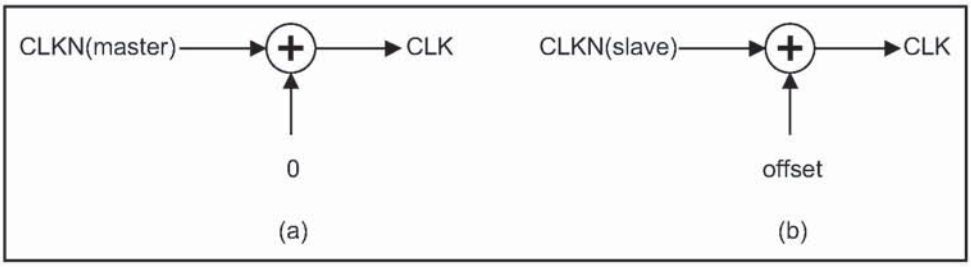

Figure 10.3: Derivation of CLK in master (a) and in slave (b).

10.4 OVERVIEW OF STATES

Figure 10.4 on page 98 shows a state diagram illustrating the different states used in the Bluetooth link controller. There are two major states: **STANDBY** and **CONNECTION**; in addition, there are seven substates, **page**, **page scan**, **inquiry**, **inquiry scan**, **master response**, **slave response**, and **inquiry response**. The substates are interim states that are used to add new slaves to a piconet. To move from one state to the other, either commands from the Bluetooth link manager are used, or internal signals in the link controller are used (such as the trigger signal from the correlator and the timeout signals).

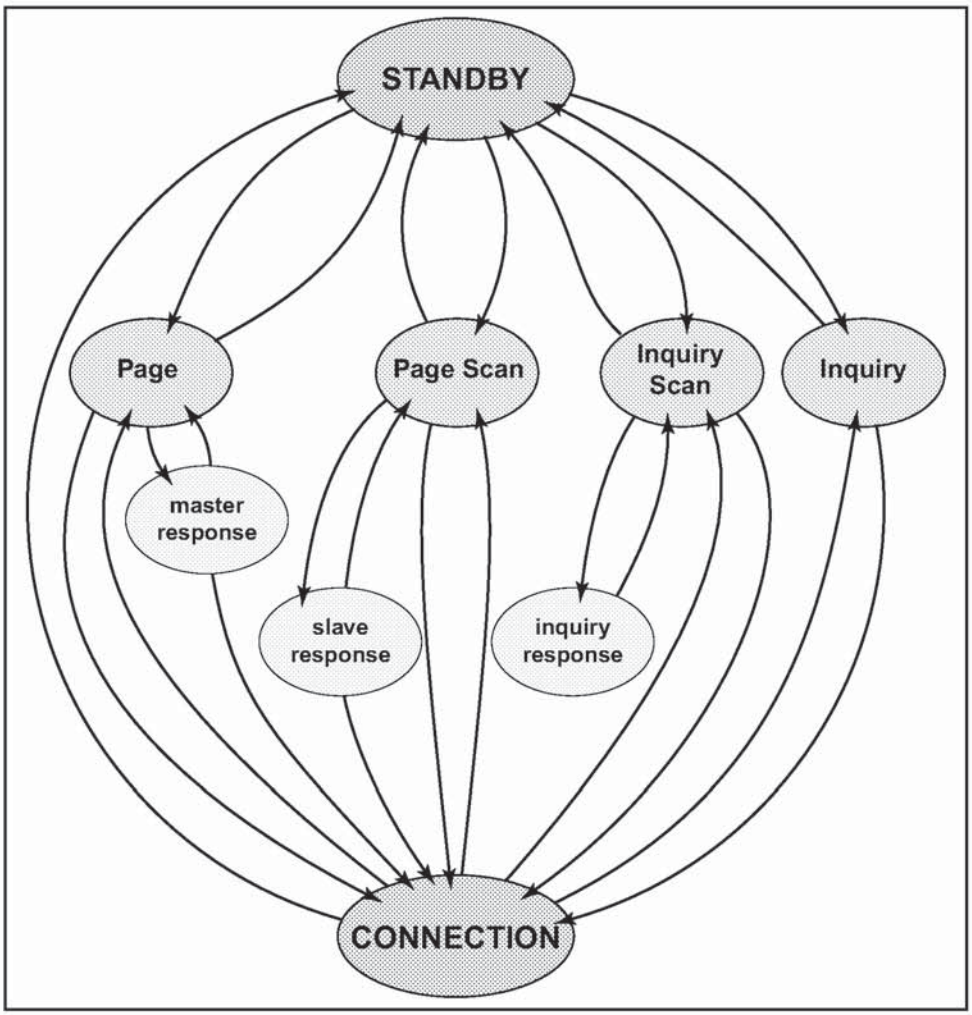

Figure 10.4: State diagram of Bluetooth link controller.

10.5 STANDBY STATE

The **STANDBY** state is the default state in the Bluetooth unit. In this state, the Bluetooth unit is in a low-power mode. Only the native clock is running at the accuracy of the LPO (or better).

The controller may leave the **STANDBY** state to scan for page or inquiry messages, or to page or inquiry itself. When responding to a page message, the unit will not return to the **STANDBY** state but enter the **CONNECTION** state as a slave. When carrying out a successful page attempt, the unit will enter the **CONNECTION** state as a master. The intervals with which scan activities can be carried out are discussed in Section 10.6.2 on page 99 and Section 10.7.2 on page 109.

10.6 ACCESS PROCEDURES

10.6.1 General

In order to establish new connections the procedures inquiry and paging are used. The inquiry procedure enables a unit to discover which units are in range, and what their device addresses and clocks are. With the paging procedure, an actual connection can be established. Only the Bluetooth device address is required to set up a connection. Knowledge about the clock will accelerate the setup procedure. A unit that establishes a connection will carry out a page procedure and will automatically be the master of the connection.

In the paging and inquiry procedures, the device access code (DAC) and the inquiry access code (IAC) are used, respectively. A unit in the **page scan** or **inquiry scan** substate correlates against these respective access codes with a matching correlator.

For the paging process, several paging schemes can be applied. There is one mandatory paging scheme which has to be supported by each Bluetooth device. This mandatory scheme is used when units meet for the first time, and in case the paging process directly follows the inquiry process. Two units, once connected using a mandatory paging/scanning scheme, may agree on an optional paging/scanning scheme. Optional paging schemes are discussed in "Appendix VII" on page 999. In the current chapter, only the mandatory paging scheme is considered.

10.6.2 Page scan

In the **page scan** substate, a unit listens for its own device access code for the duration of the scan window $T_{w\ page\ scan}$. During the scan window, the unit listens at a single hop frequency, its correlator matched to its device access code. The scan window shall be long enough to completely scan 16 page frequencies.

When a unit enters the **page scan** substate, it selects the scan frequency according to the page hopping sequence corresponding to this unit, see Section 11.3.1 on page 135. This is a 32-hop sequence (or a 16-hop sequence in case of a reduced-hop system) in which each hop frequency is unique. The page hopping sequence is determined by the unit's Bluetooth device address (BD_ADDR). The phase in the sequence is determined by $CLKN_{16-12}$ of the unit's native clock ($CLKN_{15-12}$ in case of a reduced-hop system); that is, every 1.28s a different frequency is selected.

If the correlator exceeds the trigger threshold during the **page scan**, the unit will enter the **slave response** substate, which is described in Section 10.6.4.1 on page 105.

The **page scan** substate can be entered from the **STANDBY** state or the **CONNECTION** state. In the **STANDBY** state, no connection has been established and the unit can use all the capacity to carry out the **page scan**. Before entering the **page scan** substate from the **CONNECTION** state, the unit preferably reserves as much capacity for scanning. If desired, the unit may place ACL connections in the HOLD mode or even use the PARK mode, see Section 10.8.3 on page 114 and Section 10.8.4 on page 115. SCO connections are preferably not interrupted by the **page scan**. In this case, the **page scan** may be interrupted by the reserved SCO slots which have higher priority than the **page scan**. SCO packets should be used requiring the least amount of capacity (**HV3** packets). The scan window shall be increased to minimize the setup delay. If one SCO link is present using **HV3** packets and T_{SCO}=6 slots, a total scan window $T_{w\ page\ scan}$ of at least 36 slots (22.5ms) is recommended; if two SCO links are present using **HV3** packets and T_{SCO}=6 slots, a total scan window of at least 54 slots (33.75ms) is recommended.

The scan interval $T_{page\ scan}$ is defined as the interval between the beginnings of two consecutive page scans. A distinction is made between the case where the scan interval is equal to the scan window $T_{w\ page\ scan}$ (continuous scan), the scan interval is maximal 1.28s, or the scan interval is maximal 2.56s. These three cases determine the behavior of the paging unit; that is, whether the paging unit shall use R0, R1 or R2, see also Section 10.6.3 on page 101. Table 10.1 illustrates the relationship between $T_{page\ scan}$ and modes R0, R1 and R2. Although scanning in the R0 mode is continuous, the scanning may be interrupted by for example reserved SCO slots. The scan interval information is included in the SR field in the FHS packet.

During page scan the Bluetooth unit may choose to use an optional scanning scheme. (An exception is the page scan after returning an inquiry response message. See Section 10.7.4 on page 111 for details.)

SR mode	$T_{page\ scan}$	N_{page}
R0	continuous	≥1
R1	≤ 1.28s	≥ 128
R2	≤ 2.56s	≥ 256
Reserved	-	-

Table 10.1: Relationship between scan interval, train repetition, and paging modes R0, R1 and R2.

10.6.3 Page

The **page** substate is used by the master (source) to activate and connect to a slave (destination) which periodically wakes up in the **page scan** substate. The master tries to capture the slave by repeatedly transmitting the slave's device access code (DAC) in different hop channels. Since the Bluetooth clocks of the master and the slave are not synchronized, the master does not know exactly when the slave wakes up and on which hop frequency. Therefore, it transmits a train of identical DACs at different hop frequencies, and listens in between the transmit intervals until it receives a response from the slave.

The page procedure in the master consists of a number of steps. First, the slave's device address is used to determine the page hopping sequence, see Section 11.3.2 on page 135. This is the sequence the master will use to reach the slave. For the phase in the sequence, the master uses an estimate of the slave's clock. This estimate can for example be derived from timing information that was exchanged during the last encounter with this particular device (which could have acted as a master at that time), or from an inquiry procedure. With this estimate CLKE of the slave's Bluetooth clock, the master can predict when the slave wakes up and on which hop channel.

The estimate of the Bluetooth clock in the slave can be completely wrong. Although the master and the slave use the same hopping sequence, they use different phases in the sequence and will never meet each other. To compensate for the clock drifts, the master will send its page message during a short time interval on a number of wake-up frequencies. It will in fact transmit also on hop frequencies just before and after the current, predicted hop frequency. During each TX slot, the master sequentially transmits on two different hop frequencies. Since the page message is the ID packet which is only 68 bits in length, there is ample of time (224.5 µs minimal) to switch the synthesizer. In the following RX slot, the receiver will listen sequentially to two corresponding RX hops for ID packet. The RX hops are selected according to the page_response hopping sequence. The page_response hopping sequence is strictly related to the page hopping sequence; that is: for each page hop there is a corresponding page_response hop. The RX/TX timing in the **page** sub-

state has been described in Section 9, see also Figure 9.4 on page 91. In the next TX slot, it will transmit on two hop frequencies different from the former ones. The synthesizer hop rate is increased to 3200 hops/s.

A distinction must be made between the 79-hop systems and the 23-hop systems. First the 79-hop systems are considered. With the increased hopping rate as described above, the transmitter can cover 16 different hop frequencies in 16 slots or 10 ms. The page hopping sequence is divided over two paging trains **A** and **B** of 16 frequencies. Train **A** includes the 16 hop frequencies surrounding the current, predicted hop frequency f(k), where k is determined by the clock estimate $CLKE_{16-12}$. So the first train consists of hops

f(k-8), f(k-7),...,f(k),...,f(k+7)

When the difference between the Bluetooth clocks of the master and the slave is between -8x1.28 s and +7x1.28 s, one of the frequencies used by the master will be the hop frequency the slave will listen to. However, since the master does not know when the slave will enter the **page scan** substate, he has to repeat this train **A** N_{page} times or until a response is obtained. If the slave scan interval corresponds to R1, the repetition number is at least 128; if the slave scan interval corresponds to R2, the repetition number is at least 256.
Note that $CLKE_{16-12}$ changes every 1.28 s; therefore, every 1.28 s, the trains will include different frequencies of the page hopping set.

When the difference between the Bluetooth clocks of the master and the slave is less than -8x1.28 s or larger than +7x1.28 s, more distant hops must be probed. Since in total, there are only 32 dedicated wake-up hops, the more distant hops are the remaining hops not being probed yet. The remaining 16 hops are used to form the new 10 ms train **B.** The second train consists of hops

f(k-16), f(k-15),...,f(k-9),f(k+8)...,f(k+15)

Train **B** is repeated for N_{page} times. If still no response is obtained, the first train **A** is tried again N_{page} times. Alternate use of train A and train B is continued until a response is received or the timeout *pageTO* is exceeded. If during one of the listening occasions, a response is returned by the slave, the master unit enters the **master response** substate.

The description for paging and **page scan** procedures given here has been tailored towards the 79-hop systems used in the US and Europe. For the 23-hop systems as used in Japan and some European countries, the procedure is slightly different. In the 23-hop case, the length of the page hopping sequence is reduced to 16. As a consequence, there is only a single train (train **A**) including all the page hopping frequencies. The phase to the page hopping sequence is not $CLKE_{16-12}$ but $CLKE_{15-12}$. An estimate of the slave's clock does not have to be made.

The **page** substate can be entered from the **STANDBY** state or the **CONNECTION** state. In the **STANDBY** state, no connection has been established and

the unit can use all the capacity to carry out the page. Before entering the page substate from the CONNECTION state, the unit shall free as much capacity as possible for scanning. To ensure this, it is recommended that the ACL connections are put on hold or park. However, the SCO connections shall not be disturbed by the page. This means that the page will be interrupted by the reserved SCO slots which have higher priority than the page. In order to obtain as much capacity for paging, it is recommended to use the SCO packets which use the least amount of capacity (**HV3** packets). If SCO links are present, the repetition number N_{page} of a single train shall be increased, see Table 10.2. Here it has been assumed that the **HV3** packet are used with an interval T_{SCO}=6 slots, which would correspond to a 64 kb/s voice link.

SR mode	no SCO link	one SCO link (HV3)	two SCO links (HV3)
R0	$N_{page} \geq 1$	$N_{page} \geq 2$	$N_{page} \geq 3$
R1	$N_{page} \geq 128$	$N_{page} \geq 256$	$N_{page} \geq 384$
R2	$N_{page} \geq 256$	$N_{page} \geq 512$	$N_{page} \geq 768$

Table 10.2: Relationship between train repetition, and paging modes R0, R1 and R2 when SCO links are present.

The construction of the page train is independent on the presence of SCO links; that is, SCO packets are sent on the reserved slots but do not affect the hop frequencies used in the unreserved slots, see Figure 10.5 on page 103.

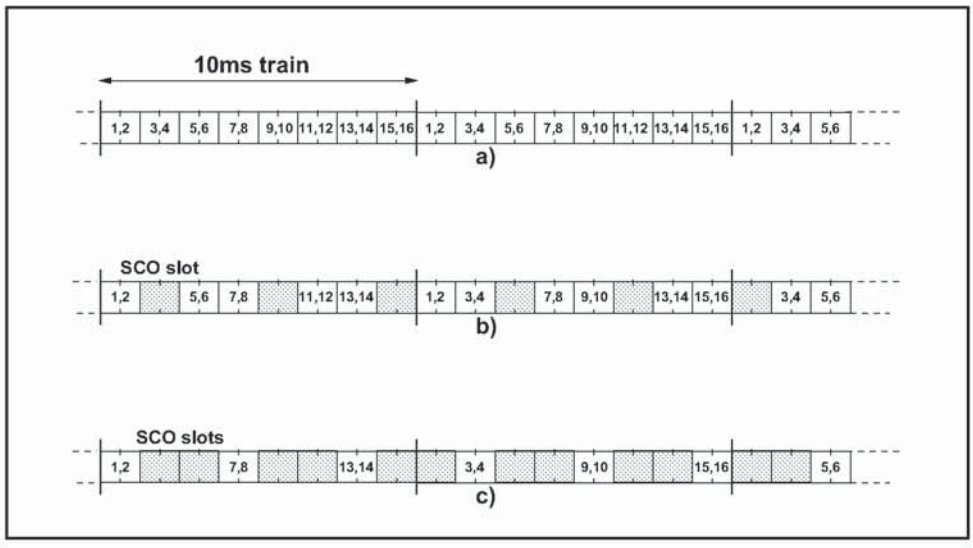

Figure 10.5: Conventional page (a), page while one SCO link present (b), page while two SCO links present (c).

For the descriptions of optional paging schemes see "Appendix VII" on page 999.

10.6.4 Page response procedures

When a page message is successfully received by the slave, there is a coarse FH synchronization between the master and the slave. Both the master and the slave enter a response routine to exchange vital information to continue the connection setup. Important for the piconet connection is that both Bluetooth units use the same channel access code, use the same channel hopping sequence, and that their clocks are synchronized. These parameters are derived from the master unit. The unit that initializes the connection (starts paging) is defined as the master unit (which is thus only valid during the time the piconet exists). The channel access code and channel hopping sequence are derived from the Bluetooth device address (BD_ADDR) of the master. The timing is determined by the master clock. An offset is added to the slave's native clock to temporarily synchronize the slave clock to the master clock. At start-up, the master parameters have to be transmitted from the master to the slave. The messaging between the master and the slave at start-up will be considered in this section.

The initial messaging between master and slave is shown in Table 10.3 on page 104 and in Figure 10.6 on page 105 and Figure 10.7 on page 105. In those two figures frequencies f (k), f(k+1), etc. are the frequencies of the page hopping sequence determined by the slave's BD_ADDR. The frequencies f'(k), f'(k+1), etc. are the corresponding page_response frequencies (slave-to-master). The frequencies g(m) belong to the channel hopping sequence.

Step	Message	Direction	Hopping Sequence	Access Code and Clock
1	slave ID	master to slave	page	slave
2	slave ID	slave to master	page response	slave
3	FHS	master to slave	page	slave
4	slave ID	slave to master	page response	slave
5	1st packet master	master to slave	channel	master
6	1st packet slave	slave to master	channel	master

Table 10.3: Initial messaging during start-up.

