# Circuit Description for CU328-US

This is a 2.4GHz WDCT cordless telephone with type 2 caller ID & call waiting and analog clock display for domestic use. Radio transmitter with FM technology provides greater mobility to the user within approximately 200 meters radius around the base.

Following paragraphs describe the detail of major building blocks.

## 1. Ringer Detection

### a. Base

Incoming ringer signal is first attenuated by C4, D12, D8,R3,R1 and Q11. The signal is then feed to micro-controller (MCU) U100 for generating response signal according to the setting of inputs. When the ringer switch is set to on position MCU sends digitally coded information to handset via RF link.

### b. Handset

When digitally coded information is received from the base it will be decoded at MCU U301. Then necessary ringer is generated and applied to U400, which drive the Speaker J400.

## 2. Surge protection

The surge absorber VR1 is mounted in the Base unit. It designed to operate when voltage over 330V. In general it is common to have induced surges in the telephone line due to lightening. If it allow entering the unit damage to the unit is imminent. The line interface, fuse and ringer detected circuit is most venerable to high voltage surges and VR1 surge absorber can prevent it.

## 3. Line control

When the unit is operated by remote handset, line control is done by MCU. It turns on transistor Q7. Then telephone line power feeds to line interface circuit (Q5), turn on the telephone line and internal voice path, and around component.

## 4. Power Control

### a. Base unit

The main power is come from AC/DC adaptor, which provide 9V DC to the unit. Inside the unit there are two different voltages available for different modules. 9V non-back up voltage is supplied to the audio amplifier. Radio part, MCU and line interface related circuit is supplied with non-backup regulated 3.3V voltage.

### b. Handset

Three cells of Ni-MH battery(3.6V) provided necessary power to the handset. In order to keep power consumption to minimum, the radio receiver is turn on and off periodically by MCU and Q303,Q304. The MCU is supplied with regulated 1.8V and 2.5V by U301.

## 5. RF system specification

This radio system uses the ISM band from 2400MHz to 2483.5MHz. The modulation is a standard GFSK, with 0.864MHz channel bandwidth and the data is transmitted at a bit rate of 576 kbit/s. All transmissions are hopped equally over 95 channels, implemented by the frequency hopping spread spectrum technique. The regular TDMA is used. The frame structure is repeated every 10ms, corresponding to 5760-bit duration, and will not be greater than 0.4 seconds on any frequency. It is in accordance with the FCC rules. This frame is divided into 4 uplink slots lasting 5ms, and 4 downlink slots. So the system is either receiving or transmitting at any given time.

### **a**.Receiver

The receiver of the radio chipset incorporates a LNA, a mixer, channel filters, amplifier stage with RSSI indicator, demodulator and data slicer.

The channel filter in the receiver path is based on the Low-IF architecture. It is equivalent to a band-pass filter around to 864kHz IF frequency combined with image rejection architecture. It can be used to further reduce the receiver’s susceptibility to out-of-band signals. Due to this IF frequency, the LO frequency should be 864kHz above the wanted RF frequency.

The data slicer converts the demodulated analog data to NRZ (binary) format.

One pin diode (D1) works as the T/R switch (Transmit / Receive Switch), which selects between the reception and transmission paths.

In our radio system, it can support two antennas. Another pin diode (D2) is performing the antenna selection. It is controlled by the signal supplied by baseband controller for the implementation of antenna diversity.

The PCB printed band-pass filter is used between T/R switch and the antenna switch. It is used to remove partially the unwanted signals occurring outside the DECT band, and to reduce undesired emissions during transmission.

### **b**.Transmitter

The transmitter of the radio chipset utilizes GFSK architecture. The VCO oscillates at twice the ISM 2.4GHz band, then pass through the integrated frequency divider and divide the signal to desired RF frequency in the 2400 to

2483.5MHz range. The VCO is directly modulated with GFSK filtered data. The transmitted power is typically 3dBm.

An external silicon power amplifier is used. It consists of 3 cascaded gain stages and amplifies the typical 3dBm at its input to nominal 20dBm at the antenna ports including overall loss. Base and handset have the same power level. The external power amplifier connects to the transmitter of the radio chipset with appropriate impedance matching and harmonics filtering components.

The same RF signal path as mentioned in the receiver, the external power amplifier also connects with T/R switch, PCB printed band-pass filter and the antenna switch. They have the same function as described in the receiver’s section.

### **c.**Voltage Controlled Oscillator (VCO) and Synthesizer path

The fully integrated VCO operates at twice the ISM 2.4GHz band frequency. The output of the prescaler is used to drive the synthesizer main divider and this output can also be switched to either the transmitter or the receiver LO output buffer of the radio chipset.

The main divider is clocked by the RF signal from the prescaler at frequencies from 2400 to 2483 MHz. The reference divider is clocked by a 13.824MHz crystal. It must be aligned better than +/- 2 ppm for frequency accuracy.

The Phase-Lock Loop (PLL) tunes the VCO to a given channel based on control signal of 3-line serial bus. These control signals are come from the baseband controller. The minimum PLL locking time is 300 us.

### **d.** Antenna

The antennas utilized in the base and handset are quarter-wave monopole. Since quarter-wave monopole is small in size and the antenna wire is mounted over the ground plane, the antenna gain is 1 dBi for this configuration. In the base unit, it contains two antennas for antenna diversity. In the handset unit, it contains only one antenna.

### **e. Shield Can**

The radio module is fully covered with a suitable shield can in order to minimize the interference to other equipment or device.

## 6. MP3, SD Card Memory Read/Write and Radio Function

This unit is capable to read and write MP3 to the SD card through the computer via the USB cable. The MP3 player function can be activated by the user to turn on the U2 S5L840 chipset. Which is a single chip digital audio player IC supporting various compressed audio format on flash memory media.

It’s internal DSP function provide various musical effect such as Pop, Jazz, Rock, Classic, Live and Normal. The MP3 and Radio function can be interrupted when ringer on and resume after off the line. The DAC data is sent to U1 WM8751L as input. This chipset is a low power high quality stereo DAC with integrated headphone and loudspeaker amplifiers for portable digital audio. The on chip headphone can deliver 40mW to a 16 ohm load.

Due to the insufficient signal level from the DAC U1 WM8751 . Therefore the keyboard circuit needs to add an additional power amplifier U5 LM4992 to drive the 8 ohm load speaker. This chipset is capable of delivering 1 watt per channel of continuous power to an 8 ohm BTL load with less than 1% THD. The U1 is independent shut down control for each channel and a low power consumption. Which is a unity gain stable and can be configured by external gain setting resistors.

The radio features can be turned on by activating the Self Tuned Radio module T1 in the menu. The tuning frequency range at 87.5 Mhz to 108 Mhz, the channel can also be auto/manual scanned and stored through the MCU U301. The tuning quality is superior and requires no IF counter for stop-detection.

# PHILIPS WDCT System Enhanced Hopper

Version 2.0p 2004 Jul 15
File under ABU11-CUS-A0001,
m11/documents/ABU09-m11-a/0007/

Philips
Semiconductors

PHILIPS

# PHILIP WDCT System Enhanced Hopper PHE V200

## 1 SCOPE

The document gives an overview about the Philips WDCT System with Enhanced Hopper. It describes the TDMA structure and the basics of the hopping algorithm used. It refers to the FCC regulations and explains how the FCC regulations are met.

## 2 WDCT SYSTEM IMPLEMENTATION

### 2.1 Basics

- 8 Bearer slots per frame at 0.576MBits per sec
- Bearers on fixed slot but frequency agile (hopping)
- ID dependent hopper 95 frequencies
- Synchronization randomly selects frequency and tries all slots
- Duplicated bearer data on new fixed slot and different agile frequency on quality problem

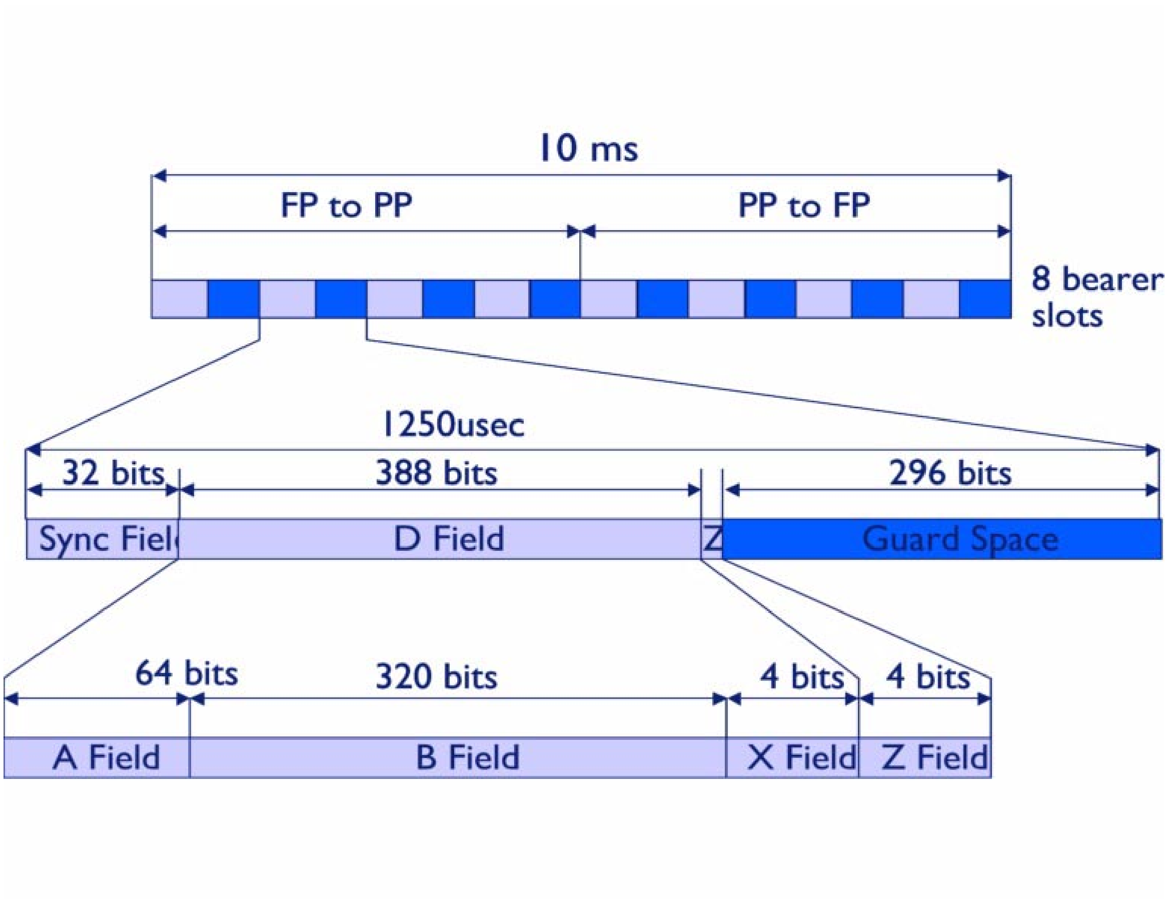


Fig.1 WDCT Frame structure

### 2.2 Duplicate Bearer Data (DBD) Mode

Duplicated bearer means that the information is sent twice within 10ms. The hopper is slot based.

- Bearer data is duplicated in free bearer slots on different frequency
- Second bearer avoids any interferer on first frequency
- Triggered by quality problem detected by PP or by user via MMI
- Maintained until quality problem no longer exists or user selects via MMI
- Only used when necessary as use reduces stand-by and talk time

## 3 HOP SEQUENCES

### 3.1 General description

Each system, comprising a BS and its associated HSs, is given the specific value of an identity value during manufacture, this value is specific to one system only. This value is used by the algorithm to generate hopping sequences that are used by this specific system. There is a 95 element sequence generated.

The 95 element sequence is used for all dummy bearers transmitted by the BS and for setting up all traffic bearers.

The hopping sequences are generated in such a way that the elements are random in both distance from each other and in the direction from each other.

When a BS transmits it uses one entry from the sequence in each frame in a cyclic manner so that once the end of the sequence is reached, the first element is taken again and so on. This means that over time each entry in the sequence will be used equally.

Each slot in the same frame is separated from the previous slot by one entry in the sequence.

Because the BS and the HSs in a system have the same identity value they generate identical sequences, so that once a HS has made a reception from the BS on any element in the sequence it can follow the same sequence and maintain synchronisation.

Because each system only knows the identity of its own hopping sequence identity and not any other system values it cannot predict other hopping sequences and so it does not have the ability to be coordinated with other FHSS systems.

### 3.2 Hop sequences

To generate the hop sequence a ‘Frequency Picking’ approach is used. Hop frequencies are picked one at a time from those available and appended to the partially formed sequence. The way in which hop frequencies are picked and appended is dependent on the given ID. The proprietary algorithm used guarantees that:

- All 95 frequencies are always used in 1 hop sequence
- A minimum hop distance is guaranteed
- A good random spread over all channels is guaranteed

Example of the hopping sequences:
95 element sequence: (70, 57, 37, 30, 24, 7, 65, 42, 34, 81, 91, 21, 5, 38, 85, 54, 68, 20, 6, 92, 50, 32, 14, 58, 40, 2, 79, 67, 15, 41, 73, 55, 93, 47, 29, 80, 11, 89, 71, 19, 59, 77, 9, 53, 90, 0, 83, 36, 62, 18, 43, 27, 72, 66, 8, 44, 22, 74, 88, 63, 48, 26, 75, 82, 3, 39, 28, 45, 86, 60, 16, 25, 76, 51, 31, 13, 94, 78, 46, 64, 23, 4, 12, 84, 52, 61, 35, 1, 17, 69, 56, 10, 49, 87, 33)

## 3.3 Frequency substitution (optional)

There is a possibility to use a frequency substituting mechanism to omit disturbed frequency channels. This substitution mechanism must be switched on by the manufacturer. So the manufacturer has to declare if it is enabled.

The design of the system utilises two different hopping sequence lengths to gain the best performance from the different length sequences at different times.The two lengths that are utilised are 95 element and 19 element sequences. This is a convenient multiple allowing simple modulo arithmetic to be utilised to calculate which frequency to use at any given frame.

After reset the FP shall request the Hopping Engine to generate the hopping sequence for the identity that it obtains from the base ID number stored in EEPROM. The Hopping Engine generates two different sequences, one 95 elements long and one 19 elements long. The 19 element sequence is copied into 9 arrays, one for storage of the defined sequence and one for each slot.