In step 1 (see Table 10.3 on page 104), the master unit is in **page** substate and the slave unit in the **page scan** substate. Assume in this step that the page message (= slave's device access code) sent by the master reaches the slave. On recognizing its device access code, the slave enters the **slave response** in step 2. The master waits for a reply from the slave and when this arrives in step 2, it will enter the **master response** in step 3. Note that during the initial message exchange, all parameters are derived from the slave's BD_ADDR, and that only the page hopping and page_response hopping sequences are used

(which are also derived from the slave's BD_ADDR). Note that when the master and slave enter the response states, their clock input to the page and page_response hop selection is frozen as is described in Section 11.3.3 on page 136.

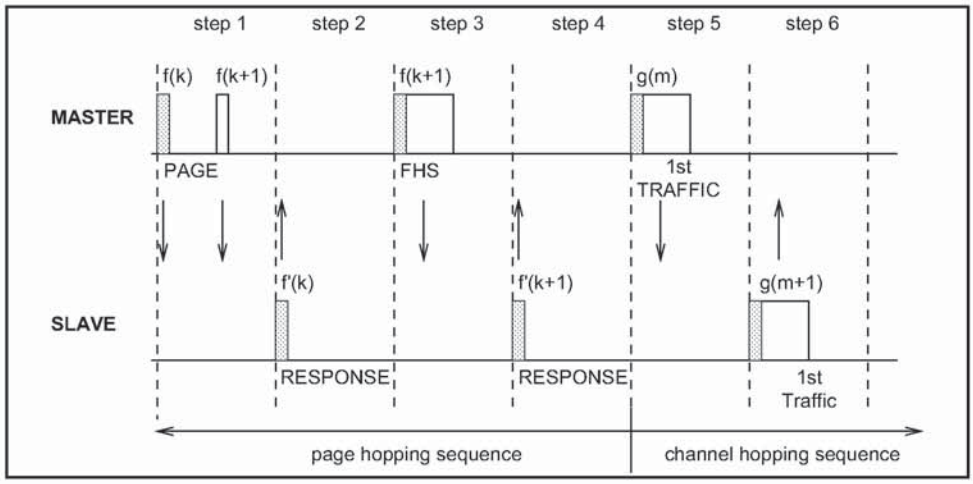

Figure 10.6: Messaging at initial connection when slave responds to first page message.

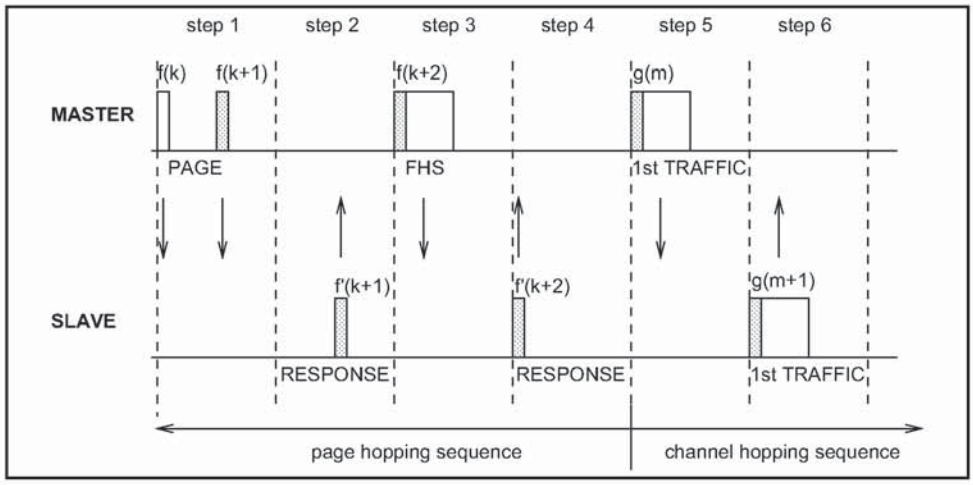

Figure 10.7: Messaging at initial connection when slave responds to second page message.

10.6.4.1 Slave response

After having received its own device access code in step 1, the slave unit transmits a response message in step 2. This response message again only consists of the slave's device access code. The slave will transmit this response 625 µs after the beginning of the received page message (slave ID packet) and at the response hop frequency that corresponds to the hop frequency in which the page message was received. The slave transmission is therefore time

aligned to the master transmission. During initial messaging, the slave still uses the page response hopping sequence to return information to the master. The clock input $CLKN_{16-12}$ is frozen at the value it had at the time the page message was received.

After having sent the response message, the slave's receiver is activated (312.5 µs after the start of the response message) and awaits the arrival of a **FHS** packet. Note that a **FHS** packet can already arrive 312.5 µs after the arrival of the page message as shown in Figure 10.7 on page 105, and not after 625 µs as is usually the case in the RX/TX timing. More details about the timing can be found in Section 9.6 on page 91.

If the setup fails before the **CONNECTION** state has been reached, the following procedure is carried out. The slave will keep listening as long as no **FHS** packet is received until *pagerespTO* is exceeded. Every 1.25 ms, however, it will select the next master-to-slave hop frequency according to the page hop sequence. If nothing is received after *pagerespTO*, the slave returns back to the **page scan** substate for one scan period. Length of the scan period depends on the SCO slots present. If no page message is received during this additional scan period, the slave will resume scanning at its regular scan interval and return to the state it was in prior to the first page scan state.

If a **FHS** packet is received by the slave in the **slave response** substate, the slave returns a response (slave's device access code only) in step 4 to acknowledge the reception of the **FHS** packet (still using the page response hopping sequence). The transmission of this response packet is based on the reception of the **FHS** packet. Then the slave changes to the channel (master's) access code and clock as received from the **FHS** packet. Only the 26 MSBs of the master clock are transferred: the timing is assumed such that CLK_1 and CLK_0 are both zero at the time the **FHS** packet was received as the master transmits in even slots only. From the master clock in the **FHS** packet, the offset between the master's clock and the slave's clock is determined and reported to the slave's link manager.

Finally, the slave enters the **CONNECTION** state in step 5. From then on, the slave will use the master's clock and the master BD_ADDR to determine the channel hopping sequence and the channel access code. The connection mode starts with a POLL packet transmitted by the master. The slave responds with any type of packet. If the POLL packet is not received by the slave, or the response packet is not received by the master, within *newconnectionTO* number of slots after FHS packet acknowledgement, the master and the slave will return to page and page scan substates, respectively.See Section 10.8 on page 112

10.6.4.2 Master response

When the master has received a response message from the slave in step 2, it will enter the **master response** routine. It freezes the current clock input to the page hop selection scheme. Then the master will transmit a **FHS** packet in step 3 containing the master's real-time Bluetooth clock, the master's 48-bit BD_ADDR address, the BCH parity bits, and the class of device. The **FHS** packet contains all information to construct the channel access code without requiring a mathematical derivation from the master device address. The **FHS** packet is transmitted at the beginning of the master-to-slave slot following the slot in which the slave has responded. So the TX timing of the **FHS** is not based on the reception of the response packet from the slave. The **FHS** packet may therefore be sent 312.5 µs after the reception of the response packet like shown in Figure 10.7 on page 105 and not 625 µs after the received packet as is usual in the RX/TX timing, see also Section 9.6 on page 91.

After the master has sent its **FHS** packet, it waits for a second response from the slave in step 4 which acknowledges the reception of the **FHS** packet. Again this is only the slave's device access code. If no response is received, the master retransmits the **FHS** packet, but with an updated clock and still using the slave's parameters. It will retransmit (the clock is updated every retransmission) until a second slave response is received, or the timeout of *pagerespTO* is exceeded. In the latter case, the master turns back to the **page** substate and sends an error message to the link manager. During the retransmissions of the **FHS** packet, the master keeps using the page hopping sequence.

If the slave's response is indeed received, the master changes to the master parameters, so the channel access code and the master clock. The lower clock bits CLK_0 and CLK_1 are zero at the start of the **FHS** packet transmission and are not included in the **FHS** packet. Finally, the master enters the **CONNECTION** state in step 5. The master BD_ADDR is used to change to a new hopping sequence, the *channel hopping sequence*. The channel hopping sequence uses all 79 hop channels in a (pseudo) random fashion, see also Section 11.3.6 on page 138. The master can now send its first traffic packet in a hop determined with the new (master) parameters. This first packet will be a POLL packet. See Section 10.8 on page 112.

The master can now send its first traffic packet in a hop determined with the new (master) parameters. The first packet in this state is a POLL packet sent by the master. This packet will be sent within *newconnectionTO* number of slots after reception of the FHS packet acknowledgement. The slave will respond with any type of packet. If the POLL packet is not received by the slave or the POLL packet response is not received by the master within *newconnectionTO* number of slots, the master and the slave will return to page and page scan substates, respectively.

10.7 INQUIRY PROCEDURES

10.7.1 General

In the Bluetooth system, an inquiry procedure is defined which is used in applications where the destination's device address is unknown to the source. One can think of public facilities like printers or facsimile machines, or access points to a LAN. Alternatively, the inquiry procedure can be used to discover which other Bluetooth units are within range. During an **inquiry** substate, the discovering unit collects the Bluetooth device addresses and clocks of all units that respond to the inquiry message. It can then, if desired, make a connection to any one of them by means of the previously described page procedure.

The inquiry message broadcasted by the source does not contain any information about the source. However, it may indicate which class of devices should respond. There is one general inquiry access code (GIAC) to inquire for any Bluetooth device, and a number of dedicated inquiry access codes (DIAC) that only inquire for a certain type of devices. The inquiry access codes are derived from reserved Bluetooth device addresses and are further described in Section 4.2.1.

A unit that wants to discover other Bluetooth units enters an **inquiry** substate. In this substate, it continuously transmits the inquiry message (which is the ID packet, see Section 4.4.1.1 on page 55) at different hop frequencies. The **inquiry** hop sequence is always derived from the LAP of the GIAC. Thus, even when DIACs are used, the applied hopping sequence is generated from the GIAC LAP. A unit that allows itself to be discovered, regularly enters the **inquiry scan** substate to respond to inquiry messages. The following sections describe the message exchange and contention resolution during inquiry response. The inquiry response is optional: a unit is not forced to respond to an inquiry message.

10.7.2 Inquiry scan

The **inquiry scan** substate is very similar to the **page scan** substate. However, instead of scanning for the unit's device access code, the receiver scans for the inquiry access code long enough to completely scan for 16 inquiry frequencies. The length of this scan period is denoted $T_{w\_inquiry\_scan}$. The scan is performed at a single hop frequency. As in the page procedure, the inquiry procedure uses 32 dedicated inquiry hop frequencies according to the *inquiry hopping sequence*. These frequencies are determined by the general inquiry address. The phase is determined by the native clock of the unit carrying out the **inquiry scan**; the phase changes every 1.28s.

Instead or in addition to the general inquiry access code, the unit may scan for one or more dedicated inquiry access codes. However, the scanning will follow the inquiry hopping sequence which is determined by the general inquiry address. If an inquiry message is recognized during an inquiry wake-up period, the Bluetooth unit enters the **inquiry response** substate.

The **inquiry scan** substate can be entered from the **STANDBY** state or the **CONNECTION** state. In the **STANDBY** state, no connection has been established and the unit can use all the capacity to carry out the **inquiry scan**. Before entering the **inquiry scan** substate from the **CONNECTION** state, the unit preferably reserves as much capacity as possible for scanning. If desired, the unit may place ACL connections in the HOLD mode or even use the PARK mode, see Section 10.8.3 on page 114. SCO connections are preferably not interrupted by the **inquiry scan**. In this case, the **inquiry scan** may be interrupted by the reserved SCO slots which have higher priority than the **inquiry scan**. SCO packets should be used requiring the least amount of capacity (**HV3** packets). The scan window, $T_{w\ inquiry\ scan}$, shall be increased to increase the probability to respond to an inquiry message. If one SCO link is present using HV3 packets and T_{SCO}=6 slots, a total scan window of at least 36 slots (22.5ms) is recommended; if two SCO links are present using HV3 packets and T_{SCO}=6 slots, a total scan window of at least 54 slots (33.75ms) is recommended.

The scan interval $T_{inquiry\ scan}$ is defined as the interval between two consecutive inquiry scans. The **inquiry scan** interval shall be at most 2.56 s.

10.7.3 Inquiry

The **inquiry** substate is used by the unit that wants to discover new devices. This substate is very similar to the **page** substate, the same TX/RX timing is used as used for paging, see Section 9.6 on page 91 and Figure 9.4 on page 91. The TX and RX frequencies follow the inquiry hopping sequence and the inquiry response hopping sequence, and are determined by the general inquiry access code and the native clock of the discovering device. In between inquiry transmissions, the Bluetooth receiver scans for inquiry response messages. When found, the entire response packet (which is in fact a **FHS** packet) is read, after which the unit continues with the inquiry transmissions. So the Bluetooth unit in an **inquiry** substate does not acknowledge the inquiry response messages. It keeps probing at different hop channels and in between listens for response packets. Like in the **page** substate, two 10 ms trains **A** and **B** are defined, splitting the 32 frequencies of the inquiry hopping sequence into two 16-hop parts. A single train must be repeated for at least $N_{inquiry}$=256 times before a new train is used. In order to collect all responses in an error-free environment, at least three train switches must have taken place. As a result, the **inquiry** substate may have to last for 10.24 s unless the inquirer collects enough responses and determines to abort the inquiry substate earlier. If desired, the inquirer can also prolong the inquiry substate to increase the probability of receiving all responses in an error-prone environment. If an inquiry procedure is automatically initiated periodically (say a 10 s period every minute), then the interval between two inquiry instances must be determined randomly. This is done to avoid two Bluetooth units to synchronize their inquiry procedures.

The **inquiry** substate is continued until stopped by the Bluetooth link manager (when it decides that it has sufficient number of responses), or when a timeout has been reached (*inquiryTO*).

The **inquiry** substate can be entered from the **STANDBY** state or the **CONNECTION** state. In the **STANDBY** state, no connection has been established and the unit can use all the capacity to carry out the inquiry. Before entering the inquiry substate from the **CONNECTION** state, the unit shall free as much capacity as possible for scanning. To ensure this, it is recommended that the ACL connections are put on hold or park. However, the SCO connections shall not be disturbed by the inquiry. This means that the inquiry will be interrupted by the reserved SCO slots which have higher priority than the inquiry. In order to obtain as much capacity for inquiry, it is recommended to use the SCO packets which use the least amount of capacity (**HV3** packets). If SCO links are present, the repetition number $N_{inquiry}$ shall be increased, see Table 10.4 on page 111.
Here it has been assumed that the **HV3** packet are used with an interval T_{SCO}=6 slots, which would correspond to a 64 kb/s voice link.

	no SCO link	one SCO link (HV3)	two SCO links (HV3)
$N_{inquiry}$	≥ 256	≥ 512	≥ 768

Table 10.4: Increase of train repetition when SCO links are present.

10.7.4 Inquiry response

For the inquiry operation, there is only a slave response, no master response. The master listens between inquiry messages for responses, but after reading a response, it continues to transmit inquiry messages. The slave response routine for inquiries differs completely from the slave response routine applied for pages. When the inquiry message is received in the **inquiry scan** substate, a response message containing the recipient's address must be returned. This response message is a conventional **FHS** packet carrying the unit's parameters. However, a contention problem may arise when several Bluetooth units are in close proximity to the inquiring unit and all respond to an inquiry message at the same time. First of all, every Bluetooth unit has a free running clock; therefore, it is highly unlikely that they all use the same phase of the inquiry hopping sequence. However, in order to avoid collisions between units that do wake up in the same inquiry hop channel simultaneously, the following protocol in the slave's **inquiry response** is used. If the slave receives an inquiry message, it generates a random number RAND between 0 and 1023. In addition, it freezes the current input value (phase) to the hop selection scheme, see also Section 11.3.5 on page 137. The slave then returns to the **CONNECTION** or **STANDBY** state for the duration of RAND time slots. Before returning to the **CONNECTION** or **STANDBY** state, the unit may go through the page scan substate; this page scan must use the mandatory page scan scheme. After at least RAND slots, the unit will return to the **inquiry response** substate. On the first inquiry message received the slave returns an **FHS** response packet to the master. If during the scan no trigger occurs within a timeout period of *inqrespTO*, the slave returns to the **STANDBY** or **CONNECTION** state. If the unit does receive an inquiry message and returns an FHS packet, it adds an offset of 1 to the phase in the inquiry hop sequence (the phase has a 1.28 s resolution) and enters the **inquiry scan** substate again. If the slave is triggered again, it repeats the procedure using a new RAND. The offset to the clock accumulates each time a **FHS** packet is returned. During a 1.28 s probing window, a slave on average responses 4 times, but on different frequencies and at different times. Possible SCO slots should have priority over response packets; that is, if a response packet overlaps with an SCO slot, it is not sent but the next inquiry message is awaited.

The messaging during the inquiry routines is summarized in Table 10.5 on page 112. In step 1, the master transmits an inquiry message using the inquiry access code and its own clock. The slave responds with the **FHS** packet which contains the slave's device address, native clock and other slave information. This **FHS** packet is returned at a semi-random time. The **FHS** packet is not acknowledged in the inquiry routine, but it is retransmitted at other times and frequencies as long as the master is probing with inquiry messages.

step	message	direction	hopping sequence	access code
1	ID	master to slave	inquiry	inquiry
2	FHS	slave to master	inquiry response	inquiry

Table 10.5: Messaging during inquiry routines.

If the scanning unit uses an optional scanning scheme, after responding to an inquiry with an FHS packet, it will perform page scan using the mandatory page scan scheme for $T_{mandatory\ pscan}$ period. Every time an inquiry response is sent the unit will start a timer with a timeout of $T_{mandatory\ pscan}$. The timer will be reset at each new inquiry response. Until the timer times out, when the unit performs page scan, it will use the mandatory page scanning scheme in the SR mode it uses for all its page scan intervals. Using the mandatory page scan scheme after the inquiry procedure enables all units to connect even if they do not support an optional paging scheme (yet). In addition to using the mandatory page scan scheme, an optional page scan scheme can be used in parallel for the $T_{mandatory\ pscan}$ period.