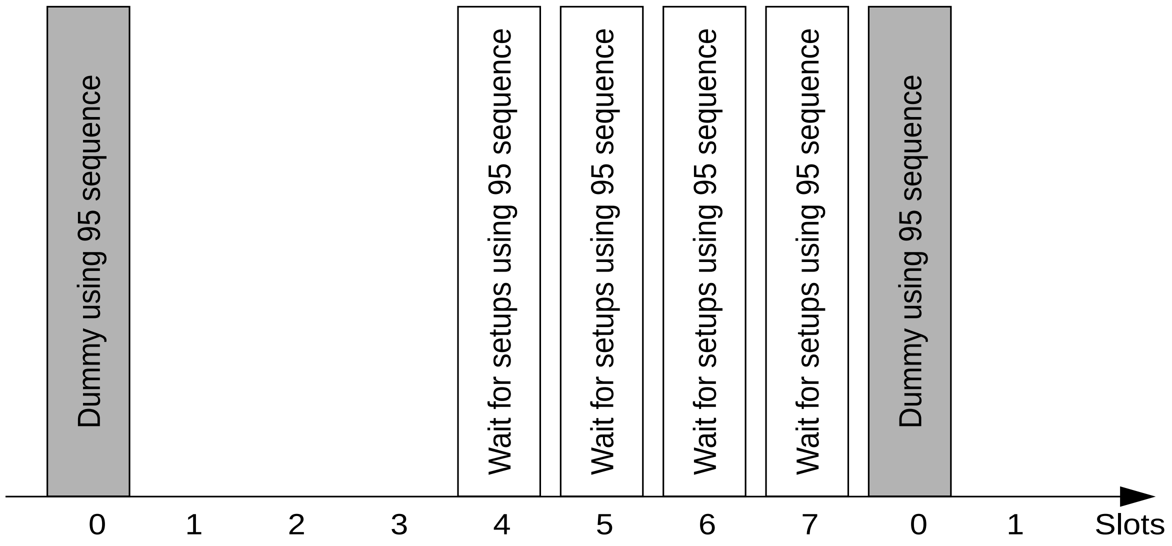

Dummy using 95 sequence
Wait for setups using 95 sequence
Wait for setups using 95 sequence
Wait for setups using 95 sequence
Wait for setups using 95 sequence
Dummy using 95 sequence
0
1
2
3
4
5
6
7
0
1
Slots

Fig.2 FP Transmitting one Dummy in Slot 0

Once the sequences have been generated the FP starts to transmit a dummy bearer using the 95 element long sequence. It looks for setups using the 95 element sequence.
When a setup is detected the traffic bearer is started using the 95 element sequence and the FP Quality Monitor maintains a value for the quality of the link on all frequencies.

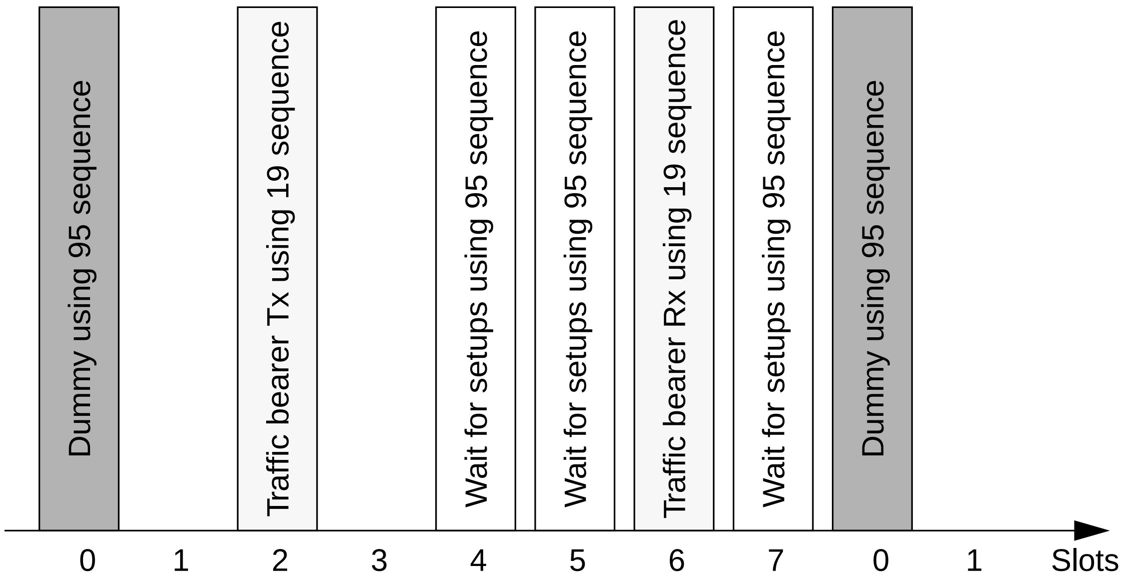

Dummy using 95 sequence
Traffic bearer Tx using 19 sequence
Wait for setups using 95 sequence
Wait for setups using 95 sequence
Traffic bearer Rx using 19 sequence
Wait for setups using 95 sequence
Dummy using 95 sequence
0
1
2
3
4
5
6
7
0
1
Slots

Fig.3 FP Traffic Bearer in Slot 2 and 6 Switched to 19 Element sequences

The 19 element sequences for the Rx and Tx slots of the traffic bearer are checked against the quality values being maintained and any frequencies that are detected as being bad frequencies are replaced in the 19 element sequence. This replacement is achieved by requesting a new frequency from the Hopping Engine and passing this information to the PP over the Ht Channel on the traffic bearer (which is still using the 95 element sequence). This process occurs until a threshold of good frequencies in the 19 element sequences is reached, (the threshold of good frequencies can already have been reached in which case no frequency substitutions are required). Once the threshold has been reached the PP is requested to switch the hopping sequences in use for this traffic bearer from the 95 element sequence to the 19 element sequences (one for Tx and one for Rx) that have just reached the threshold of good frequencies. This request is sent to the PP using the Ht Channel.

When operating using the 19 element sequences (one for Tx and one for Rx) any quality problems on frequencies within the sequences are detected and new frequencies obtained from the Hopping Engine. Then the 19 element sequences in use are modified and this information passed to the PP over the Ht Channel only now the 19 element sequence is already the sequence in use.

When the traffic bearer is taken down the 19 element sequences are returned to the default 19 element sequences (one for Rx and one for Tx). If this would result in the bearer being converted to the FPs only dummy bearer, then during the disconnection time the FP requests the PP to switch the list back to the 95 element sequence. This information is passed to the PP over the Ht Channel.

## 4 FCC REGULATIONS

1. section 15.247(a):

describe how the EUT meets the definition of a frequency hopping spread spectrum system, found in Section 2.1, based on the technical description.

- pseudo random frequency hopping sequence: describe how the hopping sequence is generated. Provide an example of the hopping sequence channels, in order to demonstrate that the sequence meets the requirement specified in the definition of a frequency hopping spread spectrum system.

- see 3.2 Hop sequences

- Equal hopping frequency use: describe how each individual EUT meets the requirement that each of its hopping channels is used equally on average (e.g., that each new transmission event begins on the next channel in the hopping sequence after the final channel used in the previous transmission event).

- The hop sequence does not contain any hop channel more than once. This ensures that the chosen 95 (19) frequencies are used evenly as required

- System receiver input bandwidth: describe how the associated receiver(s) complies with the requirement that its input bandwidth (either RF or IF) matches the bandwidth of the transmitted signal.

- A band pass filter is implemented on the RF side with pass band of the ISM band. The filter is fixed implemented in the transceiver IC (uaa3548) used

- System receiver hopping capability: describe how the associated receiver(s) has the ability to shift frequencies synchronization with the transmitted signals.

- A fixed frequency hopper dependent on the individual base ID is used. The sequence is not changed. The receiver knows the sequence and can follow it (see 3.2 Hop sequences)

2. Section 15.247(g):

describe how the EUT complies with the requirement that it be designed to be capable of operating as a true frequency hopping system.

- The hopper without frequency substitution is using all 95 frequency channels defined for 2.4G ISM. The hopper is using all 95frequency channels only once. The mean time of channel occupation of 30/95 = 0.315 is guaranteed by those means.
- The hopeer with 19 channels is using all 19 frequency channels once. The mean time of channel occupation is 30/19 = 1.58. I has to be approved to the regulations for ‘other frequency hopping systems‘

3. Section 15.247(h):

describe how the EUT complies with the requirement that it do not have the ability to be coordinated with other FHSS systems in an effort to avoid the simultaneous occupancy of individual hopping frequencies by multiple transmitters.

- The pseudo random sequence is generated ID based. So every set has an own seuquence.This is guaranteed by simulation done within Philips.

iDT
Integrated Display Technology Limited
萬 威 科 技 有 限 公 司

## LS328 WDCT Cordless Phone Base and Handset Frequency Table

| **Channel** | **Frequency (Hz)** | Channel | Frequency (Hz) | Channel | Frequency (Hz) |
|---:|---:|---:|---:|---:|---:|
| 0 | 2.401056E+09 | 42 | 2.437344E+09 | 84 | 2.473632E+09 |
| 1 | 2.401920E+09 | 43 | 2.438208E+09 | 85 | 2.474496E+09 |
| 2 | 2.402784E+09 | 44 | 2.439072E+09 | 86 | 2.475360E+09 |
| 3 | 2.403648E+09 | 45 | 2.439936E+09 | 87 | 2.476224E+09 |
| 4 | 2.404512E+09 | 46 | 2.440800E+09 | 88 | 2.477088E+09 |
| 5 | 2.405376E+09 | 47 | 2.441664E+09 | 89 | 2.477952E+09 |
| 6 | 2.406240E+09 | 48 | 2.442528E+09 | 90 | 2.478816E+09 |
| 7 | 2.407104E+09 | 49 | 2.443392E+09 | 91 | 2.479680E+09 |
| 8 | 2.407968E+09 | 50 | 2.444256E+09 | 92 | 2.480544E+09 |
| 9 | 2.408832E+09 | 51 | 2.445120E+09 | 93 | 2.481408E+09 |
| 10 | 2.409696E+09 | 52 | 2.445984E+09 | 94 | 2.482272E+09 |
| 11 | 2.410560E+09 | 53 | 2.446848E+09 | | |
| 12 | 2.411424E+09 | 54 | 2.447712E+09 | | |
| 13 | 2.412288E+09 | 55 | 2.448576E+09 | | |
| 14 | 2.413152E+09 | 56 | 2.449440E+09 | | |
| 15 | 2.414016E+09 | 57 | 2.450304E+09 | | |
| 16 | 2.414880E+09 | 58 | 2.451168E+09 | | |
| 17 | 2.415744E+09 | 59 | 2.452032E+09 | | |
| 18 | 2.416608E+09 | 60 | 2.452896E+09 | | |
| 19 | 2.417472E+09 | 61 | 2.453760E+09 | | |
| 20 | 2.418336E+09 | 62 | 2.454624E+09 | | |
| 21 | 2.419200E+09 | 63 | 2.455488E+09 | | |
| 22 | 2.420064E+09 | 64 | 2.456352E+09 | | |
| 23 | 2.420928E+09 | 65 | 2.457216E+09 | | |
| 24 | 2.421792E+09 | 66 | 2.458080E+09 | | |
| 25 | 2.422656E+09 | 67 | 2.458944E+09 | | |
| 26 | 2.423520E+09 | 68 | 2.459808E+09 | | |
| 27 | 2.424384E+09 | 69 | 2.460672E+09 | | |
| 28 | 2.425248E+09 | 70 | 2.461536E+09 | | |
| 29 | 2.426112E+09 | 71 | 2.462400E+09 | | |
| 30 | 2.426976E+09 | 72 | 2.463264E+09 | | |
| 31 | 2.427840E+09 | 73 | 2.464128E+09 | | |
| 32 | 2.428704E+09 | 74 | 2.464992E+09 | | |
| 33 | 2.429568E+09 | 75 | 2.465856E+09 | | |
| 34 | 2.430432E+09 | 76 | 2.466720E+09 | | |
| 35 | 2.431296E+09 | 77 | 2.467584E+09 | | |
| 36 | 2.432160E+09 | 78 | 2.468448E+09 | | |
| 37 | 2.433024E+09 | 79 | 2.469312E+09 | | |
| 38 | 2.433888E+09 | 80 | 2.470176E+09 | | |
| 39 | 2.434752E+09 | 81 | 2.471040E+09 | | |
| 40 | 2.435616E+09 | 82 | 2.471904E+09 | | |
| 41 | 2.436480E+09 | 83 | 2.472768E+09 | | |

Note: E+09 represents GHz

# Software Design Description
# D69 WDCT HE and Interference Rejection

|  |  |  |
|---|---|---|
| Author | : | J.D.Wiggett |
| Document Id | : | ABD11-D69-DD001 |
| Version | : | 1.1 |
| Status | : | Accepted |
| Date | : | 15-Aug-03 |
| Main Project | : | WDCT Hop Engine & Improved Interference Rejection for Vega Family |
| Main Project Id | : | D69 |
| Sub-Project Id | : | N/A |

Keywords : WDCT, Hopping Engine, Substitution

## Revision History

| Version | Date | Description | Author |
|---|---|---|---|
| 1.0 | 04-Jul-03 | Created | J.D.Wiggett |
| 1.1 | 15-Aug-03 | New message formats for Ht Channel, minor mods | J.D.Wiggett |
| | | New LMAC requirements | |

CONTENTS

# 1 Introduction

## 1.1 Purpose and Scope

This document details the additions and changes that must be made to the existing WDCT software in order to provide:

- a hopping engine based on the system identity
- hopping sequence frequency substitution to provide a means of interference rejection
- feature selection for members of the Vega family (not including the VegaPro)

This document is aimed at people involved in the D69 project and the project sponsor, familiarity with the Philips Semiconductors implementation of the WDCT system for the Vega Family is assumed.

This project will not generate a complete product on its own, it extends an existing product to provide a complete system solution for a WDCT cordless telephone system, so that customers only need to develop an MMI in order to enter the market.

## 1.2 Glossary

| | |
|---|---|
| **C** | A high level language |
| **FP_HE** | Hopping engine process in the FP |
| **FP_QM** | Quality monitor process in the FP |
| **FP_SUB** | Substitution process in the FP |
| **FT** | Fixed part |
| **FT_USE** | User defined process in the FP |
| **HE** | Hopping Engine |
| **Ht Channel** | Inter MAC layer communication channel |
| **PP_HE** | Hopping engine process in the PP |
| **PP_QM** | Quality monitor process in the PP |
| **PP_SUBS** | Substitution process in the PP |
| **PT** | Portable part |
| **PT_USE** | User defined process in the PP |
| **Rx** | Receive |

| | |
|---|---|
| **Tx** | Transmit |
| **VegaFamily** | Generic term for all chips of the Vega IC series for digital cordless applications |

## 1.3 Abbreviation

| | |
|---|---|
| **DBD** | Duplicate Bearer Data |
| **EEPROM** | Electrically Erasable Programmable Read Only Memory |
| **FP** | Fixed Part |
| **HMAC** | Higher MAC layer |
| **IC** | Integrated Circuit |
| **LMAC** | Lower MAC layer |
| **LSB** | Least Significant Bit |
| **MAC** | Media Access Control |
| **MMI** | Man Machine Interface |
| **MSC** | Message Sequence Chart |
| **OS** | Operating System |
| **PP** | Portable Part |
| **RFPI** | Radio Fixed Part Identity |
| **WDCT** | World Digital Cordless Telephony |

## 1.4 References

[1] Software Design Description D60 WDCT Software from DECT Stack Higher MAC - ABD06-D60-DD002

[2] D53 Software Design Description

[3] Hop Engine Interface Description

[4] Report Further Development of FALCON Hop Sequences; Report No. 3881

[5] Software Design Description HLP

# 2 Context

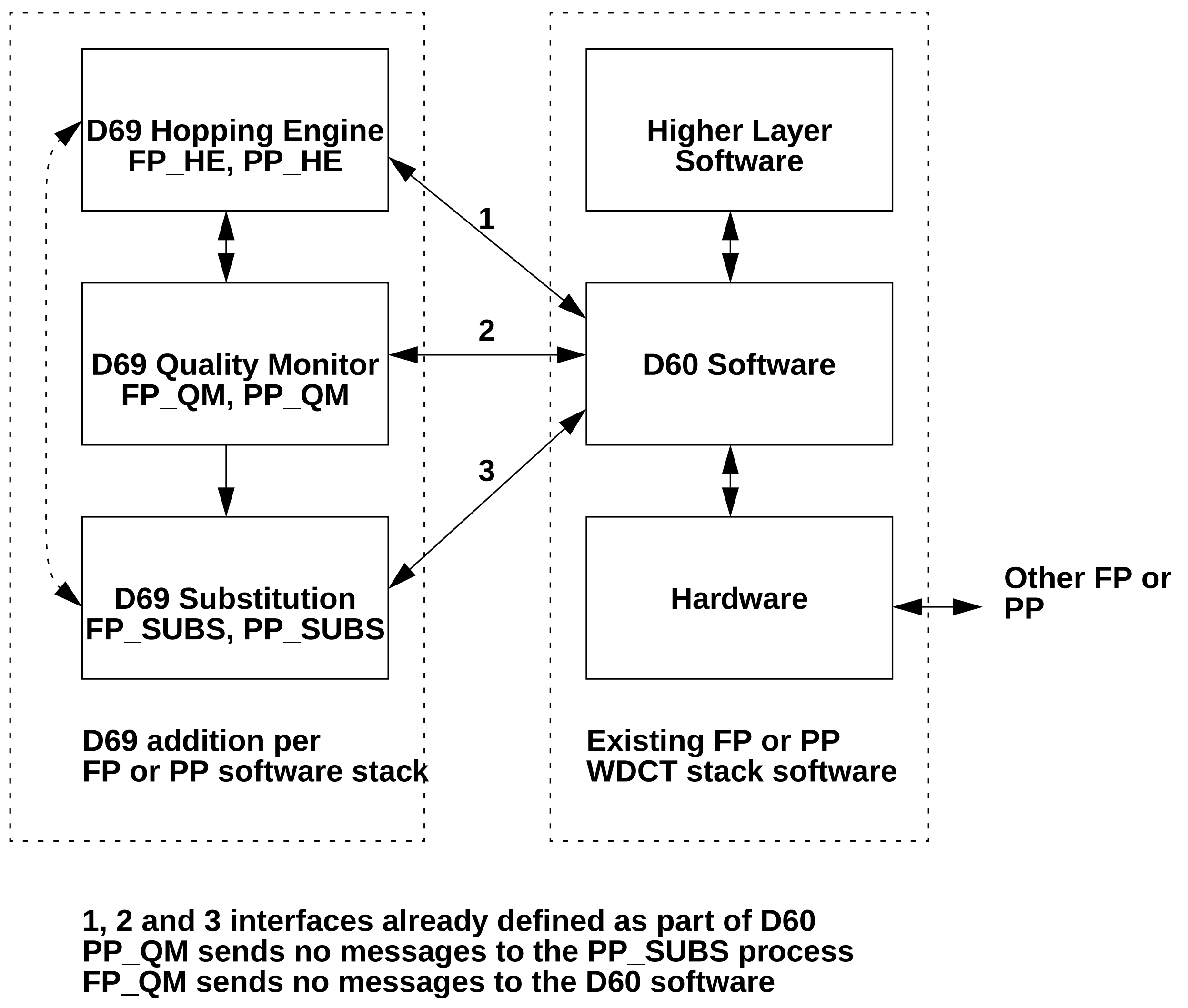


**Figure 2.1** Context of D69 Software

In the terms of the HMAC ‘processes’ described in [1] the three D69 software blocks shown in Figure 2.1 form the FT_USE and PT_USE processes. These blocks differ in detailed operation depending on whether they are in the FT or the PT, the differences are detailed in this document. The interfaces between the D69 software blocks and the D60 software can be found in [1] and [3].

## 2.1 Hopping Engine (FP_HE, PP_HE)

This block generates the sequence of frequencies (hopping sequence) to use for the system identity for which it is called, and any substitution frequencies if required.

## 2.2 Quality Monitor (FP_QM, PP_QM)

This block receives information about the quality of all the links being maintained from the D60 software and maintains a history so that channels suffering interference can be detected.

## 2.3 Substitution block (FP_SUBS, PP_SUBS)

This block controls the D60 software to communicate the changes to the frequency sequences in use with the other relevant FP or PP. Any new channels to use are requested from the Hopping Engine in the FP.

## 2.4 Operation

### 2.4.1 Use of Different Length Hopping Sequences

The design of the system utilises two different hopping sequence lengths to gain the best performance from the different length sequences at different times.

During the locking phase when the FP is transmitting a dummy bearer the handset must use the same hopping sequence if it is to find this bearer. If some of the frequencies in the sequence are suffering from interference the FP could exchange them with other frequencies. This causes problems for handsets that are trying to lock to the FP as extra information needs to be transmitted to enable the PPs to follow the modified hopping sequence. During the locking and locked periods the handset does not need to receive all transmissions as information intended for the handset tends to be repeated. Because of this a certain level of interfered frequencies can be accepted, so a static (i.e. unchanged) hopping sequence is preferred. In order to obtain the optimal performance from a static sequence under all conditions a long sequence is best as this only suffers from whatever percentage of frequencies are interfered, whereas a short list could suffer worse or could be better but there is no control over which it is. (For example in a system suffering from 10% interference a 95 element sequence will suffer from 10% interference, but a 19 element short list will suffer between 0% and 50% interference, in the worst case this could prevent locking).

During active connections any missed frames are detectable by the user in the form of muted audio or clicks in the audio, so interfered frequencies are less acceptable during active connections. When in an active connection the bandwidth is available to send the extra information about the modified hopping sequence from the FP to the PP. So modifying the hopping sequence can be performed to improve the quality of active connections. This is best achieved on a short hopping sequence so that the number of exchanges of frequencies in the sequence that are suffering from interference with ones that are good that is required is as small as possible whilst still remaining a random hopping sequence. (For example in a system suffering from 10% interference, 20 element short sequences will on average only require 2 frequency exchanges to obtain a sequence that does not suffer any interference).

The two lengths that are utilised are 95 element and 19 element sequences. This is a convenient multiple allowing simple modulo arithmetic to be utilised to calculate which frequency to use at any given frame.

### 2.4.2 FP

After reset the FP shall request the Hopping Engine to generate the hopping sequence for the identity that it obtains from the RFPI stored in EEPROM. The Hopping Engine generates two different sequences, one 95 elements long and one 19 elements long. The 19 element sequence is copied into 9 arrays, one for storage of the defined sequence and one for each slot.

Once the sequences have been generated the FP starts to transmit a dummy bearer using the 95 element long sequence. It looks for setups using the 95 element sequence.

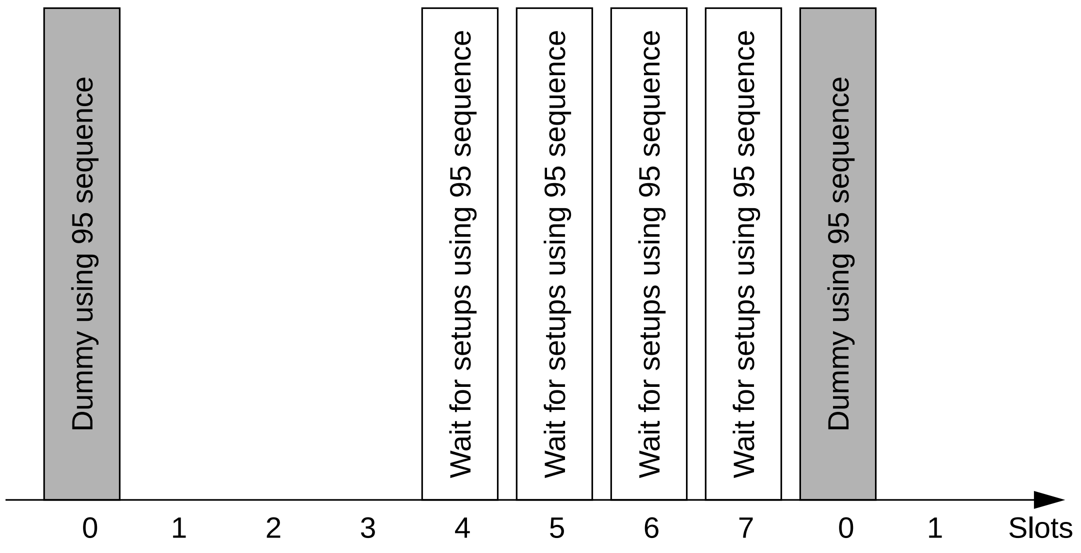

Dummy using 95 sequence
Wait for setups using 95 sequence
Wait for setups using 95 sequence
Wait for setups using 95 sequence
Wait for setups using 95 sequence
Dummy using 95 sequence
0
1
2
3
4
5
6
7
0
1
Slots

**Figure 2.2** FP Transmitting one Dummy in Slot 0

When a setup is detected the traffic bearer is started using the 95 element sequence and the FP

Quality Monitor maintains a value for the quality of the link on all frequencies.

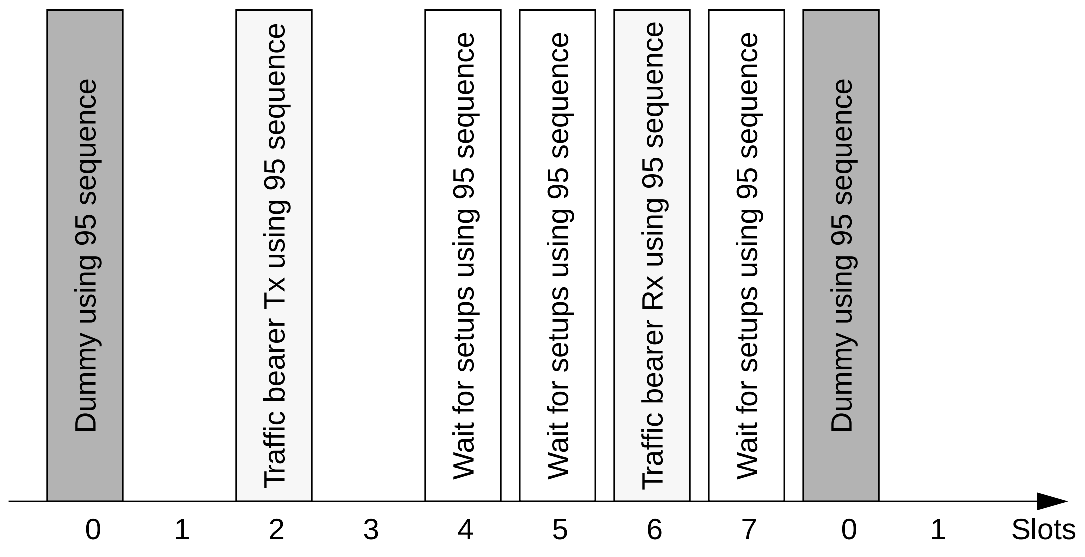

Dummy using 95 sequence
Traffic bearer Tx using 95 sequence
Wait for setups using 95 sequence
Wait for setups using 95 sequence
Traffic bearer Rx using 95 sequence
Wait for setups using 95 sequence
Dummy using 95 sequence
0 1 2 3 4 5 6 7 0 1
Slots

**Figure 2.3** FP Maintaining a Dummy in Slot 0 and a Traffic Bearer in Slot 2 and 6

The 19 element sequences for the Rx and Tx slots of the traffic bearer are checked against the quality values being maintained and any frequencies that are detected as being bad frequencies are replaced in the 19 element sequence. This replacement is achieved by requesting a new frequency from the Hopping Engine and passing this information to the PP over the Ht Channel on the traffic bearer (which is still using the 95 element sequence). This process occurs until a threshold of good frequencies in the 19 element sequences is reached, (the threshold of good frequencies can already have been reached in which case no frequency substitutions are required). Once the threshold has been reached the PP is requested to switch the hopping sequences in use for this traffic bearer from the 95 element sequence to the 19 element sequences (one for Tx and one for Rx) that have just reached the threshold of good frequencies. This request is sent to the PP using the Ht Channel.

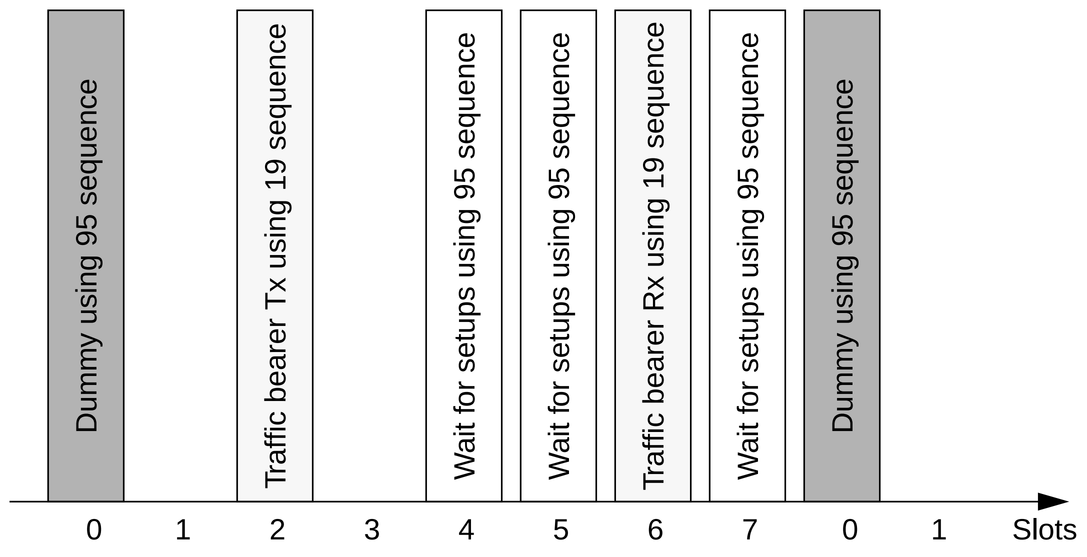

Dummy using 95 sequence
Traffic bearer Tx using 19 sequence
Wait for setups using 95 sequence
Wait for setups using 95 sequence
Traffic bearer Rx using 19 sequence
Wait for setups using 95 sequence
Dummy using 95 sequence
0
1
2
3
4
5
6
7
0
1
Slots

**Figure 2.4** FP Traffic Bearer in Slot 2 and 6 Switched to 19 Element sequences

When operating using the 19 element sequences (one for Tx and one for Rx) any quality problems on frequencies within the sequences are detected and new frequencies obtained from the Hopping Engine. Then the 19 element sequences in use are modified and this information passed to the PP over the Ht Channel only now the 19 element sequence is already the sequence in use.