The $T_{mandatory\ pscan}$ period is included in the SP field of the FHS packet returned in the inquiry response routine, see Section 4.4.1.4 on page 56. The value of the period is indicated in the Table 10.6

SP mode	$T_{mandatory\ pscan}$
P0	≥20s
P1	≥ 40s
P2	≥ 60s
Reserved	-

Table 10.6: Mandatory scan periods for P0, P1, P2 scan period modes.

10.8 CONNECTION STATE

In the **CONNECTION** state, the connection has been established and packets can be sent back and forth. In both units, the channel (master) access code and the master Bluetooth clock are used. The hopping scheme uses the *channel hopping sequence*. The master starts its transmission in even slots (CLK_{1-0}=00), the slave starts its transmission in odd slots (CLK_{1-0}=10)

The **CONNECTION** state starts with a POLL packet sent by the master to verify the switch to the master's timing and channel frequency hopping. The slave can respond with any type of packet. If the slave does not receive the POLL packet or the master does not receive the response packet for *newconnectionTO* number of slots, both devices will return to **page**/**page scan** substates.

The first information packets in the **CONNECTION** state contain control messages that characterize the link and give more details regarding the Bluetooth units. These messages are exchanged between the link managers of the units. For example, it defines the SCO links and the sniff parameters. Then the transfer of user information can start by alternately transmitting and receiving packets.

The **CONNECTION** state is left through a **detach** or **reset** command. The **detach** command is used if the link has been disconnected in the normal way. All configuration data in the Bluetooth link controller is still valid. The **reset** command is a hard reset of all controller processes. After a reset, the controller has to be reconfigured.

The Bluetooth units can be in several modes of operation during the **CONNECTION** state: active mode, sniff mode, hold mode, and park mode. These modes are now described in more detail.

10.8.1 Active mode

In the active mode, the Bluetooth unit actively participates on the channel. The master schedules the transmission based on traffic demands to and from the different slaves. In addition, it supports regular transmissions to keep slaves synchronized to the channel. Active slaves listen in the master-to-slave slots for packets. If an active slave is not addressed, it may sleep until the next new master transmission. From the type indication in the packet, the number of slots the master has reserved for its transmission can be derived; during this time, the non-addressed slaves do not have to listen on the master-to-slave slots. A periodic master transmission is required to keep the slaves synchronized to the channel. Since the slaves only need the channel access code to synchronize with, any packet type can be used for this purpose.

10.8.2 Sniff mode

In the sniff mode, the duty cycle of the slave's listen activity can be reduced. If a slave participates on an ACL link, it has to listen in every ACL slot to the master traffic. With the sniff mode, the time slots where the master can start transmission to a specific slave is reduced; that is, the master can only start transmission in specified time slots. These so-called sniff slots are spaced regularly with an interval of T_{sniff}.

The slave has to listen at D_{sniff} slot every sniff period, T_{sniff} for a $N_{sniff\ attempt}$ number of times. If the slave receives a packet in one of the $N_{sniff\ attempt}$ RX slots, it should continue listening as long as it receives packets to its own AM_ADDR. Once it stops receiving packets, it should continue listening for $N_{sniff\ timeout}$ RX slots or remaining of the $N_{sniff\ attempt}$ number of RX slots, whichever is greater.

To enter the sniff mode, the master shall issue a sniff command via the LM protocol. This message will contain the sniff interval T_{sniff} and an offset D_{sniff}. The timing of the sniff mode is then determined similar as for the SCO links. In addition, an initialization flag indicates whether initialization procedure 1 or 2 is being used. The master uses initialization 1 when the MSB of the current master clock (CLK_{27}) is 0; it uses initialization 2 when the MSB of the current master clock (CLK_{27}) is 1. The slave shall apply the initialization method as indicated by the initialization flag irrespective of its clock bit value CLK_{27}. The master-to-slave sniff slots determined by the master and the slave shall be initialized on the slots for which the clock satisfies the following equation

$$CLK_{27-1} \bmod T_{sniff} = D_{sniff} \qquad \text{for initialization 1}$$

$$(\overline{CLK}_{27}, CLK_{26-1}) \bmod T_{sniff} = D_{sniff} \qquad \text{for initialization 2}$$

The slave-to-master sniff slot determined by the master and the slave shall be initialized on the slots after the master-to-slave sniff slot defined above. After initialization, the clock value CLK(k+1) for the next master-to-slave SNIFF slot is found by adding the fixed interval T_{sniff} to the clock value of the current master-to-slave sniff slot:

$$CLK(k+1) = CLK(k) + T_{sniff}$$

10.8.3 Hold mode

During the **CONNECTION** state, the ACL link to a slave can be put in a **hold** mode. This means that the slave temporarily does not support ACL packets on the channel any more (note: possible SCO links will still be supported). With the **hold** mode, capacity can be made free to do other things like scanning, paging, inquiring, or attending another piconet. The unit in **hold** mode can also enter a low-power sleep mode. During the **hold** mode, the slave unit keeps its active member address (AM_ADDR).

Prior to entering the hold mode, master and slave agree on the time duration the slave remains in the hold mode. A timer is initialized with the *holdTO* value. When the timer is expired, the slave will wake up, synchronize to the traffic on the channel and will wait for further master instructions.

10.8.4 Park mode

When a slave does not need to participate on the piconet channel, but still wants to remain synchronized to the channel, it can enter the park mode which is a low-power mode with very little activity in the slave. In the park mode, the slave gives up its active member address AM_ADDR. Instead, it receives two new addresses to be used in the park mode

- PM_ADDR: 8-bit Parked Member Address
- AR_ADDR: 8-bit Access Request Address

The PM_ADDR distinguishes a parked slave from the other parked slaves. This address is used in the master-initiated unpark procedure. In addition to the PM_ADDR, a parked slave can also be unparked by its 48-bit BD_ADDR. The all-zero PM_ADDR is a reserved address: if a parked unit has the all-zero PM_ADDR it can only be unparked by the BD_ADDR. In that case, the PM_ADDR has no meaning. The AR_ADDR is used by the slave in the slave-initiated unpark procedure. All messages sent to the parked slaves have to be carried by broadcast packets (the all-zero AM_ADDR) because of the missing AM_ADDR.

The parked slave wakes up at regular intervals to listen to the channel in order to re-synchronize and to check for broadcast messages. To support the synchronization and channel access of the parked slaves, the master supports a beacon channel described in the next section. The beacon structure is communicated to the slave when it is being parked. When the beacon structure changes, the parked slaves are updated through broadcast messages.

In addition for using it for low power consumption, the park mode is used to connect more than seven slaves to a single master. At any one time, only seven slaves can be active. However, by swapping active and parked slaves out respectively in the piconet, the number of slave virtually connected can be much larger (255 if the PM_ADDR is used, and even a larger number if the BD_ADDR is used). There is no limitation to the number of slaves that can be parked.

10.8.4.1 Beacon channel

To support parked slaves, the master establishes a beacon channel when one or more slaves are parked. The beacon channel consists of one beacon slot or a train of equidistant beacon slots which is transmitted periodically with a constant time interval. The beacon channel is illustrated in Figure 10.8 on page 117. A train of N_B ($N_B \geq 1$) beacon slots is defined with an interval of T_B slots.

The beacon slots in the train are separated by Δ_B. The start of the first beacon slot is referred to as the **beacon instant** and serves as the beacon timing reference. The beacon parameters N_B and T_B are chosen such that there are sufficient beacon slots for a parked slave to synchronize to during a certain time window in an error-prone environment.

When parked, the slave will receive the beacon parameters through an LMP command. In addition, the timing of the beacon instant is indicated through the offset D_B. Like for the SCO link (see Section 3.2 on page 45), two initialization procedures 1 or 2 are used. The master uses initialization 1 when the MSB of the current master clock (CLK_{27}) is 0; it uses initialization 2 when the MSB of the current master clock (CLK_{27}) is 1. The chosen initialization procedure is also carried by an initialization flag in the LMP command. The slave shall apply the initiations method as indicated by the initialization flag irrespective of its clock bit CLK_{27}. The master-to-slave slot positioned at the beacon instant shall be initialized on the slots for which the clock satisfies the following equation

$$CLK_{27-1} \bmod T_B = D_B \qquad \text{for initialization 1}$$

$$(\overline{CLK}_{27}, CLK_{26-1}) \bmod T_B = D_B \qquad \text{for initialization 2}$$

After initialization, the clock value CLK(k+1) for the next beacon instant is found by adding the fixed interval T_B to the clock value of the current beacon instant:

$$CLK(k+1) = CLK(k) + T_B$$

The beacon channel serves four purposes:

1. transmission of master-to-slave packets which the parked slaves can use for re-synchronization
2. carrying messages to the parked slaves to change the beacon parameters
3. carrying general broadcast messages to the parked slaves
4. unparking of one or more parked slaves

Since a slave can synchronize to any packet which is preceded by the proper channel access code, the packets carried on the beacon slots do not have to contain specific broadcast packets for parked slaves to be able to synchronize; any packet can be used. The only requirement placed on the beacon slots is that there is master-to-slave transmission present. If there is no information to be sent, **NULL** packets can be transmitted by the master. If there is indeed broadcast information to be sent to the parked slaves, the first packet of the broadcast message shall be repeated in every beacon slot of the beacon train. However, synchronous traffic like on the SCO link, may interrupt the beacon transmission.

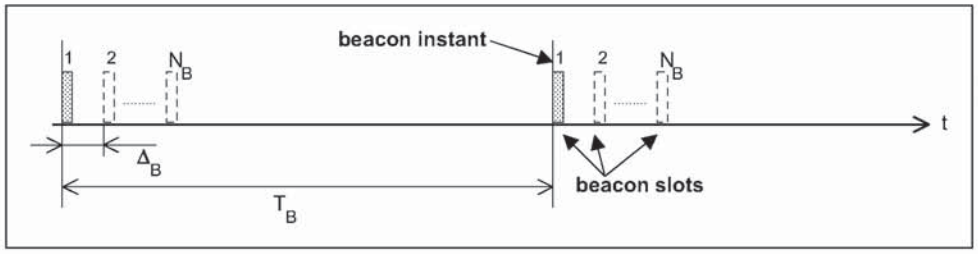

Figure 10.8: General beacon channel format

10.8.4.2 Beacon access window

In addition to the beacon slots, an access window is defined where the parked slaves can send requests to be unparked. To increase reliability, the access window can be repeated M_{access} times ($M_{access} \geq 1$), see Figure 10.9 on page 117. The access window starts a fixed delay D_{access} after the beacon instant. The width of the access window is T_{access}.

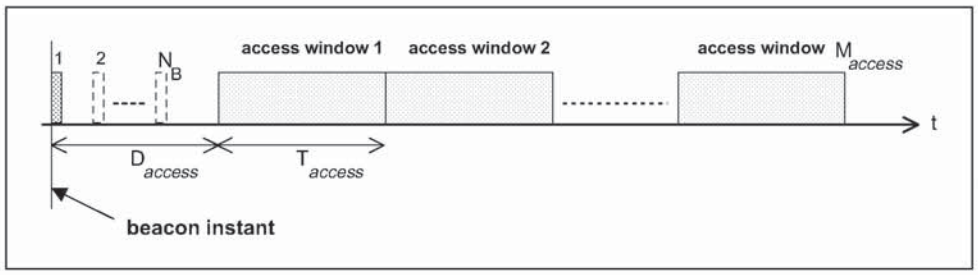

Figure 10.9: Definition of access window

The access window may support different slave access techniques, like polling, random access, or other forms of access. At this stage, only the polling technique has been defined. The format of the polling technique is shown in Figure 10.10 on page 118. The same TDD structure is used as on the piconet channel, i.e. master-to-slave transmission is alternated by slave-to-master transmission. The slave-to-master slot is divided into two half slots of 312.5 µs each. The half slot a parked slave is allowed to respond in corresponds to its access request address (AR_ADDR), see also section 10.8.4.6 on page 120. For counting the half slots to determine the access request slot, the start of the access window is used, see Figure 10.10 on page 118. The slave is only allowed to send an access request in the proper slave-to-master half slot if in the preceding master-to-slave slot a broadcast packet has been received. In this way, the master polls the parked slaves.

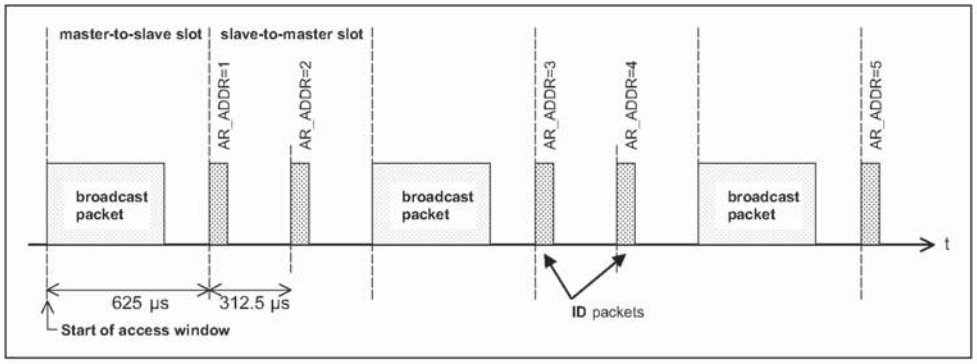

Figure 10.10: Access procedure applying the polling technique.

However, the slots of the access window can also be used for traffic on the piconet if required. For example, if an SCO connection has to be supported, the slots reserved for the SCO link may carry SCO information instead of being used for access requests, i.e. if the master-to-slave slot in the access window contains a packet different from a broadcast packet, the following slave-to-master slot cannot be used for slave access requests. Slots in the access window not affected by traffic can still be used according to the defined access structure; an example is shown in Figure 10.11 on page 118: the access procedure is continued as if no interruption had taken place.

When the slave is parked, it is indicated what type of access scheme will be used. For the polling scheme, the number of slave-to-master access slots N_{acc_slot} is indicated.

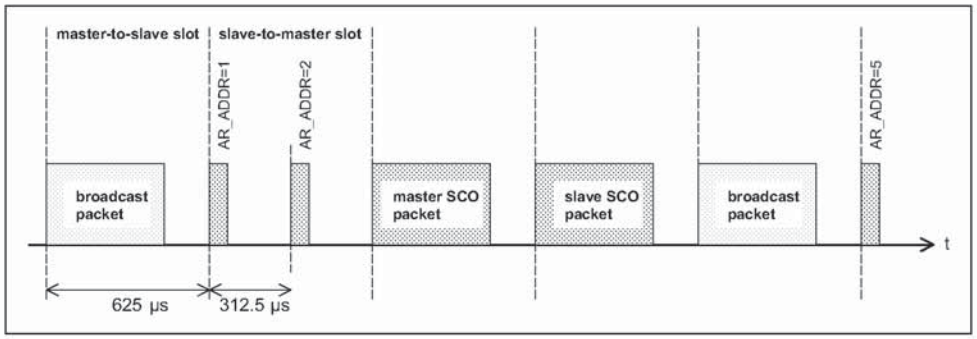

Figure 10.11: Disturbance of access window by SCO traffic

By default, the access window is always present. However, its activation depends on the master sending broadcast messages to the slave at the appropriate slots in the access window. A broadcast LMP command in the beacon slots may indicate that the access window following will not be activated. This prevents unnecessary scanning of parked slaves that want to request access.

10.8.4.3 Parked slave synchronization

Parked slaves sleep most of the time. However, periodically they wake up to re-synchronize to the channel. Any packet exchanged on the channel can be used for synchronization. Since master transmission is mandatory on the beacon slots, parked slaves will exploit the beacon channel to re-synchronize. A parked slave will wake-up at the beacon instant to read the packet sent on the first beacon slot. If this fails, it will retry on the next beacon slot in the beacon train; in total, there are N_B opportunities per beacon instant to re-synchronize. During the search, the slave may increase its search window, see also Section 9.4 on page 90. The separation between the beacon slots in the beacon train Δ_B is chosen such that consecutive search windows will not overlap.

The parked slave does not have to wake up at every beacon instant. Instead, a sleep interval can be applied which is longer than the beacon interval T_B, see Figure 10.12 on page 119. The slave sleep window must be a multiple N_{B_sleep} of T_B. The precise beacon instant the slave shall wake up on is indicated by the master with D_{B_sleep} which indicates the offset (in multiples of T_B) with respect to the beacon instant ($0 < D_{B\_sleep} < N_{B\_sleep}-1$). To initialize the wake-up period, the following equations are used:

$$CLK_{27-1} \bmod (N_{B\_sleep} \bullet T_B) = D_B + D_{B\_sleep} \bullet T_B \quad \text{for initialization 1}$$

$$(\overline{CLK_{27}}, CLK_{26-1}) \bmod (N_{B\_sleep} \bullet T_B) = D_B + D_{B\_sleep} \bullet T_B \quad \text{for initialization 2}$$

where initialization 1 is chosen by the master if the MSB in the current master clock is 0 and initialization 2 is chosen if the MSB in the current master clock is 1.

When the master wants to send broadcast messages to the parked slaves, it may use the beacon slots for these broadcast messages. However, if $N_B < N_{BC}$, the slots following the last beacon slot in the beacon train shall be used for the remaining $N_{BC}-N_B$ broadcast packets. If $N_B > N_{BC}$, the broadcast message is repeated on all N_B beacon slots.

A parked slave shall at least read the broadcast messages sent in the beacon slot(s) it wakes up in; the minimum wake-up activity is to read the channel access code for re-synchronization and the packet header to check for broadcast messages.

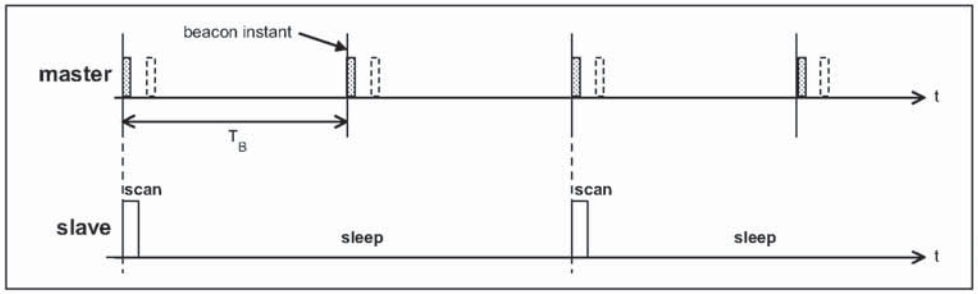

Figure 10.12: Extended sleep interval of parked slaves.