When the traffic bearer is taken down the 19 element sequences are returned to the default 19 element sequences (one for Rx and one for Tx). If this would result in the bearer being converted to the FPs only dummy bearer, then during the disconnection time the FP requests the PP to switch the list back to the 95 element sequence. This information is passed to the PP over the Ht Channel.

### 2.4.3 PP

After reset the PP selects which subscription to use and the Hopping Engine is requested to generate the hopping sequences from the identity of the subscription. The Hopping Engine generates two different sequences, one 95 elements long and one 19 elements long. The 19 element sequence is copied into 9 arrays, one for storage of the defined sequence and one for each slot.

Once the sequences have been generated the PP starts to look for an FP to lock to, it selects a frequency from the 95 element sequence at random and starts to setup sync bearers to find the

FP. Once found the PP maintains lock using the 95 element sequence.

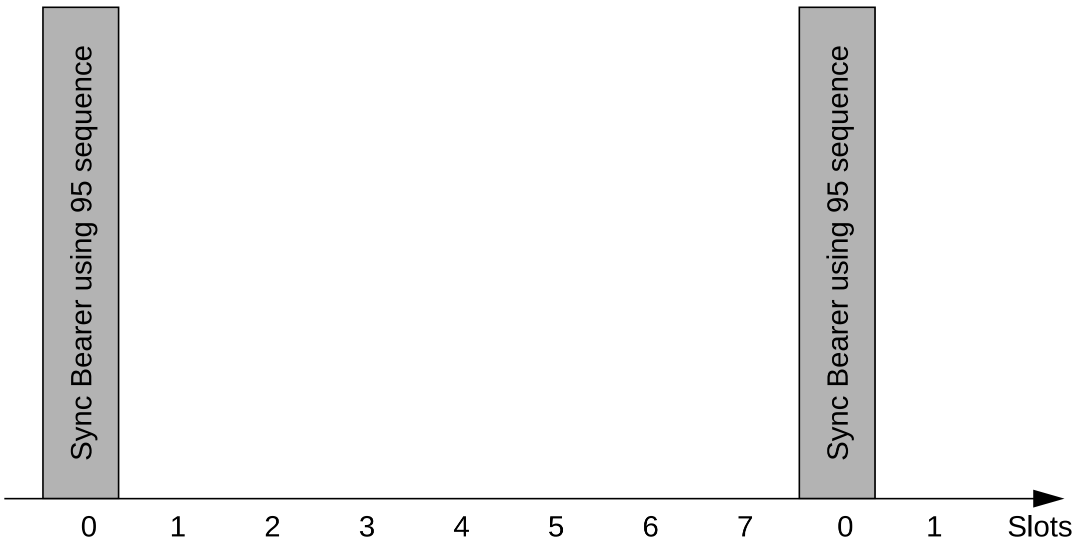


**Figure 2.5** PP Maintaining Lock using a Sync Bearer in Slot 0

When setting up traffic bearers the 95 element sequence is used.

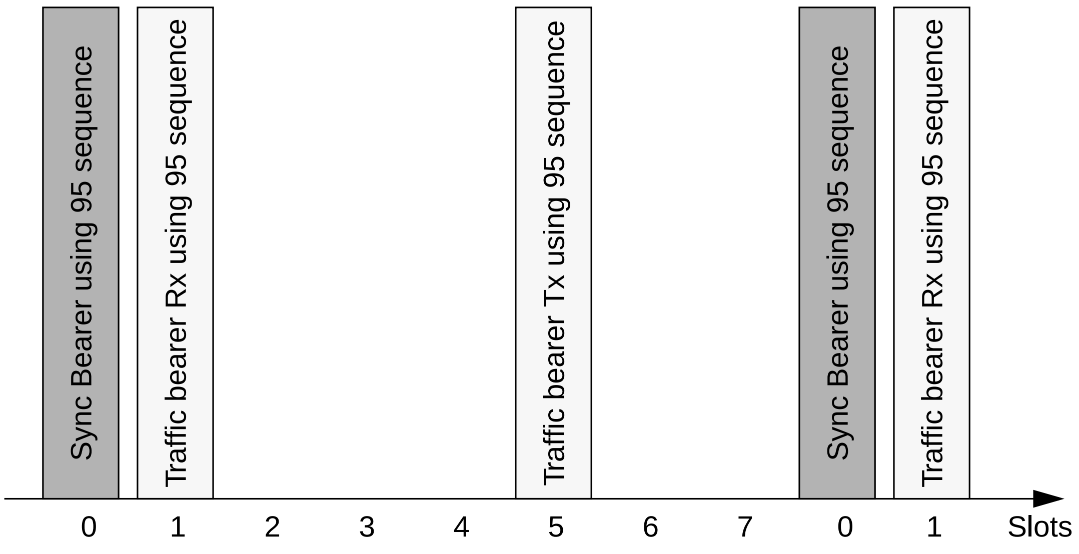

Sync Bearer using 95 sequence
Traffic bearer Rx using 95 sequence
Traffic bearer Tx using 95 sequence
Sync Bearer using 95 sequence
Traffic bearer Rx using 95 sequence
0
1
2
3
4
5
6
7
0
1
Slots

**Figure 2.6** PP Maintaining a Sync Bearer in Slot 0 and a Traffic Bearer in Slot 1 and 5

Once the traffic bearer is set up the quality of the link is monitored and if a threshold of the ratio of bad receptions to good receptions is exceeded the PP will attempt to set up a DBD bearer. Note that this is independent of the FP monitoring the quality of the link to check for individual frequencies that need to be substituted.

Messages received via the Ht Channel of the traffic bearer indicate that the FP has detected problems with certain frequencies that need to be replaced in the 19 element sequences for the Rx and Tx slots of the traffic bearer. These frequencies are replaced as soon as the information is received.

Requests to exchange the hopping sequence in use from the 95 element sequence to the 19 element sequences or vice versa for the slots in use are received via the Ht Channel, and action is taken immediately to start the exchange procedure, but the actual exchange of sequences occurs at a later time specified in the message.

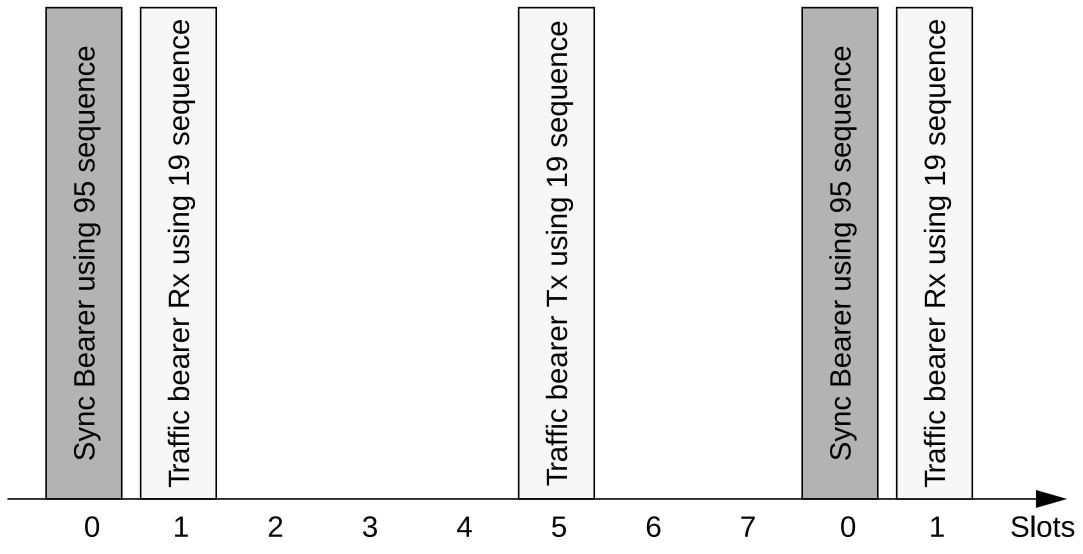

Sync Bearer using 95 sequence
Traffic bearer Rx using 19 sequence
Traffic bearer Tx using 19 sequence
Sync Bearer using 95 sequence
Traffic bearer Rx using 19 sequence
0
1
2
3
4
5
6
7
0
1
Slots

**Figure 2.7** PP Traffic Bearer in Slot 1 and 5 Switched to 19 Element sequences

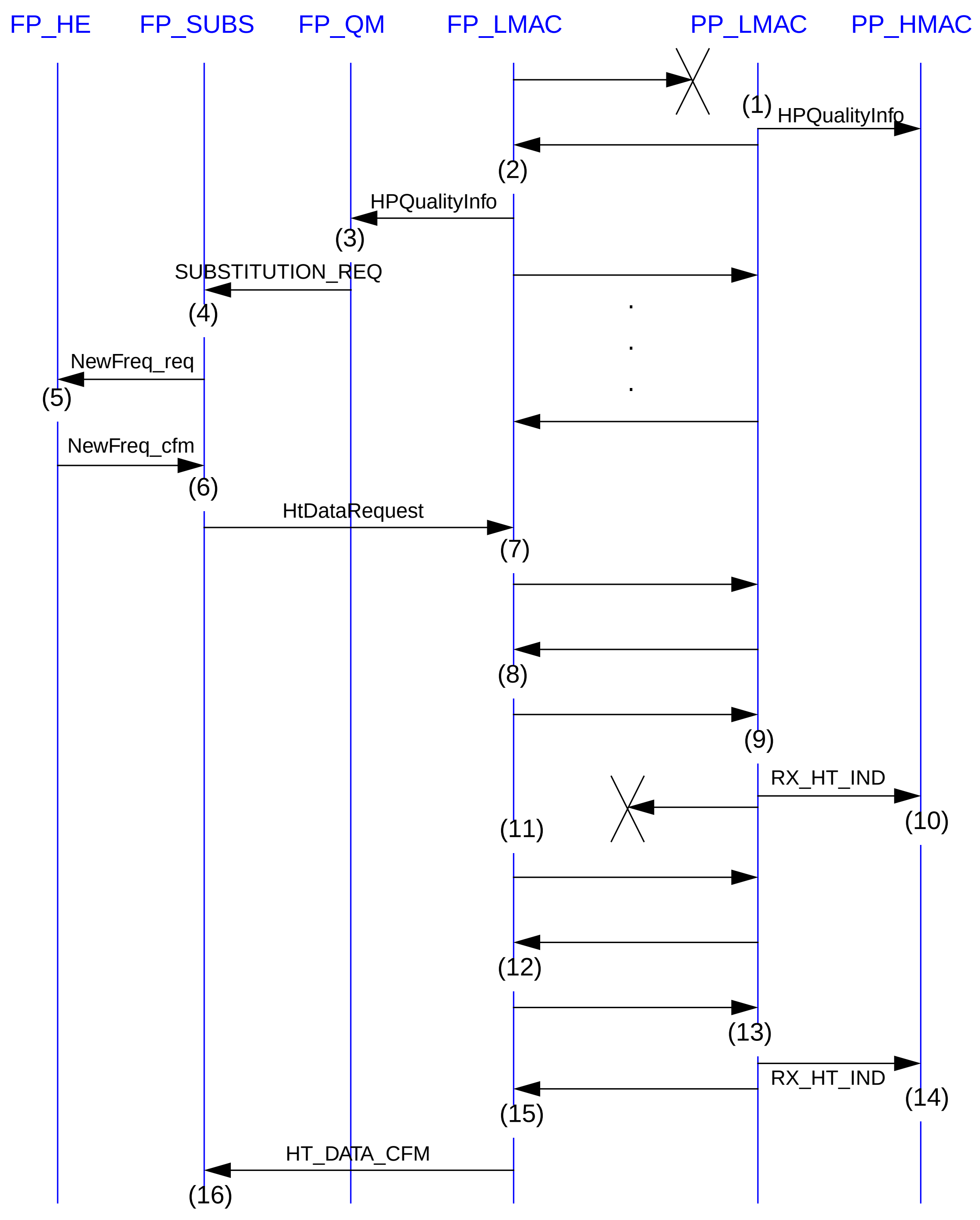
Once the traffic bearer is operating using the 19 element sequences any quality problems on frequencies within the sequences are detected by the FP and this information is passed via the Ht Channel only now the 19 element sequence is already the sequence in use.

When the traffic bearer is taken down the 19 element sequences are returned to the default 19 element sequences (one for Rx and one for Tx).

### 2.4.4 Message Exchange for Frequency Substitution

The following diagram shows a message sequence chart for the frequency substitution procedure. Not all processes are shown, and some processes are not true operating processes.

The chart starts when there is a traffic bearer established between the FP and the PP using either the 95 or the 19 element sequence, which sequence is in use has no effect on the sequence of messages. The frequencies to exchange are exchanged in the 19 element sequences that are related to the slots in use for the active connection not the 95 element sequence even if this is the one that is currently in use. The hopping sequence in use does affect which frequency each message is sent on but this is not shown in the MSC.



(1) A reception is made at the PP_LMAC in which there is an error, this information is passed to the PP_QM and the PP_LMAC indicates the failed reception in its next transmission. The PP_QM process stores the fact that an error has occurred and monitors the quality of the link to determine if DBD mode should be used.
(2) A reception is made at the FP_LMAC in which the error in the previous PP_LMAC reception is indicated, this information is passed to the FP_QM.
(3) The FP_QM updates the quality measure for the frequency for which the error has been reported, and detects that the measure has passed the substitution threshold. The BUSY flag for the FP_SUBS process for this connection indicates that the FP_SUBS process is not busy, and so the FP_QM sets the BUSY flag and sends a SUBSTITUTION_REQ message to the FP_SUBS process.
(4) The FP_SUBS process requests a new frequency from the FP_HE. (Note this is actually a procedure call not a message handshake).
(5) The FP_HE replies with a new frequency to use in the list
(6) The FP_SUBS process generates an Ht Channel message and stores it in a buffer for the LMAC to use, and requests the LMAC to send the Ht message.
(7) The FP_LMAC tail multiplexer does not allow the Ht Channel message to be sent in this frame
(8) The FP_LMAC tail multiplexer determines that the highest priority message is the waiting Ht Channel message and transmits the message.
(9) The PP_LMAC receives the Ht Channel message correctly and sends it to the PP_HMAC, it sets its next transmission to indicate successful reception of the Ht Channel message.
(10) The PP_HMAC decodes the message and updates the frequency information in the relevant 19 element sequence.
(11) The FP_LMAC reception has an error so the Ht Channel message is set to be repeated, but the FP_LMAC tail multiplexer does not allow the Ht Channel message to be sent in this frame
(12) The FP_LMAC tail multiplexer determines that the highest priority message is the waiting Ht Channel message and transmits the message.
(13) The PP_LMAC receives the Ht Channel message correctly and sends it to the PP_HMAC, it sets its next transmission to indicate successful reception of the Ht Channel message.
(14) The PP_HMAC rejects the Ht Channel message as its header indicates that the message is not a new one.
(15) The FP_LMAC reception indicates that the PP_LMAC successfully received the Ht Channel message and so the FP_LMAC informs the FP_SUBS process.
(16) The FP_SUBS process clears the BUSY flag so that it is available to send further Ht Channel messages and updates the 19 element sequence with the information that it sent in the Ht Channel.

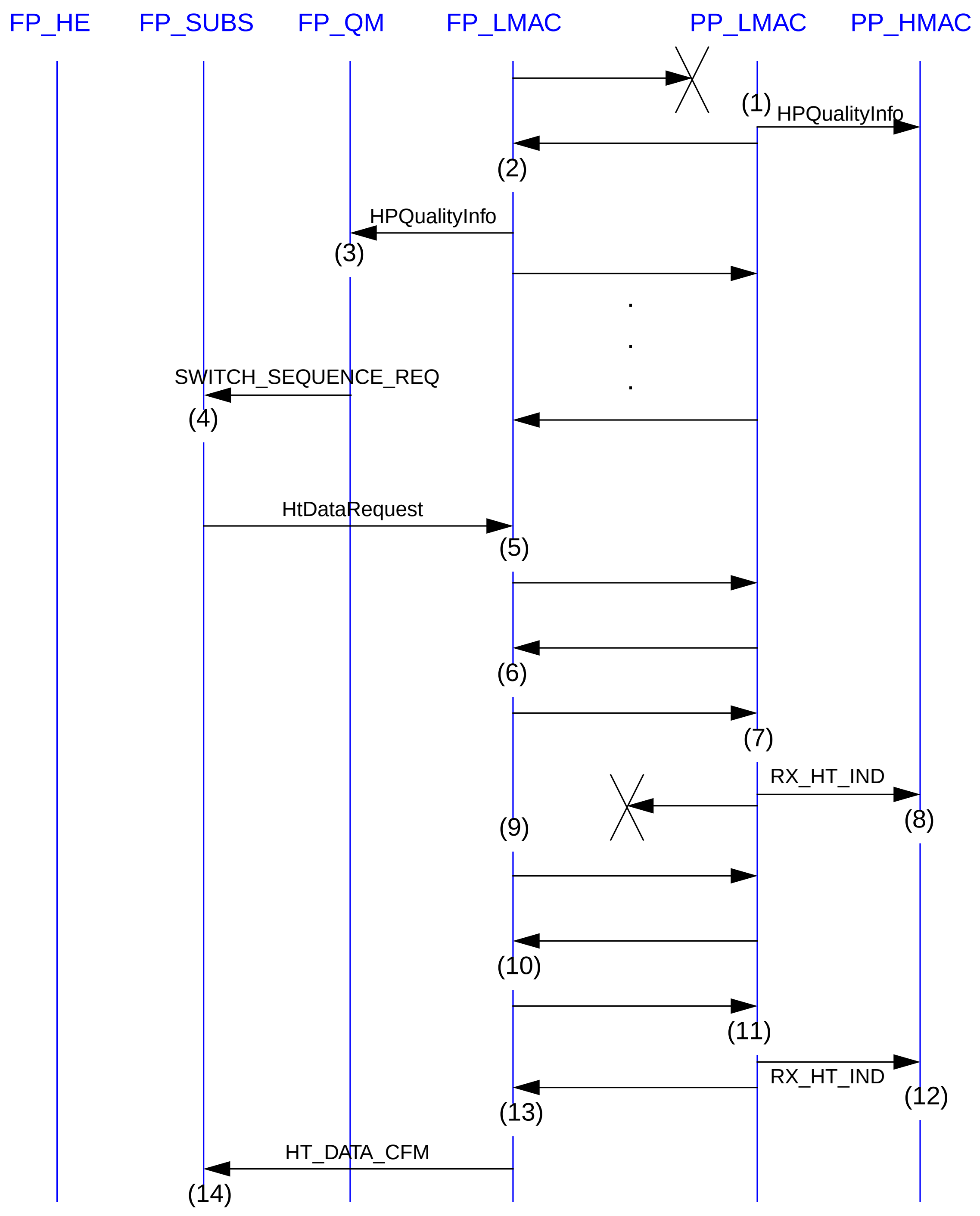
Notes.

- Only the communication between the FP_LMAC and the PP_LMAC relevant to the frequency substitution is fully detailed, other messages are shown for completeness.
- This chart only shows one possible sequence, other sequences are possible where transmission failures occur in different places or don’t occur at all.
- Every reception results in the HPQualityInfo function call, but calls that do not affect the frequency substitution procedure have been left out for clarity.

- It is possible that the PP may have substituted the frequency in the sequence and the frequency may be used before the FP makes the substitution. In this case communication is not possible on that sequence element. However, no further substitutions can take place until this one is complete and the reason for making the change was because the frequency at this element in the sequence had problems, so the quality of the connection is not reduced by this so this is not considered to be a problem.

### 2.4.5 Message Exchange for Switching Lists

The chart starts when there is a traffic bearer established between the FP and the PP using the 95 element sequence.



(1) A reception is made at the PP_LMAC in which there is an error, this information is passed to the PP_QM and the PP_LMAC indicates the failed reception in its next transmission. The PP_QM process stores the fact that an error has occurred and monitors the quality of the link to determine if DBD mode should be used.
(2) A reception is made at the FP_LMAC in which the error in the previous PP_LMAC reception is indicated, this information is passed to the FP_QM, along with the information about the good reception.
(3) The FP_QM updates the quality measure for the frequency for which the good reception has been reported, and detects that the number of good frequencies in the 19 element sequence has passed the threshold. The BUSY flag for the FP_SUBS process for this connection indicates that the FP_SUBS process is not busy, and so the FP_QM sets the BUSY flag and sends a SWITCH_SEQUENCE_REQ message to the FP_SUBS process.
(4) The FP_SUBS process obtains the value of the PSCN counter from the LMAC and uses this along with the SWITCH_SEQUENCE_REQ message to generate an Ht Channel message and stores it in a buffer for the LMAC to use, and requests the LMAC to send the Ht message.
(5) The FP_LMAC tail multiplexer does not allow the Ht Channel message to be sent in this frame
(6) The FP_LMAC tail multiplexer determines that the highest priority message is the waiting Ht Channel message and transmits the message.
(7) The PP_LMAC receives the Ht Channel message correctly and sends it to the PP_HMAC, it sets its next transmission to indicate successful reception of the Ht Channel message.
(8) The PP_HMAC decodes the message and uses the information to set the PSCN counter value at which to switch the sequence from the 95 element sequence to the 19 element sequence.
(9) The FP_LMAC reception has an error so the Ht Channel message is set to be repeated, but the FP_LMAC tail multiplexer does not allow the Ht Channel message to be sent in this frame
(10) The FP_LMAC tail multiplexer determines that the highest priority message is the waiting Ht Channel message and transmits the message.
(11) The PP_LMAC receives the Ht Channel message correctly and sends it to the PP_HMAC, it sets its next transmission to indicate successful reception of the Ht Channel message.
(12) The PP_HMAC rejects the Ht Channel message as its header indicates that the message is not a new one.
(13) The FP_LMAC reception indicates that the PP_LMAC successfully received the Ht Channel message and so the FP_LMAC informs the FP_SUBS process.
(14) The FP_SUBS process clears the BUSY flag so that it is available to send further Ht Channel messages and uses the information sent in the Ht Channel message to set the PSCN counter value at which to switch the sequence from the 95 element sequence to the 19 element sequence.

Notes.

- Only the communication between the FP_LMAC and the PP_LMAC relevant to the switch sequence request is fully detailed, other messages are shown for completeness.
- This chart only shows one possible sequence, other sequences are possible where transmission failures occur in different places or don’t occur at all.
- Every reception results in the HPQualityInfo function call, but calls that do not affect the sequence switch procedure have been left out for clarity.

- It is possible that the PP may have switched the sequence before the FP makes a good reception after transmitting the Ht Channel message. In this case communication is no longer possible. However, for this to occur the link would have to be so bad that it would be unusable anyway so this is not considered to be a problem.
- This chart is shown for the switch of sequences from 95 element to 19 element, the switch for the opposite direction is similar except for which sequences are in use and that the trigger to start the switch is generated by the higher layers instead of the quality of the reception.

### 2.4.6 DBD Mode

- DBD mode is allowed on any selection of slots.
- DBD mode is only permitted
  - if there are at least 2 remaining inactive slots, i.e. when either one or two active connections exist, neither of which are in DBD mode
  - when all subscribed handsets are in active links and there are slots available.

When setting up a bearer for DBD mode operation, the traffic bearer to use is setup using the 95 element sequence as per a main bearer, but the 19 element sequences to use are the 19 element sequences that are in use on the main bearer. The sequences for the DBD bearer are shifted by one entry from the sequences for the main bearer, this is to maintain the spectral separation between bearers in a DBD mode connection, which is controlled by the hopping sequence generation algorithm, see [4]. The sequence for the new bearer is stored in the array for the DBD bearer slot for each of the Rx and Tx bearers.

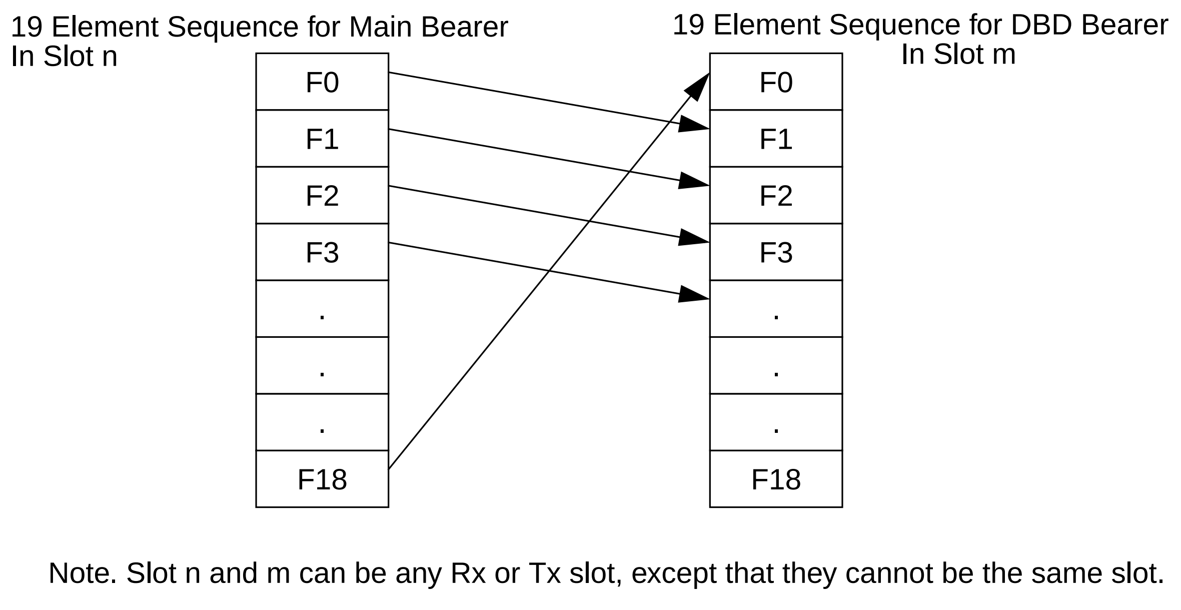
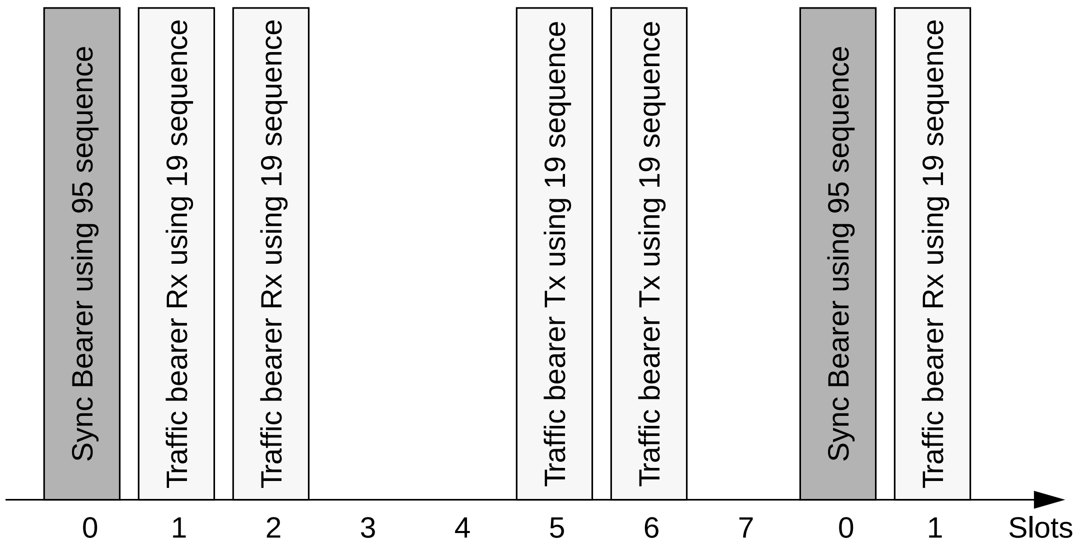

19 Element Sequence for Main Bearer
In Slot n
19 Element Sequence for DBD Bearer
In Slot m
F0
F1
F2
F3
.
.
.
F18
F0
F1
F2
F3
.
.
.
F18
Note. Slot n and m can be any Rx or Tx slot, except that they cannot be the same slot.

**Figure 2.8** Generation of DBD Bearer Sequence from Main Bearer Sequence

Frequency substitutions are applied, regardless of whether the information was sent on the main or the DBD bearer, so that the frequencies in use for both bearers are always kept the same. If the main bearer is taken down the DBD bearer becomes the main bearer and continues to use the sequence that it has inherited.