10.8.4.4 Parking

A master can park an active slave through the exchange of one or a few LMP commands. Before put into the park mode, the slave is assigned a PM_ADDR and an AR_ADDR. Every parked slave has a unique PM_ADDR; however, the AR_ADDR is not necessarily unique. Also, the beacon parameters are given by the master when the slave is parked. The slave then gives up its AM_ADDR and enters the park mode. A master can park only a single slave at a time. The park message is carried with a normal data packet and addresses the slave through its AM_ADDR.

10.8.4.5 Master-activated unparking

The master can unpark a parked slave by sending a dedicated LMP unpark command including the parked slave's address. This message is sent in a broadcast packet on the beacon slots. Either the slave's PM_ADDR is used, or its full BD_ADDR is used. The message also includes the active member address AM_ADDR the slave will use after it has re-entered the piconet. The unpark message can include a number of slave addresses so that multiple slaves can be unparked simultaneously. For each slave, a different AM_ADDR is assigned.

After having received the unpark message, the parked slave matching the PM_ADDR or BD_ADDR will leave the park mode and enter the active mode. It will keep listening to the master until it is addressed by the master through its AM_ADDR. The first packet sent by the master should be a POLL packet. The return packet in response to the POLL packet confirms that the slave has been unparked. If no response packets from the slave is received for *newconnectionTO* number of slots after the end of beacon repetition period, the master will unpark the slave again. If the slave does not receive the POLL packet for *newconnectionTO* number of slots after the end of beacon repetition period, it will return to park, with the same beacon parameters. After confirming that the slave is active, the master decides in which mode the slave will continue.

10.8.4.6 Slave-activated unparking

A slave can request access to the channel through the access window defined in section 10.8.4.2 on page 117. As shown in Figure 10.10 on page 118, the access window includes several slave-to-master half slots where the slave can send an access request message. The specific half slot the slave is allowed to respond in, corresponds to its access request address (AR_ADDR) which it has received when it was parked. The order of the half slots (in Figure 10.10 the AR_ADDR numbers linearly increase from 1 to 5) is not fixed: an LMP command sent in the beacon slots may reconfigure the access window. When a slave desires access to the channel, it sends an access request message in the proper slave-to-master half slot.The access request message of the slave is the **ID** packet containing the device access code (DAC) of the master (which is in this case the channel access code without the trailer). The parked slave is

only allowed to transmit an access request message in the half slot when in the preceding master-to-slave slot, a broadcast packet has been received. This broadcast message can contain any kind of broadcast information not necessarily related to the parked slave(s). If no broadcast information is available, a broadcast **NULL** or broadcast **POLL** packet shall be sent.

After having sent an access request, the parked slave will listen for an unpark message from the master. As long as no unpark message is received, the slave will repeat the access requests in the subsequent access windows. After the last access window (there are M_{access} windows in total, see Section 10.8.4.2 on page 117), the parked slave shall listen for an additional N_{poll} time slots for an unpark message. If no unpark message is received within N_{poll} slots after the end of the last access window, the slave may return to sleep and retry an access attempt after the next beacon instant.

After having received the unpark message, the parked slave matching the PM_ADDR or BD_ADDR will leave the park mode and enter the active mode. It will keep listening to the master until it is addressed by the master through its AM_ADDR. The first packet sent by the master should be a POLL packet. The return packet in response to the POLL packet confirms that the slave has been unparked. If no response packet from the slave is received for *newconnectionTO* number of slots after N_{poll} slots after the end of the last access window, the master will send the unpark message to the slave again. If the slave does not receive the POLL packet for *newconnectionTO* number of slots after N_{poll} slots after the end of the last access window, it will return to park, with the same beacon parameters. After confirming that the slave is active, the master decides in which mode the slave will continue.

10.8.4.7 Broadcast scan window

In the beacon train, the master can support broadcast messages to the parked slaves. However, it may extend its broadcast capacity by indicating to the parked slaves that more broadcast information is following after the beacon train. This is achieved by a special LMP command ordering the parked slaves (as well as the active slaves) to listen to the channel for broadcast messages during a limited time window. This time window starts at the beacon instant and continues for the period as indicated in the LMP command sent in the beacon train.

10.8.5 Polling schemes

10.8.5.1 Polling in active mode

The master always has full control over the piconet. Due to the stringent TDD scheme, slaves can only communicate with the master and not to other slaves. In order to avoid collisions on the ACL link, a slave is only allowed to transmit in the slave-to-master slot when addressed by the AM_ADDR in the packet

header in the preceding master-to-slave slot. If the AM_ADDR in the preceding slot does not match, or an AM_ADDR cannot be derived from the preceding slot, the slave is not allowed to transmit.

On the SCO links, the polling rule is slightly modified. The slave is allowed to transmit in the slot reserved for his SCO link unless the (valid) AM_ADDR in the preceding slot indicates a different slave. If no valid AM_ADDR can be derived in the preceding slot, the slave is still allowed to transmit in the reserved SCO slot.

10.8.5.2 Polling in park mode

In the park mode, parked slaves are allowed to send access requests in the access window provided a broadcast packet is received in the preceding master-to-slave slot. Slaves in active mode will not send in the slave-to-master slots following the broadcast packet since they are only allowed to send if addressed specifically.

10.8.6 Slot reservation scheme

The SCO link is established by negotiations between the link managers which involves the exchange of important SCO timing parameters like T_{SCO} and D_{SCO} through LMP messages.

10.8.7 Broadcast scheme

The master of the piconet can broadcast messages which will reach all slaves. A broadcast packet is characterized by the all-zero AM_ADDR. Each new broadcast message (which may be carried by a number of packets) shall start with the flush indication (L_CH=10).

A broadcast packet is never acknowledged. In an error-prone environment, the master may carry out a number of retransmissions N_{BC} to increase the probability for error-free delivery, see also Section 5.3.5 on page 72.

In order to support the **park** mode (as described in Section 10.8.4 on page 115), a master transmission shall take place at fixed intervals. This master transmission will act as a beacon to which slaves can synchronize. If no traffic takes place at the beacon event, broadcast packets shall be sent. More information is given in Section 10.8.4 on page 115.

10.9 SCATTERNET

10.9.1 General

Multiple piconets may cover the same area. Since each piconet has a different master, the piconets hop independently, each with their own channel hopping

sequence and phase as determined by the respective master. In addition, the packets carried on the channels are preceded by different channel access codes as determined by the master device addresses. As more piconets are added, the probability of collisions increases; a graceful degradation of performance results as is common in frequency-hopping spread spectrum systems.

If multiple piconets cover the same area, a unit can participate in two or more overlaying piconets by applying time multiplexing. To participate on the proper channel, it should use the associated master device address and proper clock offset to obtain the correct phase. A Bluetooth unit can act as a slave in several piconets, but only as a master in a single piconet: since two piconets with the same master are synchronized and use the same hopping sequence, they are one and the same piconet. A group of piconets in which connections consists between different piconets is called a **scatternet**.

A master or slave can become a slave in another piconet by being paged by the master of this other piconet. On the other hand, a unit participating in one piconet can page the master or slave of another piconet. Since the paging unit always starts out as master, a master-slave role exchange is required if a slave role is desired. This is described in the section 10.9.3 on page 123.

10.9.2 Inter-piconet communications

Time multiplexing must be used to switch between piconets. In case of ACL links only, a unit can request to enter the **hold** or **park** mode in the current piconet during which time it may join another piconet by just changing the channel parameters. Units in the **sniff** mode may have sufficient time to visit another piconet in between the sniff slots. If SCO links are established, other piconets can only be visited in the non-reserved slots in between. This is only possible if there is a single SCO link using **HV3** packets. In the four slots in between, one other piconet can be visited. Since the multiple piconets are not synchronized, guard time must be left to account for misalignment. This means that only 2 slots can effectively be used to visit another piconet in between the **HV3** packets.

Since the clocks of two masters of different piconets are not synchronized, a slave unit participating in two piconets has to take care of two offsets that, added to its own native clock, create one or the other master clock. Since the two master clocks drift independently, regular updates of the offsets are required in order for the slave unit to keep synchronization to both masters.

10.9.3 Master-slave switch

In principle, the unit that creates the piconet is the master. However, a master-slave (MS) switch can take place when a slave wants to become a master. For the two units involved in the switch, the MS switch results in a reversal of their TX and RX timing: a TDD switch. However, since the piconet parameters are derived from the device address and clock of the master, a master-slave switch inherently involves a redefinition of the piconet as well: a piconet switch. The

new piconet's parameters are derived from the former slave's device address and clock. As a consequence of this piconet switch, other slaves in the piconet not involved in the switch have to be moved to the new piconet, changing their timing and their hopping scheme. The new piconet parameters have to be communicated to each slave. The scenario to achieve this is described below. Assume unit A wants to become master; unit B was the former master. The following steps are taken.

- Slave A and master B agree to exchange roles.
- When confirmed by both units, both slave A and master B do the TDD switch but keep the former hopping scheme (still using the device address and clock of unit B), so there is no piconet switch yet.
- Unit A is now the master of the piconet. Since the old and new masters' clocks are asynchronous, the 1.25 ms resolution of the clock information given in the FHS packet is not sufficient for aligning the slot boundaries of the two piconets. Prior to sending the FHS packet, the new master A sends an LMP packet giving the delay between the start of the master-to-slave slots of the old and new piconet channels. This timing information ranges from 0 to 1249 µs with a resolution of 1 µs. It is used together with the clock information in the FHS packet to accurately position the correlation window when switching to the new master's timing after acknowledgment of the FHS packet.
- After the time alignment LMP message, Master A sends an FHS packet including the new AM_ADDR to slave B (the AM_ADDR in the FHS packet header is the all-zero address) still using the "old" piconet parameters. After the FHS acknowledgement, which consists of the **ID** packet and is sent by the slave on the old hopping sequence, both master A and slave B turn to the new channel parameters of the new piconet as indicated by the FHS and time alignment LMP packets (at least for the A-B connection).
- A piconet switch is enforced on each slave separately. Master A sends a time alignment and an FHS packets and waits for an acknowledgement. Transmission of the FHS packet and the acknowledgement continues on the "old" piconet parameters of unit B (compare this to the page hopping scheme used during connection establishment, see Section 10.6.4 on page 104). After FHS acknowledgement using an **ID** packet sent by the slave, the communication to this slave continues with the new device address and clock of unit A. The FHS packet sent to each slave has the old AM_ADDR in the FHS packet header and their new AM_ADDR in the FHS packet payload (the new AM_ADDR may be identical to the old AM_ADDR).
- After reception of the FHS packet acknowledgement, the new master A switches to its own timing and sends a POLL packet to verify the switch. Both the master and the slave will start a timer with a time out of *newconnectionTO* on FHS packet acknowledgement. If no response is received, the master resends the POLL packet until *newconnectionTO* is reached. After this timeout both the slave and the master return to the old piconet timing (but the TDD switch remains).The master sends the FHS packet again and the procedure is repeated.

- The new master repeats the above procedure for each slave in the old piconet.

Summarized, the MS-switch takes place in two steps: first a TDD switch of the considered master and slave, and then a piconet switch of all participants. When all slaves have acknowledged the reception of the FHS packet, each unit uses the new piconet parameters defined by the new master and the piconet switch is a fact. The information on the AM_ADDR, PM_ADDR, and other features of the old slaves is transferred from the old master to the new master. The transfer procedure is outside the scope of this procedure. Parked slaves shall be activated (using the old park parameters), be changed to the new piconet parameters, and then return to the **park** mode using the new park parameters.

10.10 POWER MANAGEMENT

Features are included into Bluetooth to ensure a low-power operation. These features are both at the microscopic level when handling the packets, and at the macroscopic level using certain operation modes.

10.10.1 Packet handling

In order to minimize power consumption, packet handling is minimized both at TX and RX sides. At the TX side, power is minimized by only sending useful data. This means that if only link control information needs to be exchanged, **NULL** packets will be used. No transmission is carried out at all if there is no link control information or involves a NAK only (NAK is implicit on no reply). If there is data to be sent, the payload length is adapted in order to send only the valid data bytes. At the RX side, packet processing takes place in different steps. If no valid access code is found in the search window, the transceiver returns to sleep. If an access code is found, the receiver unit is woken up and starts to process the packet header. If the HEC fails, the unit will return to sleep after the packet header. A valid header will indicate if a payload will follow and how many slots are involved.

10.10.2 Slot occupancy

As was described in Section 4.4 on page 54, the packet type indicates how many slots a packet may occupy. A slave not addressed in the first slot can go to sleep for the remaining slots the packet may occupy. This can be read from the TYPE code.

10.10.3 Low-power modes

In Section 10.8 on page 112, three modes were described during the **CONNECTION** state which reduce power consumption. If we list the modes in increasing order of power efficiency then the **sniff** mode has the higher duty

cycle, followed by the **hold** mode with a lower duty cycle, and finishing with the **park** mode with the lowest duty cycle.

10.11 LINK SUPERVISION

A connection may break down due to various reasons such as a device moving out of range or a power failure condition. Since this may happen without any prior warning, it is important to monitor the link on both the master and the slave side to avoid possible collisions when the AM_ADDR is reassigned to another slave.

To be able to supervise link loss, both the master and the slave use link supervision timers, $T_{supervision}$. Upon reception of a packet that passes the HEC check and has the correct AM_ADDR, the timer is reset. If at any time in connection state, the timer reaches the *supervisionTO* value, the connection is reset. The same timeout value is used for both SCO and ACL connections.

The timeout period, *supervisionTO*, is negotiated at the LM level. Its value is chosen so that the supervision timeout will be longer than hold and sniff periods. Link supervision of a parked slave will be done by unparking and re-parking the slave.

11 HOP SELECTION

In total, 10 types of hopping sequences are defined – five for the 79-hop and five for the 23-hop system, respectively. Using the notation of parentheses () for figures related to the 23-hop system, these sequences are:

- A **page hopping sequence** with 32 (16) unique wake-up frequencies distributed equally over the 79 (23) MHz, with a period length of 32 (16);
- A **page response sequence** covering 32 (16) unique response frequencies that all are in an one-to-one correspondence to the current page hopping sequence. The master and slave use different rules to obtain the same sequence;
- An **inquiry sequence** with 32 (16) unique wake-up frequencies distributed equally over the 79 (23) MHz, with a period length of 32 (16);
- A **inquiry response sequence** covering 32 (16) unique response frequencies that all are in an one-to-one correspondence to the current inquiry hopping sequence.
- A **channel hopping sequence** which has a very long period length, which does not show repetitive patterns over a short time interval, but which distributes the hop frequencies equally over the 79 (23) MHz during a short time interval;

For the page hopping sequence, it is important that we can easily shift the phase forward or backward, so we need a 1-1 mapping from a counter to the hop frequencies. For each case, both a hop sequence from master to slave and from slave to master are required.

The inquiry and inquiry response sequences always utilizes the GIAC LAP as lower address part and the DCI (Section 5.4 on page 73) as upper address part in deriving the hopping sequence, even if it concerns a DIAC inquiry.

11.1 GENERAL SELECTION SCHEME

The selection scheme consists of two parts:

- selecting a sequence;
- mapping this sequence on the hop frequencies;

The general block diagram of the hop selection scheme is shown in Figure 11.1 on page 128. The mapping from the input to a particular hop frequency is performed in the selection box. Basically, the input is the native clock and the current address. In **CONNECTION** state, the native clock (CLKN) is modified by an offset to equal the master clock (CLK). Only the 27 MSBs of the clock are used. In the **page** and **inquiry** substates, all 28 bits of the clock are used. However, in **page** substate the native clock will be modified to the master's estimate of the paged unit.

The address input consists of 28 bits, i.e., the entire LAP and the 4 LSBs of the UAP. In **CONNECTION** state, the address of the master is used. In **page** substate the address of the paged unit is used. When in **inquiry** substate, the UAP/LAP corresponding to the GIAC is used. The output constitutes a pseudo-random sequence, either covering 79 hop or 23 hops, depending on the state.

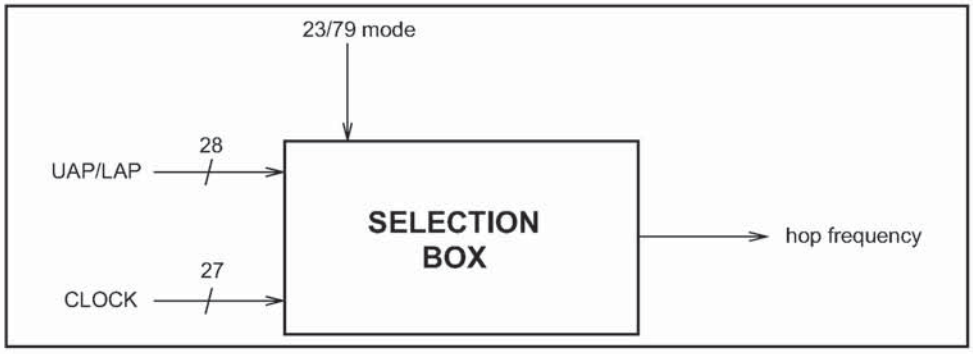

Figure 11.1: General block diagram of hop selection scheme.