Note diagram shown for PP only, the FP is similar.


Sync Bearer using 95 sequence
Traffic bearer Rx using 19 sequence
Traffic bearer Rx using 19 sequence
Traffic bearer Tx using 19 sequence
Traffic bearer Tx using 19 sequence
Sync Bearer using 95 sequence
Traffic bearer Rx using 19 sequence
0
1
2
3
4
5
6
7
0
1
Slots

Note. Rx bearer in slot 2 uses the same 19 element sequence as the Rx bearer in slot 1, but shifted by 1 entry. This is true regardless of the slot separation of the two Rx Bearers. Similarly the Tx bearer in slot 6 uses the same sequence as the Tx bearer in slot 5 but shifted by 1 entry.

**Figure 2.9** PP Traffic Bearer in Slot 1 and 5 and DBD Bearer in Slot 2 and 6

Once a DBD mode connection is in operation if a further setup request is detected and this is on the only available slot, if the number of handsets subscribed requires another slot to be made available to detect setups, the DBD mode connection is taken down. Then a new dummy bearer is set up on the slot that the DBD bearer was in. The newly setup traffic bearer cannot switch from the 95 element sequence to the 19 element sequence until the new dummy position information has been transmitted by the FP four times on the old dummy bearer.

If the FP is rejecting DBD bearer setup attempts, due to needing the slot to allow other handsets to setup, the standard PP connection timers are used to prevent a PP that wants to setup a DBD bearer from monopolising the remaining slot. If this proves to be inadequate the blind slot information message from the FP shall be modified to send 2 bits of information per slot, one is the standard slot blind or not blind as the message is now, and the second bit is slot blind or not blind for DBD access. The PP will then be able to use this information to decide whether to try to setup a DBD bearer.

### 2.4.7 Dummy Bearers

If there are subscribed handsets that are not involved in active connections, the FP must always maintain one bearer as a dummy bearer operating on the 95 element sequence. If a traffic bearer sets up on the dummy bearer then a new dummy bearer must be set up, and the traffic bearer cannot switch from the 95 element sequence to the 19 element sequence until the new dummy position information has been transmitted by the FP four times on the old dummy bearer. In order to attempt to reduce the impact of this delay, the PPs will weight the slot selection algorithm so that the dummy slot is less likely to be chosen. However, it still must be allowed or the PP may fail to setup at all if the other slots are unable to setup for some reason, like interference.

Once all the subscribed handsets are involved in active connections the requirement to maintain a dummy no longer exists so the last traffic bearer to be set up can switch to the 19 element sequence when the quality of the 19 element sequence has reached the threshold. As soon as one handset releases its traffic bearer, if there is no dummy bearer active, the FP shall switch the traffic bearer to use the 95 element sequence during the partial release timer timeout period (for details of this timer see [5]). Then the FP shall set up a dummy bearer on the vacated slot using the 95 element sequence. The PP that has recently released switches its traffic bearer to a synchronisation bearer using the 95 element sequence, this transition shall occur seamlessly.

### 2.4.8 Debug Functionality

In order to support the implementation and testing of the D69 software some debug features shall be included in the project. This will be conditionally compiled using the switch:

```
#define     HOPPING_FREQUENCY_TEST
```

In the debug mode each call to the Hopping Engine to provide a frequency for the LMAC will be monitored and stored so that checks can be made on each end of a link. The data structure shall be defined as:

```
typedef
struct
{
u32 u32_Frame;
u8 u8_Slot;
u8 u8_SequenceElement;
u8 u8_Frequency;
u8 u8_Is19Sequence;
}st_HoppingFrequencyMonitor;
```

Some debug code will also be included in the generation of the quality messages sent to the Quality Monitor software to allow the injection of errors so that the response to known errors can be tested. This code will provide a band of frequencies to interfere with to simulate some static interferers.

# 3 Task Description

This section describes the D69 software blocks, their interfaces and their data, and the modifications necessary to existing software. All interfaces are implemented as:

- procedure call interfaces
- register interfaces
- operating system message interfaces

The amount of data passed is minimised through the use of shared data where necessary. This shared data is also described in this section using the C declarations as defined below, these declarations are placed in the general header file `cg0type.h`.

```
typedef unsigned char u8
typedef unsigned short u16
typedef unsigned int u32
```

## 3.1 Fixed Part

### 3.1.1 Hopping Engine (FP_HE)

This block performs four different functions;

- generating the hopping sequence for a given system identity
- generating individual frequencies to use as substitution frequencies for hopping sequences for a given system identity
- providing the frequency for each bearer to use for a given slot and frame
- reverting the sequences to use to their default values

#### 3.1.1.1 Hopping Sequence Generation

The FP_HE shall generate two sequences, a 95 element sequence and a 19 element sequence. This is achieved by a function call made from the HMAC.
Declaration:

```
void
p_hm17_GenerateHoppingSequence( u32 u32_Identity,
                                u8 u8_IsSubscriptionMode );
```

Where the parameters are defined as follows:

`u32 u32_Identity` - the system identity for which the sequence must be generated

`u8 u8_IsSubscriptionMode` - the type of sequence to generate.

Currently this function is called twice by the HMAC the difference between the two calls is in the value for the `u8_IsSubscription` parameter. The first call is to generate the two sequences used by the system, the second call is for generating a subscription sequence which is not required in the current design of the WDCT software, and so shall be ignored.

When called this function shall generate the two sequences using the algorithm defined in [4] and store them in global data arrays, one 95 elements long and one 19 elements long as a reference sequence. The 19 element long sequence will be copied into 8 further arrays with the elements offset from the reference sequence, the offset equal to the slot number, the index into the array of hopping sequences also equal to the slot number, one sequence for each slot. The reference 19 element sequence (array index 8) is required for when the slot sequences must be returned to the original slot sequence after a period of substitution.

These arrays are defined as:

```
u8 G_u8_hm07_HoppingSequenceLong[ 95 ];
u8 G_u8_hm07_HoppingSequenceShort[ 9 ][ 19 ];
```

This function will also initialise a global array of registers that indicates which sequence is in use for which slot.

Declaration:

```
u8 G_u8_hm07_ActiveHoppingSequence[ 8 ];
```

There is one register per slot, each register is defined as:

```
0  \
1  |
2  |
3  > PSCNToSwitchAt
4  |
5  |
6  /
7  If clear use 95 element sequence, otherwise 19 element sequence
```

Where `PSCNToSwitchAt` is valid if less than or equal to 94 (0x5e) and indicates the value of the PSCN counter at which to change the sequence. When set to 127 (0x7f) it indicates that no switch shall occur. The value of bit 7 determines which sequence is in operation or which shall be the new one if the value of `PSCNToSwitchAt` is less than 95 (0x5f).

| Register value | Alternative | Action |
|---|---|---|
| 0x00 - 0x5e | 0 - 94 | Replace 95 sequence with 19 sequence at PSCN = `PSCNToSwitchAt`[a] |
| 0x5f - 0x7e | 95 - 126 | Reserved |
| 0x7f | 127 | Use 95 element sequence |
| 0x80 - 0xde | 128 - 222 | Replace 19 sequence with 95 sequence at PSCN = `PSCNToSwitchAt`[b] |
| 0xdf - 0xfe | 223 - 254 | Reserved |
| 0xff | 255 | Use 19 element sequence |

a. When the PSCN counter reaches this value the register will be loaded with 0xff to indicate that the switch has occurred and that the 19 element sequence should now be used.
b. When the PSCN counter reaches this value the register will be loaded with 0x7f to indicate that the switch has occurred and that the 95 element sequence should now be used.

The initialisation value for each register is 127 (0x7f).

This function also initialises other registers, detailed below, these are initialised to 0.

The first is a register that ensures that the FP_QM and the FP_SUBS processes are synchronised.

The definition of this register is:

`u8 G_u8_hm07_RequestsActive`

The layout of this register is defined as:

| | |
|---|---|
| 0 | CONNECTION_0_BUSY |
| 1 | CONNECTION_1_BUSY |
| 2 | CONNECTION_2_BUSY |
| 3 | CONNECTION_3_BUSY |
| 4 | \ |
| 5 | > Reserved |
| 6 | \| |
| 7 | / |

The FP_QM shall only send a message to the FP_SUBS if the relevant BUSY flag is reset. When a message is sent to the FP_SUBS process the FP_QM shall set the relevant BUSY bit.

There is also a control register that controls the switching of the sequences in use. This is used for two purposes:

- prevent a bearer that was being used as the dummy bearer being switched to the 19 element sequence when it becomes a traffic bearer before the position of the new dummy bearer has been broadcast 4 times
- start a switch of a traffic bearer from the 19 element sequence to the 95 element sequence just before the traffic bearer is released if it will become the dummy bearer.

This register is defined as:

`u8 G_u8_hm07_ControlSwitching`

The layout of this register is defined as:

| | |
|---|---|
| 0 | CONNECTION_0_USE_LONG |
| 1 | CONNECTION_1_USE_LONG |
| 2 | CONNECTION_2_USE_LONG |
| 3 | CONNECTION_3_USE_LONG |

| | |
|---|---|
| 4 | CONNECTION_0_STAY_LONG |
| 5 | CONNECTION_1_STAY_LONG |
| 6 | CONNECTION_2_STAY_LONG |
| 7 | CONNECTION_3_STAY_LONG |

The higher layers access this register via functions these are defined as:

```
void p_hm17_SetUseLong( u8 u8_Connection );
void p_hm17_ClearUseLong( u8 u8_Connection );
void p_hm17_SetStayLong( u8 u8_Connection );
void p_hm17_ClearStayLong( u8 u8_Connection );
```

Where the parameter in each function is the relevant connection which is used to index into the `G_u8_hm07_ControlSwitching` register.

The bits are used as follows:

When the higher layers want to release a traffic bearer a timer is started, known as the partial release timer, (LCE.02 link maintain timer) see [5], when this timer is started the higher layers also set the `USE_LONG` bit for the relevant connection, by making a call to the `p_hm17_SetUseLong` function. If there is no bearer already using the 95 element sequence, then a switch of sequences is started to change from the 19 element sequence to the 95 element sequence. If during the partial release timer the higher layers decide they want to continue with the traffic bearer, the `USE_LONG` bit shall be cleared, by making a call to the `p_hm17_ClearUseLong` function. Then the usual procedure will apply and switch the sequence in use to the 19 element sequence again. This enables a controlled switch back to a dummy bearer using the 95 element sequence for the first handset to release from a situation where 4 handsets are in use.

When a connection is started on the existing dummy bearer and there are other slots still available the `STAY_LONG` bit is set for that connection, by making a call to the `p_hm17_SetStayLong` function. As a result of setting up on the dummy the HMAC will start a new dummy bearer and start to send the details of the new dummy position in the broadcast messages, see [2]. Once the new dummy position has been sent 4 times the HMAC will clear the `STAY_LONG` bit for the connection setup on the old dummy bearer, by making a call to the `p_hm17_ClearStayLong` function. This allows the new traffic bearer to switch to use the 19 element sequence following the usual procedure.

### 3.1.1.2 Substitution Frequency Generation

The substitution frequency is only ever generated for the 19 element sequence for a particular slot, the current sequence for that slot shall be the starting point for each substitution frequency generated. This is achieved using the following function which uses the algorithm defined in [4]:

```
u8
p_hm17_GenerateNewFrequency( u32 u32_Identity,
                             u8 u8_SequenceElement,
                                        u8 u8_Slot )
```

Where the parameters are defined as follows:

`u32 u32_Identity` - the system identity for which the frequency must be generated
`u8_SequenceElement` - the position in the sequence that needs substituting.
`u8_Slot` - the slot for which the element needs replacing.

The value returned is a frequency suitable for using in the sequence at the position given, based on the current sequence and the `u32_Identity` value. The function always returns a frequency. The algorithm to provide this is defined in [4].

#### 3.1.1.3 Hopping Sequence Element Retrieval

This function obtains the frequency to use on a particular slot for a particular sequence element. The slot determines which sequence to use and the sequence element determines which element of the sequence to use, this is achieved using the function call defined as follows;

```
u8
p_hm17_FindCarrier( u8 u8_SequenceElement,
                    u8 u8_Slot,
                    u8 u8_IsDBDBearer );
```

Where `u8_IsDBDBearer` is currently unused.

This function checks the `G_u8_hm07_ActiveHoppingSequence` array indexed by the `u8_Slot` parameter to find which type of sequence is in use in the slot for which the call has been mode. If the sequence in use is the 95 element sequence then the value returned is:

```
G_u8_hm07_HoppingSequenceLong[ ( u8_SequenceElement + u8_Slot ) % 95 ]
```

If the sequence in use is the 19 element sequence the value returned is:

```
G_u8_hm07_HoppingSequenceShort[ u8_Slot ][ u8_SequenceElement % 19 ]
```

When determining which sequence to use, if the `u8_SequenceElement` value is equal to the `PSCNToSwitchAt` value and the 95 element list is in use, then the sequence to use is updated to the 19 element sequence, and the `G_u8_hm07_ActiveHoppingSequence[ u8_Slot ]` is set to 0xff, to indicate that for subsequent accesses the 19 element sequence shall be used.

The actual frequency data in the sequences, and the selection of the 95 element or 19 element sequence is maintained by the substitution block.

#### 3.1.1.4 Reverting Hopping Sequences

This function forces the hopping sequence to use on a particular slot pair to be the 95 element sequence and reloads the 19 element sequences with their default values. This is achieved using the function call defined below:

```
void p_hm17_RevertSequences( u8 u8_Slot );
```

Where `u8_Slot` indicates for which slot pair to modify the sequences.

### 3.1.2 Quality Monitor (FP_QM)

The Quality Monitor performs four different functions:

- maintain a measure of quality for all the 95 frequencies possible in the WDCT band
- check for sequences that can be switched from 95 to 19 element sequences
- check the control of the switching of sequences
- check for elements of 19 element sequences that need to be substituted

#### 3.1.2.1 Maintain Quality Measure

Data are passed to this function by all Rx processes operating as part of a traffic bearer. Each of these LMAC processes calls the following function via the interface defined in D60 (see [3]):

```
void
p_hm17_HPQualityInfo( u8 u8_SequenceElement,
                      u8 u8_Slot,
                      u8 u8_UplinkQuality,
                      u8 u8_DownlinkQuality )
```

The `u8_Slot` indicates the Rx process from which the call originated and from this the two hopping sequences that this report has been generated for shall be determined (one for Rx and one for Tx). From these the actual frequencies to which the quality data refers is indicated by the `u8_SequenceElement` parameter. Then the quality information is stored for the actual frequencies. Each time this function is called, two counters shall be modified, one for the Rx and one for the Tx direction.