For the 79-hop system, the selection scheme chooses a segment of 32 hop frequencies spanning about 64 MHz and visits these hops once in a random order. Next, a different 32-hop segment is chosen, etc. In case of the **page, page scan,** or **page response** substates, the same 32-hop segment is used all the time (the segment is selected by the address; different units will have different paging segments). In connection state, the output constitutes a pseudo-random sequence that slides through the 79 hops or 23 hops, depending on the selected hop system. For the 23-hop systems, the segment size is 16. The principle is depicted in Figure 11.2

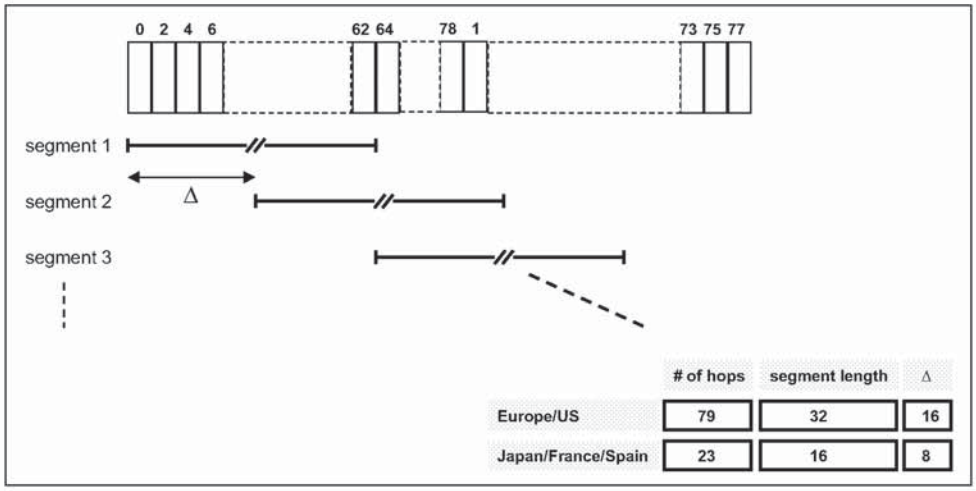

Figure 11.2: Hop selection scheme in CONNECTION state.

11.2 SELECTION KERNEL

The hop selection kernels for the 79 hop system and the 23 hop system are shown in Figure 11.3 on page 129 and Figure 11.4 on page 129, respectively. The X input determines the phase in the 32-hop segment, whereas Y1 and Y2 selects between master-to-slave and slave-to-master transmission. The inputs A to D determine the ordering within the segment, the inputs E and F determine the mapping onto the hop frequencies. The kernel addresses a register containing the hop frequencies. This list should be created such that first all even hop frequencies are listed and then all odd hop frequencies. In this way, a 32-hop segment spans about 64 MHz, whereas a 16-hop segment spans the entire 23-MHz.

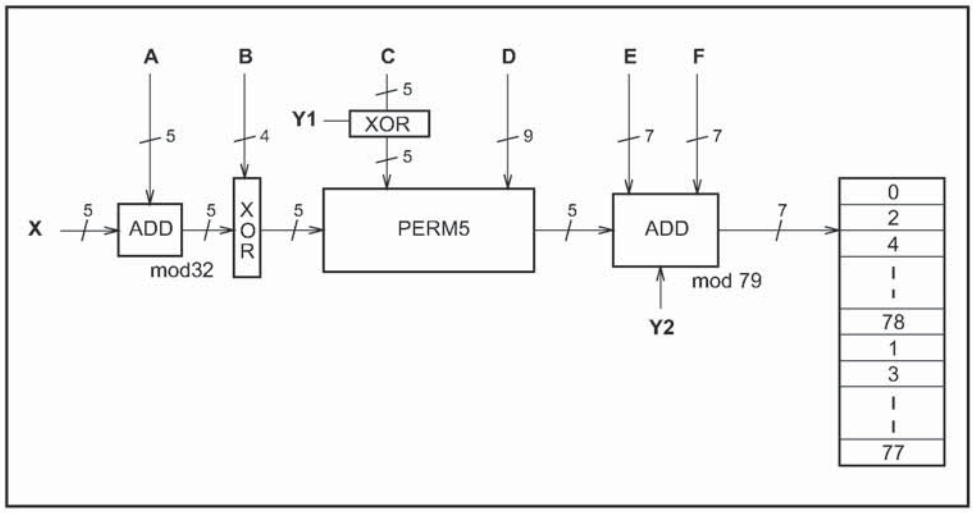

Figure 11.3: Block diagram of hop selection kernel for the 79-hop system.

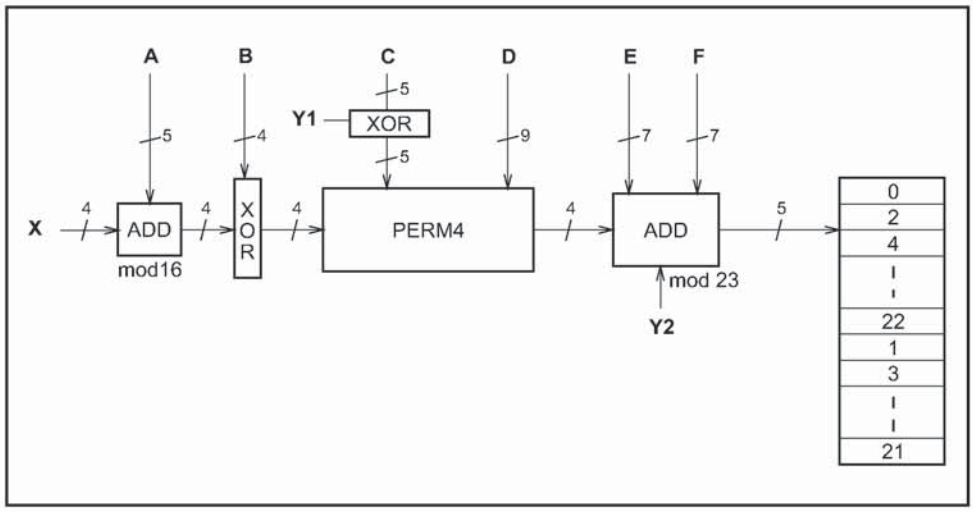

Figure 11.4: Block diagram of hop selection kernel for the 23-hop system.

The selection procedure consists of an addition, an XOR operation, a permutation operation, an addition, and finally a register selection. In the remainder of this chapter, the notation A_i is used for bit i of the BD_ADDR.

11.2.1 First addition operation

The first addition operation only adds a constant to the phase and applies a modulo 32 or a modulo 16 operation. For the page hopping sequence, the first addition is redundant since it only changes the phase within the segment. However, when different segments are concatenated (as in the channel hopping sequence), the first addition operation will have an impact on the resulting sequence.

11.2.2 XOR operation

Let Z' denote the output of the first addition. In the XOR operation, the four LSBs of Z' are modulo-2 added to the address bits $A_{22\text{-}19}$. The operation is illustrated in Figure 11.5 on page 130.

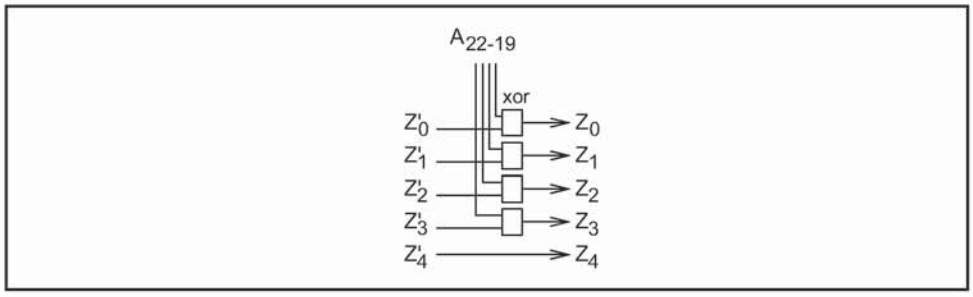

Figure 11.5: XOR operation for the 79-hop system. The 23-hop system is the same except for the Z'_4/Z_4 wire that does not exist.

11.2.3 Permutation operation

The permutation operation involves the switching from 5 inputs to 5 outputs for the 79 hop system and from 4 inputs to 4 outputs for 23 hop system, in a manner controlled by the control word. In Figure 11.6 on page 132 and Figure 11.7 on page 132 the permutation or switching box is shown. It consists of 7 stages of butterfly operations. Table 11.1 and Table 11.2 shows the control of the butterflies by the control signals P. Note that $P_{0\text{-}8}$ corresponds to $D_{0\text{-}8}$, and, P_{i+9} corresponds to $C_i \oplus Y1$ for $i = 0 \ldots 4$ in Figure 11.3 and Figure 11.4.

Control signal	Butterfly		Control signal	Butterfly
P_0	$\{Z_0,Z_1\}$		P_8	{Z1,Z4}
P_1	$\{Z_2,Z_3\}$		P_9	$\{Z_0,Z_3\}$
P_2	$\{Z_1,Z_2\}$		P_{10}	$\{Z_2,Z_4\}$
P_3	$\{Z_3,Z_4\}$		P_{11}	$\{Z_1,Z_3\}$
P_4	$\{Z_0,Z_4\}$		P_{12}	$\{Z_0,Z_3\}$
P_5	$\{Z_1,Z_3\}$		P_{13}	$\{Z_1,Z_2\}$
P_6	$\{Z_0,Z_2\}$			
P_7	$\{Z_3,Z_4\}$			

Table 11.1: Control of the butterflies for the 79 hop system

Control signal	Butterfly		Control signal	Butterfly
P_0	$\{Z_0,Z_1\}$		P_8	$\{Z_0,Z_2\}$
P_1	$\{Z_2,Z_3\}$		P_9	$\{Z_1,Z_3\}$
P_2	$\{Z_0,Z_3\}$		P_{10}	$\{Z_0,Z_3\}$
P_3	$\{Z_1,Z_2\}$		P_{11}	$\{Z_1,Z_2\}$
P_4	$\{Z_0,Z_2\}$		P_{12}	$\{Z_0,Z_1\}$
P_5	$\{Z_1,Z_3\}$		P_{13}	$\{Z_2,Z_3\}$
P_6	$\{Z_0,Z_1\}$			
P_7	$\{Z_2,Z_3\}$			

Table 11.2: Control of the butterflies for the 23 hop system

The Z input is the output of the XOR operation as described in the previous section. The butterfly operation can be implemented with multiplexers as depicted in Figure 11.8 on page 132.

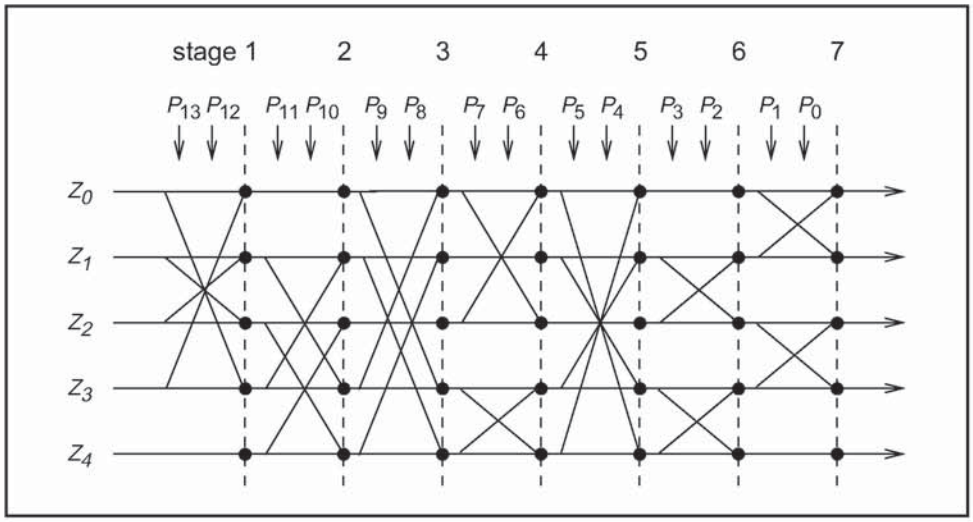

Figure 11.6: Permutation operation for the 79 hop system.

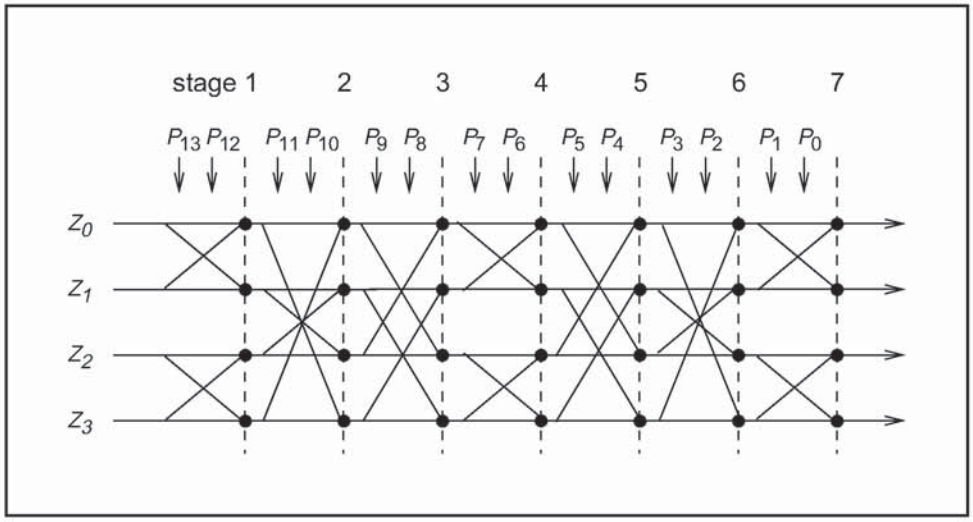

Figure 11.7: Permutation operation for the 23 hop system.

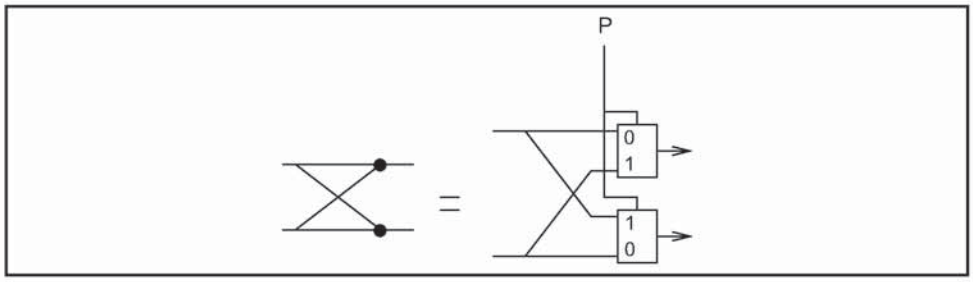

Figure 11.8: Butterfly implementation.

11.2.4 Second addition operation

The addition operation only adds a constant to the output of the permutation operation. As a result, the 16-hop or 32-hop segment is mapped differently on the hop frequencies. The addition is applied modulo 79 or modulo 23 depending on the system type (Europe/US vs. others).

11.2.5 Register bank

The output of the adder addresses a bank of 79 or 23 registers. The registers are loaded with the synthesizer code words corresponding to the hop frequencies 0 to 78 or 0 to 22. Note that the upper half of the bank contains the even hop frequencies, whereas the lower half of the bank contains the odd hop frequencies.

11.3 CONTROL WORD

In the following section X_{j-i}, $i<j$, will denote bits i, i+1,...,j of the bit vector X. By convention, X_0 is the least significant bit of the vector X.

The control word P of the kernel is controlled by the overall control signals X, Y1, Y2, and A to F as illustrated in Figure 11.3 on page 129 and Figure 11.4 on page 129. During paging and inquiry, the inputs A to E use the address values as given in the corresponding columns of Table 11.3 on page 134 and Table 11.4 on page 134. In addition, the inputs X, Y1 and Y2 are used. The F input is unused. In the 79-hop system, the clock bits CLK_{6-2} (i.e., input X) specifies the phase within the length 32 sequence, while for the 23-hop system, CLK_{5-2} specifies the phase within the length 16 sequence. For both systems, CLK_1 (i.e., inputs Y1 and Y2) is used to select between TX and RX. The address inputs determine the sequence order within segments. The final mapping onto the hop frequencies is determined by the register contents.

In the following we will distinguish between three types of clocks: the piconet's master clock, the Bluetooth unit's native clock, and the clock estimate of a paged Bluetooth unit. These types are marked in the following way:

1. CLK_{27-0}: Master clock of the current piconet.
2. $CLKN_{27-0}$: Native clock of the unit.
3. $CLKE_{27-0}$: The paging unit's estimate of the paged unit's native clock.

During the **CONNECTION** state, the inputs A, C and D result from the address bits being bit-wise XORed with the clock bits as shown in the "Connection state" column of Table 11.3 on page 134 and Table 11.4 on page 134 (the two MSBs are XORed together, the two second MSBs are XORed together, etc.). Consequently, after every 32 (16) time slots, a new length 32 (16) segment is selected in the 79-hop (23-hop) system. The sequence order within a specific

segment will not be repeated for a very long period. Thus, the overall hopping sequence consists of concatenated segments of 32-hops each. Since each 32-hop sequence spans more than 80% of the 79 MHz band, the desired frequency spreading over a short time interval is obtained.

	Page scan/ Inquiry scan	Page/Inquiry	Page response (master/slave) and Inquiry response	Connection state
X	$CLKN_{16-12}$	$Xp^{(79)}_{4-0}/Xi^{(79)}_{4-0}$	$Xprm^{(79)}_{4-0}/Xprs^{(79)}_{4-0}$	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	$16 \times CLK_{27-7} \bmod 79$

Table 11.3: Control for 79-hop system.

	Page scan/ Inquiry scan	Page/Inquiry	Page response (master/slave) and Inquiry response	Connection state
X	$CLKN_{15-12}$	$Xp^{(23)}_{3-0}/Xi^{(23)}_{3-0}$	$Xprm^{(23)}_{3-0}/Xprs^{(23)}_{3-0}$	CLK_{5-2}
Y1	0	$CLKE_1/CLKN_1$	$CLKE_1/CLKN_1$	CLK_1
Y2	0	$16 \times CLKE_1/$ $16 \times CLKN_1$	$16 \times CLKE_1/$ $16 \times CLKN_1$	$16 \times CLK_1$
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	$8 \times CLK_{27-6} \bmod 23$

Table 11.4: Control for 23-hop system.

11.3.1 Page scan and Inquiry scan substates

In **page scan**, the Bluetooth device address of the scanning unit is used as address input. In **inquiry scan**, the GIAC LAP and the four LSBs of the DCI (as A_{27-24}), are used as address input for the hopping sequence. Naturally, for the transmitted access code and in the receiver correlator, the appropriate GIAC or DIAC is used. The application decides which inquiry access code to use depending on the purpose of the inquiry.

The five X input bits vary depending on the current state of the unit. In the **page scan** and **inquiry scan** substates, the native clock (CLKN) is used. In **CONNECTION** state the master clock (CLK) is used as input. The situation is somewhat more complicated for the other states.

11.3.2 Page substate

In the **page** substate of the 79-hop system, the paging unit shall start using the **A**-train, i.e., $\{f(k-8), \ldots, f(k), \ldots, f(k+7)\}$, where $f(k)$ is the source's estimate of the current receiver frequency in the paged unit. Clearly, the index k is a function of all the inputs in Figure 11.3. There are 32 possible paging frequencies within each 1.28 second interval. Half of these frequencies belongs to the **A**-train, the rest (i.e., $\{f(k+8), \ldots, f(k+15), f(k-16), \ldots, f(k-9)\}$) belongs to the **B**-train. In order to achieve the -8 offset of the **A**-train, a constant of 24 can be added to the clock bits (which is equivalent to -8 due to the modulo 32 operation). Clearly, the **B**-train may be accomplished by setting the offset to 8. A cyclic shift of the order within the trains is also necessary in order to avoid a possible repetitive mismatch between the paging and scanning units. Thus,

$$Xp^{(79)} = [\mathrm{CLKE}_{16-12} + k_{offset} + (\mathrm{CLKE}_{4-2,0} - \mathrm{CLKE}_{16-12}) \mod 16] \mod 32, \quad \text{(EQ 2)}$$

where

$$k_{offset} = \begin{cases} 24 & \text{A-train,} \\ 8 & \text{B-train.} \end{cases} \quad \text{(EQ 3)}$$

Alternatively, each switch between the **A**- and **B**-trains may be accomplished by adding 16 to the current value of k_{offset} (originally initialized with 24).

In the **page** substate of the 23-hop system, the paging unit makes use of the **A**-train only. A constant offset of 8 is used in order to start with $f(k-8)$. Moreover, only four bits are needed since the additions are modulo 16. Consequently,

$$Xp^{(23)} = [\mathrm{CLKE}_{15-12} + 8 + \mathrm{CLKE}_{4-2,0}] \mod 16, \quad \text{(EQ 4)}$$

11.3.3 Page response

11.3.3.1 Slave response

A unit in the **page scan** substate recognizing its own access code enters the **slave response** substate. In order to eliminate the possibility of loosing the link due to discrepancies of the native clock CLKN and the master's clock estimate CLKE, the four bits CLKN_{16-12} must be frozen at their current value. The value is frozen to the content it has in the slot where the recipient's access code is detected. Note that the actual native clock is *not* stopped; it is merely the values of the bits used for creating the X-input that are kept fixed for a while. In the sequel, a frozen value is marked by an asterisk (*).

For each response slot the paged unit will use an X-input value one larger (modulo 32 or 16) than in the preceding response slot. However, the first response is made with the X-input kept at the same value as it was when the access code was recognized. Let N be a counter starting at zero. Then, the X-input in the $(N+1)$-th response slot (the first response slot being the one immediately following the page slot now responding to) of the **slave response** substate becomes

$$Xprs^{(79)} = [\mathrm{CLKN}^*_{16-12} + N] \mod 32, \qquad \text{(EQ 5)}$$

and

$$Xprs^{(23)} = [\mathrm{CLKN}^*_{15-12} + N] \mod 16, \qquad \text{(EQ 6)}$$

for the 79-hop and 23-hop systems, respectively. The counter N is set to zero in the slot where the slave acknowledges the page (see Figure 10.6 on page 105 and Figure 10.7 on page 105). Then, the value of N is increased by one each time CLKN_1 is set to zero, which corresponds to the start of a master TX slot. The X-input is constructed this way until the first accepted **FHS** packet is received *and* the immediately following response packet has been transmitted. After this the slave enters the **CONNECTION** state using the parameters received in the **FHS** packet.

11.3.3.2 Master response

The paging unit enters **master response** substate upon receiving a slave response. Clearly, also the master must freeze its estimated slave clock to the value that triggered a response from the paged unit. It is equivalent to using the values of the clock estimate when receiving the slave response (since only CLKE_1 will differ from the corresponding page transmission). Thus, the values are frozen when the slave **ID** packet is received. In addition to the used clock bits, also the current value of k_{offset} must be frozen. The master will adjust its X-input in the same way the paged unit does, i.e., by incrementing this value by

one for each time CLKE_1 is set to zero. The first increment shall be done before sending the **FHS** packet to the paged unit. Let N be a counter starting at one. The rules for forming the X-inputs become

$$Xprm^{(79)} = [\mathrm{CLKE}^*_{16-12} + k_{offset}^* + (\mathrm{CLKE}^*_{4-2,0} - \mathrm{CLKE}^*_{16-12}) \bmod 16 + N] \bmod 32, \quad \text{(EQ 7)}$$

and

$$Xprm^{(23)} = [\mathrm{CLKE}^*_{15-12} + 8 + \mathrm{CLKE}^*_{4-2,0} + N] \bmod 16, \quad \text{(EQ 8)}$$

for the 79-hop and 23-hop systems, respectively. The value of N is increased each time CLKE_1 is set to zero, which corresponds to the start of a master TX slot.

11.3.4 Inquiry substate

The X-input of the **inquiry** substate is quite similar to what is used in the **page** substate. Since no particular unit is addressed, the native clock CLKN of the inquirer is used. Moreover, which of the two train offsets to start with is of no real concern in this state. Consequently,

$$Xi^{(79)} = [\mathrm{CLKN}_{16-12} + k_{offset} + (\mathrm{CLKN}_{4-2,0} - \mathrm{CLKN}_{16-12}) \bmod 16] \bmod 32, \quad \text{(EQ 9)}$$

where k_{offset} is defined by (EQ 3). The initial choice of the offset is arbitrary. For the 23-hop system,

$$Xi^{(23)} = [\mathrm{CLKN}_{15-12} + 8 + \mathrm{CLKN}_{4-2,0}] \bmod 16, \quad \text{(EQ 10)}$$

The GIAC LAP and the four LSBs of the DCI (as A_{27-24}) are used as address input for the hopping sequence generator.

11.3.5 Inquiry response

The **inquiry response** substate is similar to the **slave response** with respect to the X-input. Thus, (EQ 5) and (EQ 6) holds. However, the counter N is increased not on CLKN_1 basis, but rather after each **FHS** packet has been transmitted in response to the inquiry.

The GIAC LAP and the four LSBs of the DCI (as A_{27-24}) are used as address input for the hopping sequence generator. The other input bits to the generator are the same as in the case of page response.

11.3.6 Connection state

In **CONNECTION** state, the clock bits to use in the channel hopping sequence generation are always according to the master clock, CLK. The address bits are taken from the Bluetooth device address of the master.

12 BLUETOOTH AUDIO

On the Bluetooth air-interface, either a 64 kb/s log PCM format (A-law or μ-law) is used, or a 64 kb/s CVSD (Continuous Variable Slope Delta Modulation) is used. The latter format applies an adaptive delta modulation algorithm with syllabic companding.

The voice coding on the line interface should have a quality equal to or better than the quality of 64 kb/s log PCM.

Table 12.1 on page 139 summarizes the voice coding schemes supported on the air interface. The appropriate voice coding scheme is selected after negotiations between the Link Managers.

Voice Codecs	
linear	CVSD
8-bit logarithmic	A-law
	μ-law

Table 12.1: Voice coding schemes supported on the air interface.

12.1 LOG PCM CODEC

Since the voice channels on the air-interface can support a 64 kb/s information stream, a 64 kb/s log PCM traffic can be used for transmission. Either A-law or μ-law compression can be applied. In the event that the line interface uses A-law and the air interface uses μ-law or vice versa, a conversion from A-law to μ-law is performed. The compression method follows ITU-T recommendations G. 711.

12.2 CVSD CODEC

A more robust format for voice over the air interface is a delta modulation. This modulation scheme follows the waveform where the output bits indicate whether the prediction value is smaller or larger then the input waveform. To reduce slope overload effects, syllabic companding is applied: the step size is adapted according to the average signal slope. The input to the CVSD encoder is 64 ksamples/s linear PCM. Block diagrams of the CVSD encoder and CVSD decoder are shown in Figure 12.1 on page 140, Figure 12.2 on page 140 and Figure 12.3 on page 140. The system is clocked at 64 kHz.

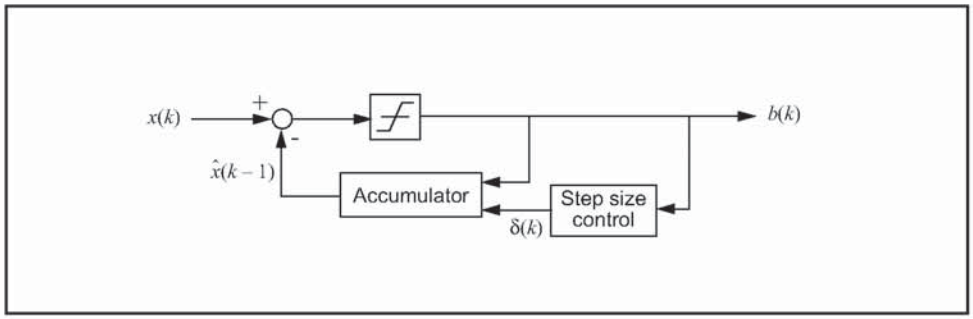

Figure 12.1: Block diagram of CVSD encoder with syllabic companding.

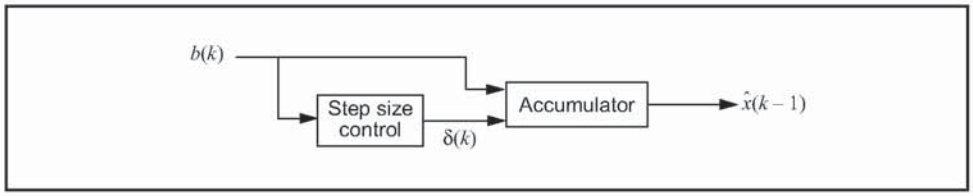

Figure 12.2: Block diagram of CVSD decoder with syllabic companding.

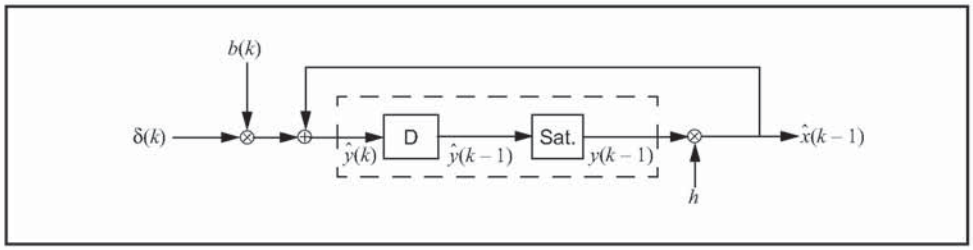

Figure 12.3: Accumulator procedure

Let $\mathrm{sgn}(x) = 1$ for $x \geq 0$, otherwise $\mathrm{sgn}(x) = -1$. On air these numbers are represented by the sign bit; i.e. negative numbers are mapped on "1" and positive numbers are mapped on "0". Denote the CVSD encoder output bit $b(k)$, the accumulator contents $y(k)$, and the step size $\delta(k)$. Furthermore, let *h* be the decay factor for the accumulator, let β denote the decay factor for the step size, and, let α be the syllabic companding parameter. The latter parameter monitors the slope by considering the *K* most recent output bits

Let

$$\hat{x}(k) = hy(k). \qquad \text{(EQ 11)}$$

Then, the CVSD encoder internal state is updated according to the following set of equations:

$$b(k) = \mathrm{sgn}\{x(k) - \hat{x}(k-1)\}, \qquad \text{(EQ 12)}$$

$$\alpha = \begin{cases} 1, & \text{if } J \text{ bits in the last } K \text{ output bits are equal,} \\ 0, & \text{otherwise,} \end{cases} \quad \text{(EQ 13)}$$

$$\delta(k) = \begin{cases} \min\{\delta(k-1) + \delta_{min}, \delta_{max}\}, & \alpha = 1, \\ \max\{\beta\delta(k-1), \delta_{min}\}, & \alpha = 0, \end{cases} \quad \text{(EQ 14)}$$

$$y(k) = \begin{cases} \min\{\hat{y}(k), y_{max}\}, & \hat{y}(k) \geq 0. \\ \max\{\hat{y}(k), y_{min}\}, & \hat{y}(k) < 0. \end{cases} \quad \text{(EQ 15)}$$

where

$$\hat{y}(k) = \hat{x}(k-1) + b(k)\delta(k). \quad \text{(EQ 16)}$$

In these equations, δ_{min} and δ_{max} are the minimum and maximum step sizes, and, y_{min} and y_{max} are the accumulator's negative and positive saturation values, respectively.

For a 64 kb/s CVSD, the parameters as shown in Table 12.2 must be used. The numbers are based on a 16 bit signed number output from the accumulator. These values result in a time constant of 0.5 ms for the accumulator decay, and a time constant of 16 ms for the step size decay

Parameter	Value
h	$1 - \frac{1}{32}$
β	$1 - \frac{1}{1024}$
J	4
K	4
δ_{min}	10
δ_{max}	1280
y_{min}	-2^{15} or $-2^{15}+1$
y_{max}	$2^{15}-1$

Table 12.2: CVSD parameter values. The values are based on a 16 bit signed number output from the accumulator.

12.3 ERROR HANDLING

In the **DV** and **HV3** packet, the voice is not protected by FEC. The quality of the voice in an error-prone environment then depends on the robustness of the voice coding scheme. CVSD, in particular, is rather insensitive to random bit errors, which are experienced as white background noise. However, when a packet is rejected because either the channel access code or the HEC test was unsuccessful, measures have to be taken to "fill" in the lost speech segment.

The voice payload in the **HV2** packet is protected by a 2/3 rate FEC. If errors occur which cannot be corrected, these should be ignored. That is, from the 15-bit FEC segment with uncorrected errors, the 10-bit information part as found before the FEC decoder should be used. The **HV1** packet is protected by a 3-repeat FEC. In the majority detection scheme uncorrected errors cannot occur.

12.4 GENERAL AUDIO REQUIREMENTS

These specifications are tentative and will be fixed within 18 months after the release of the Bluetooth Specification version 1.0 Draft Foundation.

12.4.1 Signal levels

For A-law and μ-law log-PCM encoded signals the requirements on signal levels follows ITU-T G.711.

Full swing at the 16 bit linear PCM interface to the CVSD encoder is defined to be 3 dBm0. A digital CVSD encoded test signal is provided in a Test Signal file available on the website. This signal is generated by a software implementation of a reference CVSD encoder. The digital encoder input signal (1020 Hz, sine-wave) generating the test signal has a nominal power of -15 dBm0. When the CVSD encoded test signal is fed through the CVSD receiver chain, the nominal output power should be -15 ± 1.0 dBm0.

12.4.2 CVSD audio quality

For Bluetooth audio quality the requirements are put on the transmitter side. The 64 ksamples/s linear PCM input signal must have negligible spectral power density above 4 kHz. A set of reference input signals are encoded by the transmitter and sent through a reference decoder (available on the website). The power spectral density in the 4-32 kHz band of the decoded signal at the 64 ksample/s linear PCM output, should be more than 20 dB below the maximum in the 0-4 kHz range.

13 BLUETOOTH ADDRESSING

13.1 BLUETOOTH DEVICE ADDRESS (BD_ADDR)

Each Bluetooth transceiver is allocated a unique 48-bit Bluetooth device address (BD_ADDR). This address is derived from the IEEE802 standard. This 48-bit address is divided into three fields:

- LAP field: lower address part consisting of 24 bits
- UAP field: upper address part consisting of 8 bits
- NAP field: non-significant address part consisting of 16 bits

The LAP and UAP form the significant part of the BD_ADDR. The total address space obtained is 2^{32}.

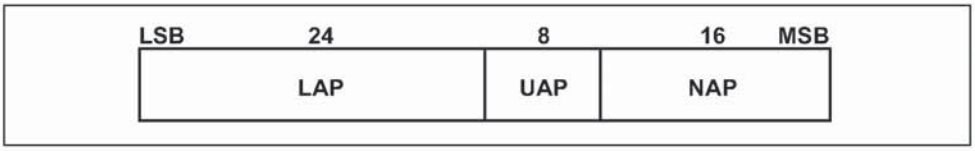

Figure 13.1: Format of BD_ADDR

13.2 ACCESS CODES

In the Bluetooth system, 72-bit and 68-bit access codes are used for signalling purposes. Three different access codes are defined, see also Section 4.2.1 on page 48:

- device access code (DAC)
- channel access code (CAC)
- inquiry access code (IAC)

There is one general IAC (GIAC) for general inquiry operations and there are 63 dedicated IACs (DIACs) for dedicated inquiry operations. All codes are derived from a LAP of the BD_ADDR. The device access code is used during page, page scan and page response substates. It is a code derived from the unit's BD_ADDR. The channel access code characterizes the channel of the piconet and forms the preamble of all packets exchanged on the channel. The channel access code is derived from the LAP of the master BD_ADDR. Finally, the inquiry access code is used in inquiry operations. A general inquiry access code is common to all Bluetooth units; a set of dedicated inquiry access codes is used to inquire for classes of devices.

The access code is also used to indicate to the receiver the arrival of a packet. It is used for timing synchronization and offset compensation. The receiver correlates against the entire sync word in the access code, providing a very robust signalling. During channel setup, the code itself is used as an ID packet to sup-

port the acquisition process. In addition, it is used during random access procedures in the PARK state.

The access code consists of preamble, sync word and a trailer, see Section 4.2 on page 48. The next two sections describe the generation of the sync word.