The array to store the frequency quality values is defined as:

```
u8 G_u8_hm07_QualityMeasure[ 95 ];
```

Each entry in this array is defined as:

| Bits | 7-4 | 3-0 |
|---|---|---|
| Meaning | Rx Quality Measure | Tx Quality Measure |

Where the quality measures are counters, that operate between a worst possible limit and a best possible limit. Each time an error occurs on a particular direction of transmission on a particular frequency the relevant counter is made worse by the error value. Similarly each time a good communication was achieved the relevant counter is made better by the good value. If either limit is reached, the value is clamped at that limit. There are two further values set for these counters, these are the thresholds for substitution and for usability.

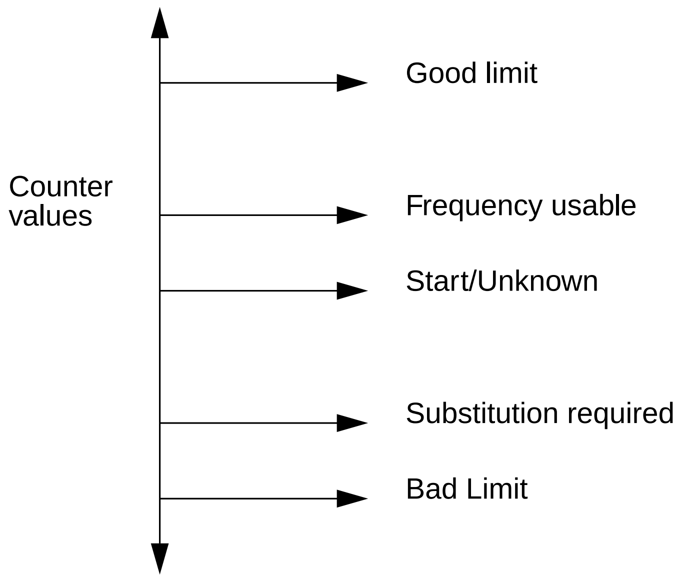


**Figure 3.1** Quality Counter

Each time a function call is made where it is detected that the complete sequence has been updated, before making any changes for the data in the function call, all counters are moved back towards the start value by HM17_QUALITY_DILUTION. If the counters reach the start value during this operation, they are clamped at that value.

The counter value definitions are as follows:

```
HM17_START_VALUE                  0x07
HM17 ERROR_DECREMENT              0x03
HM17_GOOD_INCREMENT               0x03
HM17_SUBSTITUTION_THRESHOLD       0x03
HM17_USABLE_THRESHOLD             0x08
HM17_BAD_LIMIT                    0x01
HM17_GOOD_LIMIT                   0x0b
HM17_QUALITY_DILUTION             0x01
```

These values are based on the algorithm defined in [4], the changes made are to enable the detection of the usable frequencies as well as the bad ones.

#### 3.1.2.2 Checking Quality Data

The Quality Monitor also checks the data in the quality measure for certain criteria and if any action is required messages are sent to the Substitution block of the D69 software. The communication between these two parts of the D69 software is synchronised using a control register to which they both have access, see `G_u8_hm07_RequestsActive` in section 3.1.1.1 starting on page 23. This synchronisation is necessary so that no extra RAM is required for queues of pending requests.

When a message is sent to the Substitution block of the D69 software the `BUSY` bit for the connection for which the message is being sent is set in the `G_u8_hm07_RequestsActive`, register, this bit is cleared by the Substitution block when the action to which it relates has been completed.

If the `BUSY` bit is clear then the quality data and the control bit shall be checked to see if any action is required. There are three checks to perform:

- check if sequence needs to be switched from 95 element to 19 element sequence
- check if sequence to use is forced to 95 and current sequence is 19 element sequence
- check for any frequency substitutions required

If a threshold level of the number of frequencies in a sequence for any of the slots to which the call to the FP_QM relates, are better or equal to the `HM17_MIN_NUMBER_OF_GOOD_FREQUENCIES` threshold, and the `STAY_LONG` bit is reset then a `SWITCH_SEQUENCE_REQ` message shall be sent for all slots meeting the criteria. This message goes to the substitution block of the D69 software which controls the switch from the 95 element sequence to the 19 element sequence.

If a threshold level of the number of frequencies in a sequence for any of the slots to which the call to the FP_QM relates, are better or equal to the `HM17_MIN_NUMBER_OF_GOOD_FREQUENCIES` threshold, and the `STAY_LONG` bit is set then a `SWITCH_SEQUENCE_REQ` message shall be sent for the Rx slot only. (If only the Tx slot meets the criteria no message is sent.) This message goes to the substitution block of the D69 software which controls the switch from the 95 element sequence to the 19 element sequence.

The threshold value definition is as follows:

```
HM17_MIN_NUMBER_OF_GOOD_FREQUENCIES             19
```

If the control register for this connection has its `USE_LONG` bit set and the current sequence in use on the Tx bearer of the connection is the 19 element sequence then a `SWITCH_SEQUENCE_REQ` message shall be sent for the Tx slot. This message goes to the substitution block of the D69 software which controls the switch from the 19 element sequence to the 95 element sequence.

If a switch of a sequence is not required then the check can be made if any substitution is required. If a counter value, within the `G_u8_hm07_QualityMeasure` array, for a frequency that is in the 19 element sequences for any of the slots to which this function call relates, reaches the substitution required threshold then a `SUBSTITUTION_REQ` message is sent to the Substitution block of the D69 software. If the #define HM00_PROTECTED_HT_SUPPORTED is not set then 2 substitutions can be requested, but if it is set then only one can be requested.

### 3.1.3 Substitution (FP_SUBS)

This part of the D69 software shall be realised as a finite state machine running as a true OS process. This is because some of the processing done by this block can be time consuming and so is not suitable for running as a procedure called from an interrupt service routine. Note the details below are shown for one instance of the process, there shall be one instance per possible connection in total, i.e. in a 2 handset system there shall be 2 instances, in a 4 handset system there shall be 4. Note that each instance is related to a connection instance of the MBC which can use two traffic bearers for when DBD mode is active.

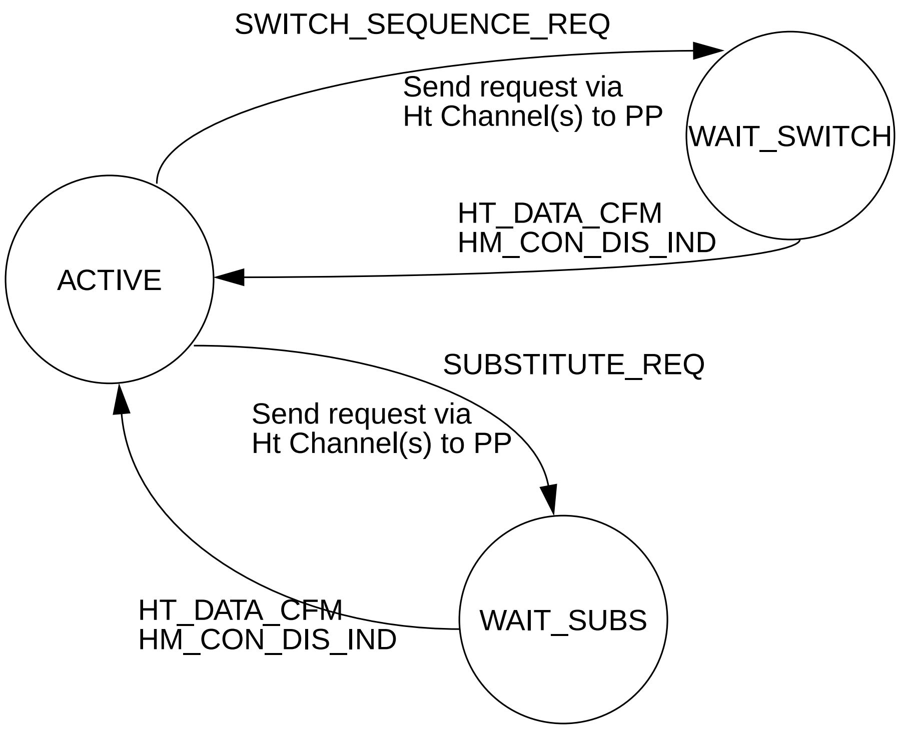


**Figure 3.2** States of Substitution Process

| State | Description |
|---|---|
| ACTIVE | This is the start up state of the process. `SUBSTITUTE_REQ` messages received in this state cause the process to call the Hopping Engine to obtain new frequencies for the sequences requested. Once obtained a request is sent to the PP via the Ht Channel and the process goes to the WAIT_SUBS state. `SWITCH_SEQUENCE_REQ` messages received in this state cause the process to send a request to the PP via the Ht Channel and the process goes to the WAIT_SWITCH state. |

| State | Description |
|---|---|
| WAIT_SUBS | In this state a request has been made to the Ht Channel to send a SUBSTITUTE_REQ message and the process is awaiting a response. HT_DATA_CFM messages received in this state cause the process to clear the relevant busy flag in the synchronization register, to use the data from the Ht Channel message buffer that was transmitted to update the relevant 19 element sequences with the new frequencies and the process returns to the ACTIVE state. HM_CON_DIS_IND messages received in this state cause the process to clear the relevant busy flag in the synchronization register and the process returns to the ACTIVE state[a]. |
| WAIT_SWITCH | In this state a request has been made to the Ht Channel to send a SWITCH_SEQUENCE_REQ message and the process is awaiting a response. HT_DATA_CFM messages received in this state cause the process to clear the relevant busy flag in the synchronization register, to use the data in the Ht Channel message buffer that was transmitted to set the relevant element of the G_u8_hm07_ActiveHoppingSequence register so that at the requested value of the PSCN counter the sequence shall be switched from the 95 to the 19 element sequence or vice versa and the process returns to the ACTIVE state. HM_CON_DIS_IND messages received in this state cause the process to clear the relevant busy flag in the synchronization register and the process returns to the ACTIVE state.[a] |

a. When the HM_CON_DIS_IND message is sent, the LMAC has already modified the hopping sequence to use to the long sequence, and reverted all the relevant short hopping sequences back to their original values. It does this by making a call to the function p_lm19_BearerRelease(), for details see section 3.1.4.2 starting on page 36.

#### 3.1.3.1 FP_SUBS Process Details

This process receives messages from the Quality Monitor block and the MAC and controls the Ht Channel.

| Message | Message Source | Info Field Contents | Shared Data |
|---|---|---|---|
| SUBSTITUTION_REQ | Quality Monitor | Slot1 + Slot2 + SequenceElement1 + SequenceElement2 | None |
| SWITCH_SEQUENCE_REQ | Quality Monitor | Slot1 + Slot2 + PSCNToSwitchAt | None |
| HT_DATA_CFM | MAC | Connection ID + error flag | None |
| HM_CON_DIS_IND | MAC | Connection ID | None |

NOTE. The SUBSTITUTION_REQ and the SWITCH_SEQUENCE_REQ message can have data associated with two slots or only one slot. For the SUBSTITUTION_REQ the maximum allowed will be conditionally compilable. The single slot version is required to be able to include some identity information in the Ht Channel message so that the source of the Ht Channel reception can be at least partially checked. Testing of previous versions of the WDCT software in an interfered environment has revealed a system feature where a good reception can be made but the source is not the correct one. For this reason some extra protection is being included. The double slot version is being left in the design as this may have to be used anyway if the throughput of the Ht Channel is insufficient for substitution or if a different scheme for protecting the reception against incorrect source is implemented.

**SUBSTITUTION_REQ**

Requests the Substitution software to obtain one or two new frequencies to use at the SequenceElement**n** entries in the sequences indicated by Slot**n,** all values coming from the info field contents. If both Slot1 and Slot2 are used, they shall only be set to the slots of a single traffic bearer (i.e. they shall be both for the Rx, both for the Tx or one for the Rx and one for the Tx slot of the same traffic bearer). As a result of this message the `p_hm17_GenerateNewFrequency()` function is called once or twice, see section 3.1.1.2 starting on page 26, once the new frequencies have been obtained the information is passed to the PP via the Ht Channel.

The information field is coded as follows:

| | |
|---|---|
| 0 | \ |
| 1 | > Slot1 |
| 2 | / |
| 3 | \ |
| 4 | > Slot2 |
| 5 | / |
| 6 | \ |
| 7 | \| |
| 8 | > SequenceElement1 |
| 9 | \| |
| 10 | / |
| 11 | \ |
| 12 | \| |
| 13 | > SequenceElement2 |
| 14 | \| |
| 15 | / |

Where `Slot1` and `SequenceElement1` shall always be valid, but if `SequenceElement2` is greater than or equal to 19 only one substitution has been requested. If the `#define HM00_PROTECTED_HT_SUPPORTED` is set then only `Slot1` can be used, the fields for `Slot2` and `SequenceElement2` are ignored.

The Ht Channel message is defined as follows:

If HM00_PROTECTED_HT_SUPPORTED is not set.

| Bits | 39-32 | 31-24 | 23-16 | 15-8 | 7-4 | 3-0 |
|---|---|---|---|---|---|---|
| Meaning | Frequency2 | Element2 | Frequency1 | Element1 | Slot2 | Slot1 |

If HM00_PROTECTED_HT_SUPPORTED is set.

| Bits | 39-24 | 23-16 | 15-8 | 7-4 | 3-0 |
|---|---|---|---|---|---|
| Meaning | 16 LSBs of RFPI | Frequency1 | Element1 | 0x0e | Slot1 |

Where `Slot1` and `Slot2` indicate for which slot the sequence is to be modified (0-7), if both are set to an invalid slot number (8-15) then the message is not a `SUBSTITUTION_REQ` message. If the value of `Slot`**n** is valid, then for the 19 element sequence for `Slot`**n** the sequence element `Element`**n** shall be replaced with the frequency `Frequency`**n. I**f both `Slot1` and `Slot2` are valid they shall only be set to the slots of a single traffic bearer (i.e. they shall be both for the Rx, both for the Tx or one for the Rx and one for the Tx slot of the same traffic bearer).

Setting bits 4-7 to `0x0e` indicates that the message is in the protected format. If the unprotected format is used but with information for only one slot, then the invalid slot value used for `Slot2` field shall not be set to `0x0e`.

**SWITCH_SEQUENCE_REQ**

Requests the Substitution software to exchange the hopping sequence in use for the 19 element sequence or the 95 element sequence for one or two slots as indicated in the info field. A call is made to the LMAC to obtain the current value for the PSCN counter, see section 3.1.4 starting on page 36, so that this value plus `HM17_TIME_FOR_SWITCH` can be issued as the time to update the sequences. This information is then sent to the PP via the Ht Channel.

The information field is coded as follows:

| | |
|---|---|
| 0 | \ |
| 1 | > Slot1 |
| 2 | \| |
| 3 | / |
| 4 | \ |
| 5 | > Slot2 |
| 6 | \| |
| 7 | / |
| 8 | \ |
| 9 | \| |
| 10 | \| |
| 11 | > Reserved |
| 12 | \| |
| 13 | \| |
| 14 | / |
| 15 | Switch95To19 |

If the value of `Slotn` is valid (0-7) then send the data over the Ht Channel so that the exchange of sequence for `Slotn` will occur at the PSCN counter value (PSCN counter value when Ht Channel request is made + `HM18_TIME_FOR_SWITCH`), if both `Slot1` and `Slot2` are invalid ignore the message, if both `Slot1` and `Slot2` are valid they shall only be set to both the slots of a single traffic bearer (i.e. they shall be one for the Rx and one for the Tx slot of the same traffic bearer). If `Switch95To19` is set then switch the sequences from the 95 to the 19 element sequence, if reset switch the sequences from the 19 to the 95 element sequence.

The time delay for switching the sequences from the time the request is made in frames is defined as:

```
#define HM18_TIME_FOR_SWITCH      90
```

The Ht Channel message is defined as follows:

| Bits | 39-24 | 23 | 22-16 | 15-12 | 11-8 | 7-4 | 3-0 |
|---|---|---|---|---|---|---|---|
| Meaning | 16 LSBs of RFPI | Switch95To19 | PSCNToSwitchAt | Slot2 | Slot1 | 0xe/0xf | 0xf |

Where bits 0-7 indicate that this is not a `SUBSTITUTION_REQ` message but a `SWITCH_SEQUENCE_REQ` message. If bits 4-7 are set to 15, then the field containing the RFPI bits is not valid. If the `#define HM00_PROTECTED_HT_SUPPORTED` is set then the RFPI field must be used. `Slot1` and `Slot2` indicate for which slot the sequence is to be switched (0-7), if both are set to an invalid slot number (8-15) then the message is ignored. If the value of `Slotn` is valid, then the sequences shall be exchanged for `Slotn` when the PSCN counter equals the `PSCNToSwitchAt`, if `Switch95To19` is set, then the sequences shall be exchanged from the 95 element to the 19 element sequences, if reset then the 19 element sequence shall be exchanged with the 95 element sequence.

### HT_DATA_CFM

This message (defined in D60, see [3]) indicates that the last issued request over the Ht Channel of the connection indicated in the info part of the message has been completed. This shall result in the busy flag for the relevant connection in the `G_u8_hm07_RequestsActive` register being cleared. This message is sent with an error flag, this flag is set when the Ht Channel has been unable to complete the transmission of the message within a time set when the request is first issued. The time set for all requests is infinite so the error flag will never be set.

### HM_CON_DIS_IND

This message (defined in D60, see [3]) indicates that the connection identified by the info part of the message has been disconnected. This shall result in the sequences to be used by the slot pair associated with this connection being unconditionally reset to use the 95 element list in the `G_u8_hm07_ActiveHoppingSequence` array, and if any Ht Channel activity is in progress it is aborted so the busy flag for the relevant connection in the `G_u8_hm07_RequestsActive` register shall be cleared.

### 3.1.4 Changes Required in LMAC

These changes are all included in the LMAC when the compiler switch detailed below is set:
`#define LM00_SWITCH_SEQUENCE_SUPPORTED`

#### 3.1.4.1 Current PSCN Value

In order to find out the current value of the PSCN counter a new function must be added to the LMAC functionality that returns this value, this function is defined as:
`u8 p_lm10_CurrentPSCN( void );`

#### 3.1.4.2 Reverting Hopping Sequences

When a traffic bearer is taken down the slots that it used must be instantly reverted to use the 95 element hopping sequence so that a setup attempt from a handset can be seen in the following frame. At the same time the 19 element sequences that may have been in use are reloaded with their default values. This is achieved by adding a new function to the LMAC functionality, this function is defined as:
`void p_lm19_BearerRelease( u8 u8_Slot );`

Where the `u8_Slot` parameter passes the slot pair that was in use by the bearer.

This function will be called by the FP LMAC receive process whenever a traffic bearer is released or converted into a dummy bearer.

This function will then make a call to the `p_hm17_RevertSequences()` see section 3.1.1.4 starting on page 27.

### 3.1.5 File Names

The files required for the implementation of this functionality are:

| Filename | Used for |
|---|---|
| bhm07use.h | FP_HE, FP_QM |
| bhm17use.c | |
| bhm08fsh.h | FP_SUBS |
| bhm18fsv.c | |
| bhm28fss.sdl | |

## 3.2 Portable Part

### 3.2.1 Hopping Engine (PP_HE)

This block performs three different functions;

- generating the hopping sequence for a given system identity
- providing the frequency for each bearer to use for a given slot and frame
- reverting the sequences to use to default values

#### 3.2.1.1 Hopping Sequence Generation

The Hopping Engine shall generate two sequences, a 95 element sequence and a 19 element sequence. This is achieved by a function call made from the HMAC.
Declaration:

```
void
p_hm17_GenerateHoppingSequence( u32 u32_Identity,
                                 u8 u8_IsSubscriptionMode );
```

Where the parameters are defined as follows:

`u32 u32_Identity` - the system identity for which the sequence must be generated

`u8 u8_IsSubscriptionMode` - the type of sequence to generate.

Currently this function is called twice by the HMAC the difference between the two calls is in the value for the `u8_IsSubscription` parameter. The first call is to generate the two sequences used by the system, the second call is for generating a subscription sequence which is not required in the current design of the WDCT software, and so shall be ignored.

When called this function shall generate the two sequences using the algorithm defined in [4] and store them in global data arrays, one 95 elements long and one 19 elements long as a reference sequence. The 19 element long sequence will be copied into 8 further arrays with the elements offset from the reference sequence, the offset equal to the slot number, the index into the array of hopping sequences also equal to the slot number, one sequence for each slot. The reference 19 element sequence (array index 8) is required for when the slot sequences must be returned to the original slot sequence after a period of substitution.

These arrays are defined as:

```
u8 G_u8_hm07_HoppingSequenceLong[ 95 ];
```

```
u8 G_u8_hm07_HoppingSequenceShort[ 9 ][ 19 ];
```

This function will also initialise a global array of registers that indicates which sequence is in use for which slot.

Declaration:

```
u8 G_u8_hm07_ActiveHoppingSequence[ 8 ];
```

There is one register per slot, each register is defined as:

| | |
|---|---|
| 0 | \ |
| 1 | \| |
| 2 | \| |
| 3 | > `PSCNToSwitchAt` |
| 4 | \| |
| 5 | \| |
| 6 | / |
| 7 | If clear use 95 element sequence, otherwise 19 element sequence |

Where `PSCNToSwitchAt` is valid if less than or equal to 94 (0x5e) and indicates the value of the PSCN counter at which to change the sequence. When set to 127 (0x7f) it indicates that no switch shall occur. The value of bit 7 determines which sequence is in operation or which shall be the new one if the value of `PSCNToSwitchAt` is less than 95 (0x5f).

| Register value | Alternative | Action |
|---|---|---|
| 0x00 - 0x5e | 0 - 94 | Replace 95 sequence with 19 sequence at PSCN = register value[a] |
| 0x5f - 0x7e | 95 - 126 | Reserved |
| 0x7f | 127 | Use 95 element sequence |
| 0x80 - 0xde | 128 - 222 | Replace 19 sequence with 95 sequence at PSCN = register value[b] |
| 0xdf - 0xfe | 223 - 254 | Reserved |
| 0xff | 255 | Use 19 element sequence |

a. When the PSCN counter reaches this value the register will be loaded with 0xff to indicate that the switch has occurred and that the 19 element sequence should now be used.
b. When the PSCN counter reaches this value the register will be loaded with 0x7f to indicate that the switch has occurred and that the 95 element sequence should now be used.

The initialisation value for each register is 127 (0x7f).

#### 3.2.1.2 Hopping Sequence Element Retrieval

This function obtains the frequency to use on a particular slot for a particular sequence element. The slot determines which sequence to use and the sequence element determines which element of the sequence to use, this is achieved using the function call defined as follows;

```
u8
p_hm17_FindCarrier( u8 u8_SequenceElement,
                    u8 u8_Slot,
                    u8 u8_IsDBDBearer );
```

Where `u8_IsDBDBearer` is currently unused.

This function checks the `G_u8_hm07_ActiveHoppingSequence` array indexed by the `u8_Slot` parameter to find which type of sequence is in use in the slot for which the call has been mode. If the sequence in use is the 95 element sequence then the value returned is:

```
G_u8_hm07_HoppingSequenceLong[ ( u8_SequenceElement + u8_Slot ) % 95 ]
```

If the sequence in use is the 19 element sequence the value returned is:

```
G_u8_hm07_HoppingSequenceShort[ u8_Slot ][ u8_SequenceElement % 19 ]
```

When determining which sequence to use, if the `u8_SequenceElement` value is equal to the `PSCNToSwitchAt` value and the 95 element list is in use, then the sequence to use is updated to the 19 element sequence, and the `G_u8_hm07_ActiveHoppingSequence[ u8_Slot ]` is set to 0xff, to indicate that for subsequent accesses the 19 element sequence shall be used. Similarly if the `u8_SequenceElement` value is equal to the `PSCNToSwitchAt` value and the

19 element list is in use, then the sequence to use is updated to the 95 element sequence, and the `G_u8_hm07_ActiveHoppingSequence[ u8_Slot ]` is set to 0x7f, to indicate that for subsequent accesses the 95 element sequence shall be used.

The actual frequency data in the sequences, and the selection of the 95 element or 19 element sequence is maintained by the substitution block.

#### 3.2.1.3 Reverting Hopping Sequences

This function forces the hopping sequence to use on a particular slot pair to be the 95 element sequence and reloads the 19 element sequences with their default values. This is achieved using the function call defined below:

```
void p_hm17_RevertSequences( u8 u8_Slot );
```

Where `u8_Slot` indicates for which slot pair to modify the sequences.

### 3.2.2 Quality Monitor (PP_QM)

The Quality Monitor performs two different functions:

- maintain a measure of quality for the last 95 frames
- check the quality for a connection to see if DBD mode should be requested

#### 3.2.2.1 Maintain Quality Measure

Data are passed to this function by all Rx processes operating as part of a traffic bearer. Each of these LMAC processes calls the following function via the interface defined in D60 (see [3]):

```
void
p_hm17_HPQualityInfo( u8 u8_SequenceElement,
                      u8 u8_Slot,
                      u8 u8_UplinkQuality,
                      u8 u8_DownlinkQuality )
```

The `u8_Slot` indicates the Rx process from which the call originated and from this the two hopping sequences that this report has been generated for can be determined (one for Rx and one for Tx), and from these the actual frequencies to which the quality data refers is indicated by the `u8_SequenceElement` parameter. For the PP_QM process the only information that is stored is whether the link was good or bad in the frame indicated by the `u8_SequenceElement` parameter. Each time this function is called, one bit shall be modified in an array.

The array to store the frequency quality data is defined as:

```
u32 G_u32_hm07_QualityMeasure[ 3 ];
```

This array is defined as 3 words but is used as an array of bits, each bit from 0 - 94 in this array is defined as:

| Bit Value | 0 | 1 |
|---|---|---|
| Meaning | Quality OK | Quality Bad |

The quality measures merely record if there was an error in either the Rx or Tx direction in the frame indicated by the `u8_SequenceElement` parameter.

This array is cleared to indicate all quality OK at the start of each connection, this is achieved by the MAC making a call to the following function when it sends the `HM_CON_SETUP_IND` message, see [3].

```
void p_hm17_ResetQualityRecord( void );
```

Once the quality data has been updated the 95 bits are checked to see how many bits have been recorded as bad quality in the last 95 receptions. If the number exceeds a threshold value then DBD mode is requested via the register interface described in [1]. If there are no bits recorded as bad, then a counter is incremented to indicate how long the quality has been good. If this counter exceeds a theshold value then a request is made to take down DBD mode via the register interface described in [1], if DBD mode had not been activated this request will be ignored. This counter is reset to 0 if any bad quality is reported. Once the good quality threshold is reached it is clamped at that value.

The counter is defined as:

```
u16 G_u16_hm07_GoodQualityCounter;
```

The values for the counters are defined as:

```
#define HM17_DBD_ON_THRESHOLD              3
#define HM17_GOOD_QUALITY_THRESHOLD          475
```

### 3.2.3 Substitution (PP_SUBS)

Unlike the corresponding process in the FP this part of the D69 software shall not be realised as a true OS process. This is because this is a much simpler process than the FP_SUBS, as it merely responds to the requests from the FP, and also because the message from the MAC is placed in shared data that is used for every Ht Channel message received from the FP, so it must be responded to before the next message is received or it will be lost.

The MAC informs the PP_SUBS process that a message has arrived by calling the following function:

```
void p_hm17_HtRxIndication( u8 *pu8_Source );
```

The pointer parameter points to a 5 byte data buffer that contains the Ht Channel message as received by the PP this data can be one of two messages

- `SUBSTITUTION_REQ`
- `SWITCH_SEQUENCE_REQ`

These messages are constructed by the FP, for details see section 3.1.3.1 starting on page 32.

**SUBSTITUTION_REQ**

On receipt of this message the PP_SUBS process immediately updates the 19 element sequence(s) for the slot(s) requested at the sequence element(s) requested with the frequencies supplied in the message.

**SWITCH_SEQUENCE_REQ**

On receipt of this message the PP_SUBS process loads the relevant entries in the `G_u8_hm07_ActiveHoppingSequence` array with the value for the PSCN counter to switch at and the bit to indicate which sequence to switch to.

### 3.2.4 Changes Required in LMAC

These changes are all included in the LMAC when the compiler switch detailed below is set:
```
#define LM00_SWITCH_SEQUENCE_SUPPORTED
```

#### 3.2.4.1 Current PSCN Value

In order to find out the current value of the PSCN counter a new function must be added to the LMAC functionality that returns this value, this function is defined as:
```
u8 p_lm10_CurrentPSCN( void );
```

#### 3.2.4.2 Reverting Hopping Sequences

When a traffic bearer is taken down the slots that it used must be reverted to use the 95 element hopping sequence so that a sync bearer can be setup to find the base using the correct sequence. At the same time the 19 element sequences that may have been in use are reloaded with their default values. This is achieved by adding a new function to the LMAC functionality, this function is defined as:
```
void p_lm19_BearerRelease( u8 u8_Slot );
```
Where the `u8_Slot` parameter passes the slot pair that was in use by the bearer.

This function will be called by the PP LMAC receive process whenever a traffic bearer is released or converted into a synchronisation bearer.

This function will then make a call to the `p_hm17_RevertSequences()` see section 3.1.4.2 starting on page 36

### 3.2.5 File Names

The files required for the implementation of this functionality are:

| Filename | Used for |
|---|---|
| hhm07use.h | PP_HE, PP_QM, PP_SUBS |
| hhm17use.c | |