13.2.1 Synchronization word definition

The sync words are based on a (64,30) expurgated block code with an overlay (bit-wise XOR) of an 64 bit full length PN-sequence. The expurgated code guarantees large Hamming distance ($d_{min} = 14$) between sync words based on different addresses. The PN sequence improves the autocorrelation properties of the access code. The following steps describe how to generate the sync word:

1. Generate information sequence;
2. XOR this with the "information covering" part of the PN overlay sequence;
3. Generate the codeword;
4. XOR the codeword with all 64 bits of the PN overlay sequence;

The information sequence is generated by appending 6 bits to the 24 bit LAP (step 1). The appended bits are 001101 if the MSB of the LAP equals 0. If the MSB of the LAP is 1 the appended bits are 110010. The LAP MSB together with the appended bits constitute a length-seven Barker sequence. The purpose of including a Barker sequence is to further improve the autocorrelation properties. In step 2 the information is pre-scrambled by XORing it with the bits $p_{34} \ldots p_{63}$ of the *pseudo-random noise* (PN) sequence (defined in section 13.2.2 on page 146). After generating the codeword (step 3), the complete PN sequence is XORed to the codeword (step 4). This step de-scrambles the information part of the codeword. At the same time the parity bits of the codeword are scrambled. Consequently, the original LAP and Barker sequence are ensured a role as a part of the access code sync word, and the cyclic properties of the underlying code is removed. The principle is depicted in Figure 13.2 on page 145

In the sequel, binary sequences will be denoted by their corresponding D-transform (in which D^i represents a delay of i time units). Let $p'(D) = p'_0 + p'_1 D + \ldots + p'_{62} D^{62}$ be the 63 bit pseudo-random sequence, where p'_0 is the first bit (LSB) leaving the PRNG (see Figure 13.3 on page 147), and, p'_{62} is the last bit (MSB). To obtain 64 bits, an extra zero is appended at the *end* of this sequence (thus, $p'(D)$ is unchanged). For notational convenience, the reciprocal of this extended polynomial, $p(D) = D^{63} p'(1/D)$, will be used in the sequel. This is the sequence $p'(D)$ in reverse order. We denote the 24 bit lower

address part (LAP) of the Bluetooth address by $a(D) = a_0 + a_1 D + \ldots + a_{23} D^{23}$ (a_0 is the LSB of the Bluetooth address).

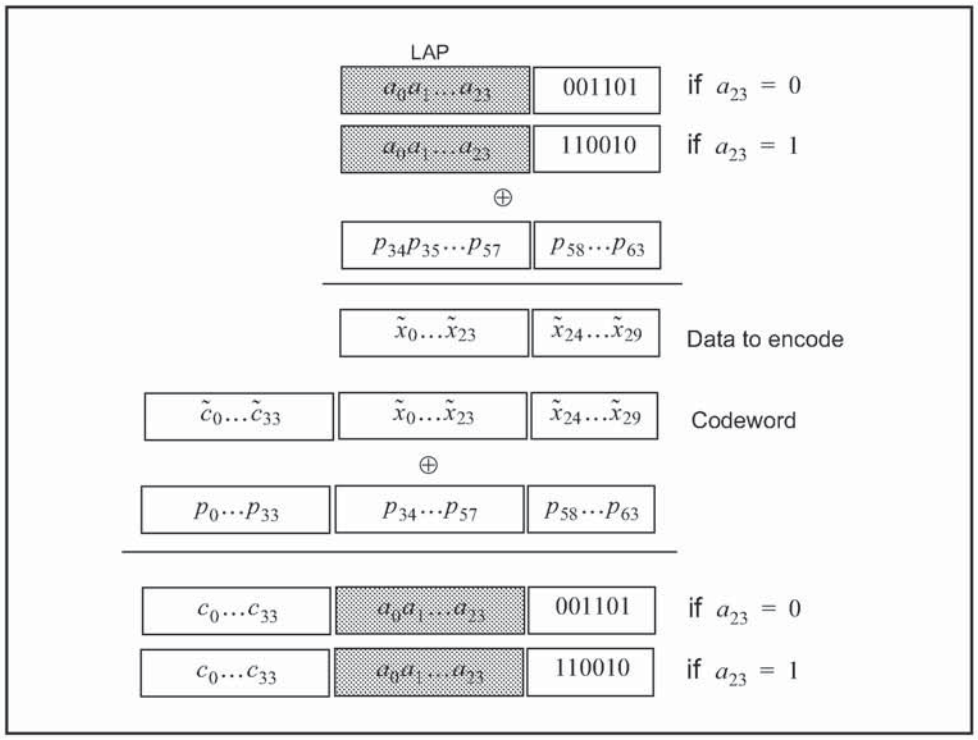

Figure 13.2: Construction of the sync word.

The (64,30) block code generator polynomial is denoted $g(D) = (1 + D)g'(D)$, where $g'(D)$ is the generator polynomial `157464165547` (octal notation) of a primitive binary (63,30) BCH code. Thus, in octal notation we have

$$g(D) = 260534236651, \quad \text{(EQ 17)}$$

the left-most bit corresponds to the high-order (g_{34}) coefficient.The DC-free four bit sequences `0101` and `1010` can be written

$$\begin{cases} F_0(D) = D + D^3, \\ F_1(D) = 1 + D^2, \end{cases} \quad \text{(EQ 18)}$$

respectively. Furthermore, we define

$$\begin{cases} B_0(D) = D^2 + D^3 + D^5, \\ B_1(D) = 1 + D + D^4, \end{cases} \quad \text{(EQ 19)}$$

which are used to create the length seven Barker sequences. Then, the access code is generated by the following procedure:

1. Format the 30 information bits to encode:

$$x(D) = a(D) + D^{24}B_{a_{23}}(D).$$

2. Add the information covering part of the PN overlay sequence:

$$\tilde{x}(D) = x(D) + p_{34} + p_{35}D + \ldots + p_{63}D^{29}.$$

3. Generate parity bits of the (64,30) expurgated block code:[1]

$$\tilde{c}(D) = D^{34}\tilde{x}(D) \mod g(D).$$

4. Create the codeword:

$$\tilde{s}(D) = D^{34}\tilde{x}(D) + \tilde{c}(D).$$

5. Add the PN sequence:

$$s(D) = \tilde{s}(D) + p(D).$$

6. Append the (DC-free) preamble and trailer:

$$y(D) = F_{c_0}(D) + D^4 s(D) + D^{68}F_{a_{23}}(D).$$

13.2.2 Pseudo-random noise sequence generation

To generate the pseudo-random noise sequence we use the primitive polynomial $h(D) = 1 + D + D^3 + D^4 + D^6$. The LFSR and its starting state are shown in Figure 13.3 on page 147. The PN sequence generated (including the extra terminating zero) becomes (hexadecimal notation) 83848D96BBCC54FC. The LFSR output starts with the left-most bit of this PN sequence. This corresponds to $p^7(D)$ of the previous section. Thus, using the reciprocal $p(D)$ as overlay gives the 64 bit sequence

$$\boldsymbol{p} = 3F2A33DD69B121C1, \quad \text{(EQ 20)}$$

1. $x(D)$ mod $y(D)$ denotes the rest when $x(D)$ is divided by $y(D)$.

where the left-most bit is $p_0 = 0$ (there are two initial zeros in the binary representation of the hexadecimal digit 3), and $p_{63} = 1$ is the right-most bit.

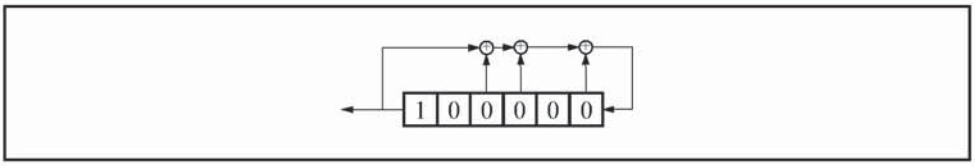

Figure 13.3: LFSR and the starting state to generate $p(D)$

13.2.3 Reserved addresses for GIAC and DIAC

There is a block of 64 contiguous LAPs reserved for Bluetooth inquiry operations; one LAP common to all Bluetooth devices is reserved for general inquiry, the remaining 63 LAPs are reserved for dedicated inquiry of specific classes of Bluetooth devices. The same 64-block is used regardless of the contents of UAP and NAP. Consequently, none of these LAPs can be part of a user BD_ADDR.

When initializing HEC and CRC for the FHS packet of **inquiry response**, the UAP is replaced by DCI. Likewise, whenever one of the reserved BD_ADDRs is used for generating a frequency hop sequence, the UAP will be replaced by the DCI.

The reserved LAP addresses are tentatively chosen as `0x9E8B00-0x9E8B3F`. The general inquiry LAP is tentatively chosen to `0x9E8B33`. All addresses have the LSB at the rightmost position, hexadecimal notation.

13.3 ACTIVE MEMBER ADDRESS (AM_ADDR)

Each slave active in a piconet is assigned a 3-bit active member address (AM_ADDR). The all-zero AM_ADDR is reserved for broadcast messages. The master does not have an AM_ADDR. Its timing relative to the slaves distinguishes it from the slaves. A slave only accepts a packet with a matching AM_ADDR and broadcast packets. The AM_ADDR is carried in the packet header. The AM_ADDR is only valid as long as a slave is active on the channel. As soon as it is disconnected or parked, it loses the AM_ADDR.

The AM_ADDR is assigned by the master to the slave when the slave is activated. This is either at connection establishment or when the slave is unparked. At connection establishment, the AM_ADDR is carried in the **FHS** payload (the **FHS** header itself carries the all-zero AM_ADDR). When unparking, the AM_ADDR is carried in the unpark message.

13.4 PARKED MEMBER ADDRESS (PM_ADDR)

A slave in park mode can be identified by its BD_ADDR or by a dedicated parked member address (PM_ADDR). This latter address is a 8-bit member address that separates the parked slaves. The PM_ADDR is only valid as long as the slave is parked. When the slave is activated it is assigned an AM_ADDR but loses the PM_ADDR. The PM_ADDR is assigned to the slave the moment it is parked.

The all-zero PM_ADDR is reserved for parked slaves that only use their BD_ADDR to be unparked.

13.5 ACCESS REQUEST ADDRESS (AR_ADDR)

The access request address is used by the parked slave to determine the slave-to-master half slot in the access window it is allowed to send access request messages in, see also Section 10.8.4.6 on page 120. The AR_ADDR is assigned to the slave when it enters the park mode and is only valid as long as the slave is parked. The AR_ADDR is not necessarily unique; i.e. different parked slaves may have the same AR_ADDR.

14 BLUETOOTH SECURITY

The Bluetooth technology provides peer-to-peer communications over short distances. In order to provide usage protection and information confidentiality, the system has to provide security measures both at the application layer and the link layer. These measures shall be appropriate for a peer environment. This means that in each Bluetooth unit, the authentication and encryption routines are implemented in the same way. Four different entities are used for maintaining security at the link layer: a public address which is unique for each user[1], two secret keys, and a random number which is different for each new transaction. The four entities and their sizes as used in Bluetooth are summarized in Table 14.1.

Entity	Size
BD_ADDR	48 bits
Private user key, authentication	128 bits
Private user key, encryption configurable length (byte-wise)	8-128 bits
RAND	128 bits

Table 14.1: Entities used in authentication and encryption procedures.

The Bluetooth device address (BD_ADDR) is the 48-bit IEEE address which is unique for each Bluetooth unit. The Bluetooth addresses are publicly known, and can be obtained via MMI interactions, or, automatically, via an inquiry routine by a Bluetooth unit.

The secret keys are derived during initialization and are further never disclosed. Normally, the encryption key is derived from the authentication key during the authentication process. For the authentication algorithm, the size of the key used is always 128 bits. For the encryption algorithm, the key size may vary between 1 and 16 octets (8 - 128 bits). The size of the encryption key shall be configurable for two reasons. The first has to do with the many different requirements imposed on cryptographic algorithms in different countries – both w.r.t. export regulations and official attitudes towards privacy in general. The second reason is to facilitate a future upgrade path for the security without the need of a costly redesign of the algorithms and encryption hardware; increasing the effective key size is the simplest way to combat increased computing power at the opponent side. Currently (1999) it seems that an encryption key size of 64 bits gives satisfying protection for most applications.

The encryption key is entirely different from the authentication key (even though the latter is used when creating the former, as is described in Section 14.5.4 on page 177). Each time encryption is activated, a new encryption key

1. The BD_ADDR is not a secured identity.

shall be generated. Thus, the lifetime of the encryption key does not necessarily correspond to the lifetime of the authentication key.

It is anticipated that the authentication key will be more static in its nature than the encryption key – once established, the particular application running on the Bluetooth device decides when, or if, to change it. To underline the fundamental importance of the authentication key to a specific Bluetooth link, it will often be referred to as the link key.

The RAND is a random number which can be derived from a random or pseudo-random process in the Bluetooth unit. This is not a static parameter, it will change frequently.

In the remainder of this chapter, the terms user and application will be used interchangeably to designate the entity that is at the originating or receiving side.

14.1 RANDOM NUMBER GENERATION

Each Bluetooth unit has a random number generator. Random numbers are used for many purposes within the security functions – for instance, for the challenge-response scheme, for generating authentication and encryption keys, etc. Ideally, a true random generator based on some physical process with inherent randomness is used. Examples of such processes are thermal noise from a semiconductor or resistor and the frequency instability of a free running oscillator. For practical reasons, a software based solution with a pseudo-random generator is probably preferable. In general, it is quite difficult to classify the randomness of a pseudo-random sequence. Within Bluetooth, the requirements placed on the random numbers used are that they be non-repeating and randomly generated.

The expression 'non-repeating' means that it shall be highly unlikely that the value should repeat itself within the lifetime of the authentication key. For example, a non-repeating value could be the output of a counter that is unlikely to repeat during the lifetime of the authentication key, or a date/time stamp.

The expression 'randomly generated' means that it shall not be possible to predict its value with a chance that is significantly larger than 0 (e.g., greater than $1/2^L$ for a key length of L bits).

Clearly, the LM can use such a generator for various purposes; i.e. whenever a random number is needed (such as the RANDs, the unit keys, K_{init}, K_{master}, and random back-off or waiting intervals).

14.2 KEY MANAGEMENT

It is important that the encryption key size within a specific unit cannot be set by the user – this must be a factory preset entity. In order to prevent the user

from over-riding the permitted key size, the Bluetooth baseband processing does not accept an encryption key given from higher software layers. Whenever a new encryption key is required, it must be created as defined in Section 14.5.4 on page 177.

Changing a link key should also be done through the defined baseband procedures. Depending on what kind of link key it is, different approaches are required. The details are found in Section 14.2.2.7 on page 157.

14.2.1 Key types

The link key is a 128-bit random number which is shared between two or more parties and is the base for all security transactions between these parties. The link key itself is used in the authentication routine. Moreover, the link key is used as one of the parameters when the encryption key is derived.

In the following, a session is defined as the time interval for which the unit is a member of a particular piconet. Thus, the session terminates when the unit disconnects from the piconet.

The link keys are either semi-permanent or temporary. A semi-permanent link key is stored in non-volatile memory and may be used after the current session is terminated. Consequently, once a semi-permanent link key is defined, it may be used in the authentication of several subsequent connections between the Bluetooth units sharing it. The designation semi-permanent is justified by the possibility to change it. How to do this is described in Section 14.2.2.7 on page 157.

The lifetime of a temporary link key is limited by the lifetime of the current session – it cannot be reused in a later session. Typically, in a point-to-multipoint configuration where the same information is to be distributed securely to several recipients, a common encryption key is useful. To achieve this, a special link key (denoted master key) can temporarily replace the current link keys. The details of this procedure are found in Section 14.2.2.6 on page 157.

In the sequel we sometimes refer to what is denoted as the current link key. This is simply the link key in use at the current moment. It can be semi-permanent or temporary. Thus, the current link key is used for all authentications and all generation of encryption keys in the on-going connection (session).

In order to accommodate for different types of applications, four types of link keys have been defined:

- the combination key K_{AB}
- the unit key K_A
- the temporary key K_{master}
- the initialization key K_{init}

In addition to these keys there is an encryption key, denoted K_c. This key is derived from the current link key. Whenever the encryption is activated by a LM command, the encryption key shall be changed automatically. The purpose of separating the authentication key and encryption key is to facilitate the use of a shorter encryption key without weakening the strength of the authentication procedure. There are no governmental restrictions on the strength of authentication algorithms. However, in some countries, such restrictions exist on the strength of encryption algorithms.

For a Bluetooth unit, the combination key K_{AB} and the unit key K_A are functionally indistinguishable; the difference is in the way they are generated. The unit key K_A is generated in, and therefore dependent on, a single unit A. The unit key is generated once at installation of the Bluetooth unit; thereafter, it is very rarely changed. The combination key is derived from information in both units A and B, and is therefore always dependent on two units. The combination key is derived for each new combination of two Bluetooth units.

It depends on the application or the device whether a unit key or a combination key is used. Bluetooth units which have little memory to store keys, or, when installed in equipment that must be accessible to a large group of users, will preferably use their own unit key. In that case, they only have to store a single key. Applications that require a higher security level preferably use the combination keys. These applications will require more memory since a combination key for each link to a different Bluetooth unit has to be stored.

The master key, K_{master}, is a link key only used during the current session. It will replace the original link key only temporarily. For example, this may be utilized when a master wants to reach more than two Bluetooth units simultaneously using the same encryption key, see Section 14.2.2.6 on page 157.

The initialization key, K_{init}, is used as link key during the initialization process when no combination or unit keys have been defined and exchanged yet or when a link key has been lost. The initialization key protects the transfer of initialization parameters. The key is derived from a random number, an L-octet PIN code, and the BD_ADDR of the claimant unit. This key is only to be used during initialization.

The PIN can be a fixed number provided with the Bluetooth unit (for example when there is no MMI as in a PSTN plug). Alternatively, the PIN can be selected arbitrarily by the user, and then entered in both units that have to be matched. The latter procedure is used when both units have an MMI, for example a phone and a laptop. Entering a PIN in both units is more secure than using a fixed PIN in one of the units, and should be used whenever possible. Even if a fixed PIN is used, it shall be possible to change the PIN; this in order to prevent re-initialization by users who once got hold of the PIN. If no PIN is available, a default value of zero is to be used.

For many applications the PIN code will be a relatively short string of numbers. Typically, it may consist of only four decimal digits. Even though this gives suffi-

cient security in many cases, there exist countless other, more sensitive, situations where this is not reliable enough. Therefore, the PIN code can be chosen to be any length from 1 to 16 octets. For the longer lengths, we envision the units exchanging PIN codes not through mechanical (i.e. human) interaction, but rather through means supported by software at the application layer. For example, this can be a Diffie-Hellman key agreement, where the exchanged key is passed on to the K_{init} generation process in both units, just as in the case of a shorter PIN code.

14.2.2 Key generation and initialization

The link keys have to be generated and distributed among the Bluetooth units in order to be used in the authentication procedure. Since the link keys must be secret, they cannot be obtained through an inquiry routine in the same way as the Bluetooth addresses. The exchange of the keys takes place during an initialization phase which has to be carried out separately for each two units that want to implement authentication and encryption. All initialization procedures consist of the following five parts:

- generation of an initialization key
- authentication
- generation of link key
- link key exchange
- generating of encryption key in each unit

After the initialization procedure, the units can proceed to communicate, or the link can be disconnected. If encryption is implemented, the E_0 algorithm is used with the proper encryption key derived from the current link key. For any new connection established between units A and B, they will use the common link key for authentication, instead of once more deriving K_{init} from the PIN. A new encryption key derived from that particular link key will be created next time encryption is activated.

If no link key is available, the LM shall automatically start an initialization procedure.

14.2.2.1 Generation of initialization key, K_{init}

A link key used temporarily during initialization is derived – the initialization key K_{init}. This key is derived by the E_{22} algorithm from the BD_ADDR of the claimant unit, a PIN code, the length of the PIN (in octets), and a random number IN_RAND_A issued (and created) by verifier. The principle is depicted in Figure 14.15 on page 177. The 128-bit output from E_{22} will be used for key exchange during the generation of a link key. It is also used for authentication when two

units have no record of a previous link key. When the units have performed the link key exchange, the initialization key shall be discarded.

When the initialization key is generated, the PIN is augmented with the BD_ADDR of the claimant unit. Since the maximum length of the PIN used in the algorithm cannot exceed 16 octets, it is possible that not all octets of BD_ADDR will be used. This procedure ensures that K_{init} depends on the identity of the unit trying to connect to it (at least when short PIN codes are used). A fraudulent Bluetooth unit may try to test a large number of PINs by each time claiming another BD_ADDR. It is the application's responsibility to take countermeasures against this threat. If the device address is kept fixed, the waiting interval until next try is permitted is increased exponentially (see Section 14.4.1 on page 170).

The details of the E_{22} algorithm can be found in Section 14.5.3 on page 175.

14.2.2.2 Authentication

The authentication procedure is carried out as described in Section 14.4 on page 169. If the two units have not been in contact before, the initialization key K_{init} is used as link key. Note that during each authentication, a new AU_RAND_A is issued.

Mutual authentication is achieved by first performing the authentication procedure in one direction and, if successful, immediately followed by performing the authentication procedure in the opposite direction.

As a side effect of a successful authentication procedure an auxiliary parameter, the Authenticated Ciphering Offset (ACO), will be computed. The ACO is used for ciphering key generation as described in Section 14.2.2.5 on page 156. In case of mutual authentication, the ACO value from the second authentication is retained. However, in some situations an authentication event may be initiated simultaneously in both devices. When this happens, there is no way of telling which is the first and which is the second event. Then, both units shall use the ACO resulting from the challenge generated in the master unit.

The claimant/verifier status is determined by the LM.

14.2.2.3 Generation of a unit key

A unit key K_A is generated when the Bluetooth unit is for the first time in operation; i.e. not during each initialization! The unit key is generated by the E_{21} algorithm as described in Section 14.5.3 on page 175. Once created, the unit key is stored in non-volatile memory and (almost) never changed. If after initialization the unit key is changed, the previously initialized units will possess a wrong link key. At initialization, the application has to determine which of the

two parties will provide the unit key as link key. Typically, this will be the unit with restricted memory capabilities, since this unit only has to remember its own unit key. The unit key is transferred to the other party and then stored as link key for that particular party. So, for example in Figure 14.1 on page 155, the unit key of unit A, K_A, is being used as link key for the connection A-B; unit A sends the unit key K_A to unit B; unit B will store K_A as the link key K_{BA}. For another initialization, for example with unit C, unit A will reuse its unit key K_A, whereas unit C stores it as K_{CA}.

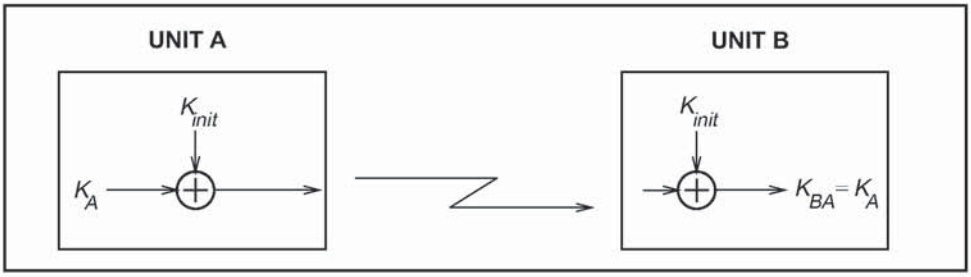

Figure 14.1: Generation of unit key. When the unit key has been exchanged, the initialization key shall be discarded in both units.

14.2.2.4 Generation of a combination key

If it is desired to use a combination key, this key is first generated during the initialization procedure. The combination key is the combination of two numbers generated in unit A and B, respectively. First, each unit generates a random number, say LK_RAND_A and LK_RAND_B. Then, utilizing E_{21} with the random number and the own BD_ADDR, the two random numbers

$$LK\_K_A = E_{21}(LK\_RAND_A, BD\_ADDR_A), \quad \text{(EQ 21)}$$

and

$$LK\_K_B = E_{21}(LK\_RAND_B, BD\_ADDR_B), \quad \text{(EQ 22)}$$

are created in unit A and unit B, respectively. These numbers constitute the units' contribution to the combination key that is to be created. Then, the two random numbers LK_RAND_A and LK_RAND_B are exchanged securely by XOR:ing with the current link key, say K. Thus, unit A sends $K \oplus LK\_RAND_A$ to unit B, while unit B sends $K \oplus LK\_RAND_B$ to unit A. Clearly, if this is done during the initialization phase the link key $K = K_{init}$.

When the random numbers LK_RAND_A and LK_RAND_B have been mutually exchanged, each unit recalculates the other units contribution to the combination key. This is possible since each unit knows the Bluetooth device address of the other unit. Thus, unit A calculates (EQ 22) and unit B calculates (EQ 21).

After this, both units combine the two numbers to generate the 128-bit link key. The combining operation is a simple bitwise modulo-2 addition (i.e. XOR). The result is stored in unit A as the link key K_{AB} and in unit B as the link key K_{BA}. When both units have derived the new combination key, a mutual authentication procedure shall be initiated to confirm the success of the transaction. The old link key shall be discarded after a successful exchange of a new combination key. The message flow between master and slave and the principle for creating the combination key is depicted in Figure 14.2 on page 156.

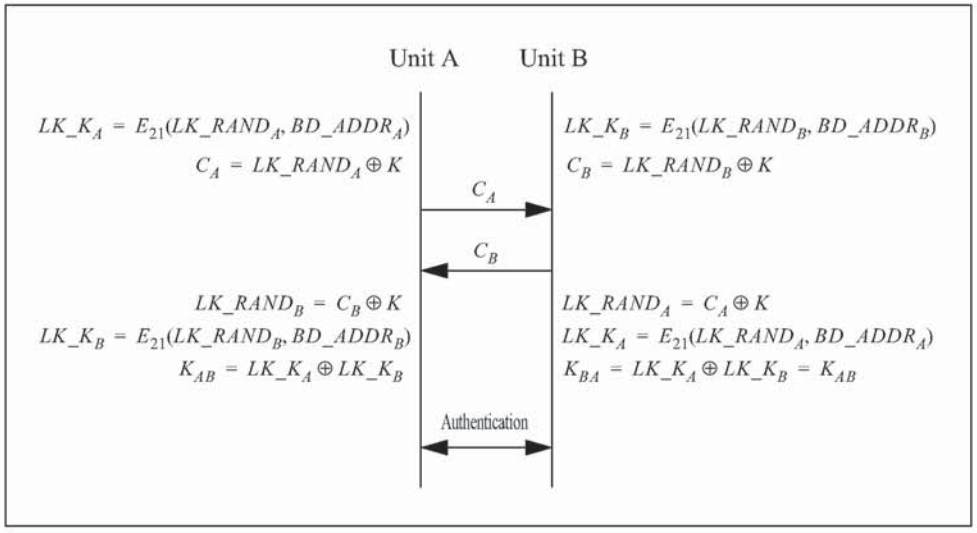

Figure 14.2: Generating a combination key. The old link key (K) shall be discarded after the exchange of a new combination key has succeeded

14.2.2.5 Generating the encryption key

The encryption key, K_C, is derived by algorithm E_3 from the current link key, a 96-bit Ciphering OFfset number (COF), and a 128-bit random number. The COF is determined in one of two ways. If the current link key is a master key, then COF is derived from the master BD_ADDR. Otherwise the value of COF is set to the value of ACO as computed during the authentication procedure. More precisely, we have[1]

$$\text{COF} = \begin{cases} \text{BD\_ADDR} \cup \text{BD\_ADDR}, & \text{if link key is a master key} \\ \text{ACO}, & \text{otherwise.} \end{cases} \quad \text{(EQ 23)}$$

There is an explicit call of E_3 when the LM activates encryption. Consequently, the encryption key is automatically changed each time the unit enters the

1. $x \cup y$ denotes the concatenation of the octet strings x and y.

encryption mode. The details of the key generating function E_3 can be found in Section 14.5.4 on page 177.

14.2.2.6 Point-to-multipoint configuration

It is quite possible for the master to use separate encryption keys for each slave in a point-to-multipoint configuration with ciphering activated. Then, if the application requires more than one slave to listen to the same payload, each slave must be addressed individually. This may cause unwanted capacity loss for the piconet. Moreover, a Bluetooth unit (slave) is not capable of switching between two or more encryption keys in real time (e.g., after looking at the AM_ADDR in the header). Thus, the master cannot use different encryption keys for broadcast messages and individually addressed traffic. Alternatively, the master may tell several slave units to use a common link key (and, hence, indirectly also to use a common encryption key) and broadcast the information encrypted. For many applications, this key is only of temporary interest. In the sequel, this key is denoted K_{master}.

The transfer of necessary parameters is protected by the routine described in Section 14.2.2.8 on page 158. After the confirmation of successful reception in each slave, the master shall issue a command to the slaves to replace their respective current link key by the new (temporary) master key. Before encryption can be activated, the master also has to generate and distribute a common EN_RAND to all participating slaves. Using this random number and the newly derived master key, each slave generates a new encryption key.

Note that the master must negotiate what encryption key length to use individually with each slave who wants to use the master key. Since the master has already negotiated at least once with each slave, it has some knowledge of what sizes can be accepted by the different slaves. Clearly, there might be situations where the permitted key lengths of some units are incompatible. In that case, the master must have the limiting unit excluded from the group.

When all slaves have received the necessary data, the master can communicate information on the piconet securely using the encryption key derived from the new temporary link key. Clearly, each slave in possession of the master key can eavesdrop on all encrypted traffic, not only the traffic intended for itself. If so desired, the master can tell all participants to fall back to their old link keys simultaneously.

14.2.2.7 Modifying the link keys

In certain circumstances, it is desirable to be able to modify the link keys. A link key based on a unit key can be changed, but not very easily. The unit key is created once during the first use. Changing the unit key is a less desirable alternative, as several units may share the same unit key as link key (think of a printer whose unit key is distributed among all users using the printer's unit key

as link key). Changing the unit key will require re-initialization of all units trying to connect. In certain cases, this might be desirable; for example to deny access to previously allowed units.

If the key change concerns combination keys, then the procedure is rather straightforward. The change procedure can be identical to the procedure illustrated in Figure 14.2 on page 156, using the current value of the combination key as link key. This procedure can be carried out at any time after the authentication and encryption start. In fact, since the combination key corresponds to a single link, it can be modified each time this link is established. This will improve the security of the system since then old keys lose their validity after each session.

Of course, starting up an entirely new initialization procedure is also a possibility. In that case, user interaction is necessary since a PIN is required in the authentication and encryption procedures.

14.2.2.8 Generating a master key

The key-change routines described so far are semi-permanent. To create the master link key, which can replace the current link key during an initiated session (see Section 14.2.2.6), other means are needed. First, the master creates a new link key from two 128-bit random numbers, RAND1 and RAND2. This is done by

$$K_{master} = E_{22}(\text{RAND1}, \text{RAND2}, 16). \quad \text{(EQ 24)}$$

Clearly, this key is a 128-bit random number. The reason to use the output of E_{22} and not directly chose a random number as the key, is to avoid possible problems with degraded randomness due to a poor implementation of the random number generator within the Bluetooth unit.

Then, a third random number, say RAND, is transmitted to the slave. Using E_{22} with the current link key and RAND as inputs, both the master and slave computes a 128-bit overlay. The master sends the bitwise XOR of the overlay and the new link key to the slave. The slave, who knows the overlay, recalculates K_{master}. To confirm the success of this transaction, the units can perform an authentication procedure using the new link key (with the master as verifier and the slave as claimant). This procedure is then repeated for each slave who shall receive the new link key. The ACO values from the involved authentications should not replace the current existing ACO as this ACO is needed to (re)compute a ciphering key when the master wants to fall back to the previous link (non-temporary) key.

When so required – and potentially long after the actual distribution of the master key – the master activates encryption by an LM command. Before doing that, the master must ensure that all slaves receive the same random number,

say EN_RAND, since the encryption key is derived through the means of E_3 individually in all participating units. Then, each slave computes a new encryption key,

$$K_C = E_3(K_{master}, \text{EN\_RAND}, \text{COF}), \quad \text{(EQ 25)}$$

where the value of COF is derived from the master's BD_ADDR as specified by equation (EQ 23). The details on the encryption key generating function can be found in Section 14.5.4 on page 177. The principle of the message flow between the master and slave when generating the master key is depicted in Figure 14.3. Note that in this case the ACO produced during the authentication is not used when computing the ciphering key.

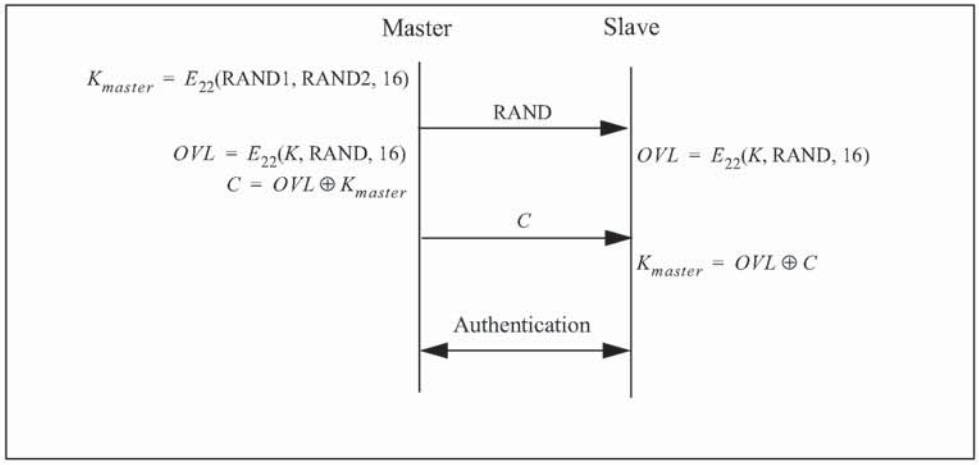

Figure 14.3: Master link key distribution and computation of the corresponding encryption key.

14.3 ENCRYPTION

User information can be protected by encryption of the packet payload; the access code and the packet header are never encrypted. The encryption of the payloads is carried out with a stream cipher called E_0 that is re-synchronized for every payload. The overall principle is shown in Figure 14.4 on page 160.

The stream cipher system E_0 consists of three parts. One part performs the initialization (generation of the payload key), the second part generates the key stream bits, and the third part performs the encryption and decryption. The payload key generator is very simple – it merely combines the input bits in an appropriate order and shift them into the four LFSRs used in the key stream generator. The main part of the cipher system is the second, as it also will be used for the initialization. The key stream bits are generated by a method derived from the summation stream cipher generator attributable to Massey and Rueppel. The method has been thoroughly investigated, and there exist good estimates of its strength with respect to presently known methods for cryptanalysis. Although the summation generator has weaknesses that can be

used in so-called correlation attacks, the high re-synchronization frequency will disrupt such attacks.

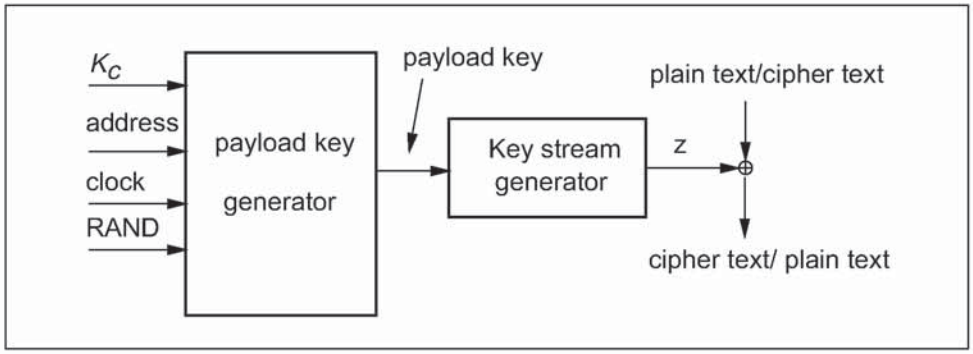

Figure 14.4: Stream ciphering for Bluetooth with E_0.

14.3.1 Encryption key size negotiation

Each Bluetooth device implementing the baseband specification needs a parameter defining the maximal allowed key length, L_{max}, $1 \le L_{max} \le 16$ (number of octets in the key). For each application, a number L_{min} is defined indicating the smallest acceptable key size for that particular application. Before generating the encryption key, the involved units must negotiate to decide what key size to actually use.

The master sends a suggested value, $L_{sug}^{(M)}$, to the slave. Initially, the suggested value is set to $L_{max}^{(M)}$. If $L_{min}^{(S)} \le L_{sug}^{(M)}$, and, the slave supports the suggested length, the slave acknowledges and this value will be the length of the encryption key for this link. However, if both conditions are not fulfilled, the slave sends a new proposal, $L_{sug}^{(S)} < L_{sug}^{(M)}$, to the master. This value should be the largest among all supported lengths less than the previous master suggestion. Then, the master performs the corresponding test on the slave suggestion. This procedure is repeated until a key length agreement is reached, or, one unit aborts the negotiation. An abortion may be caused by lack of support for L_{sug} and all smaller key lengths, or if $L_{sug} < L_{min}$ in one of the units. In case of abortion Bluetooth link encryption can not be employed.

The possibility of a failure in setting up a secure link is an unavoidable consequence of letting the application decide whether to accept or reject a suggested key size. However, this is a necessary precaution. Otherwise a fraudulent unit could enforce a weak protection on a link by claiming a small maximum key size.