Technical Specification Group Services and System Aspects
Meeting #5, Kyongju, Korea, 11-13 October 1999

TSGS#5(99)353

Source:	**TSG-S4 Codec Working Group**
Title:	**3G TS 26.094 v. 2.0.0 - AMR speech codec; Voice Activity Detector (VAD)**
Document for:	**Approval**
Agenda Item:	**5.4.3**

3G TS 26.094 V.2.0.0 (1999-10)

Technical Specification

3rd Generation Partnership Project;
TSG-SA Codec Working Group
Mandatory Speech Codec speech processing functions
AMR speech codec; Voice Activity Detector (VAD)

Reference

TSG-SA4-W1

Keywords

Adaptive Multi-Rate, Mandatory speech codec

Contents

Intellectual Property Rights

Foreword

This Technical Specification has been produced by the 3GPP.

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of this TS, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 Indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the specification;

1 Scope

This document specifies two alternatives for the Voice Activity Detector (VAD) to be used in the Discontinuous Transmission (DTX) as described in [3]. Implementors of mobile station and infrastructure equipment conforming to the AMR specifications can choose which of the two VAD options to implement. There are no interoperability factors associated with this choice.

The requirements are mandatory on any VAD to be used either in User Equipment (UE) or Base Station Systems (BSS)s that utilize the AMR speech codec.

2 Normative References

This TS incorporates by dated and undated reference, provisions from other publications. These normative references are cited in the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this TS only when incorporated in it by amendment or revision. For undated references, the latest edition of the publication referred to applies.

[1] TS 26.73: "ANSI-C code for the Adaptive Multi Rate speech codec" .

[2] TS 26.90: "AMR Speech Codec Speech Transcoding Functions" .

[3] TS 26.93: "AMR Speech codec; Source Controlled Rate Operation".

[4] ITU, The International Telecommunications Union, Blue Book, Vol. III, Telephone Transmission Quality, IXth Plenary Assembly, Melbourne, 14-25 November, 1988, Recommendation G.711, Pulse code modulation (PCM) of voice frequencies.

3 Technical Description of VAD Option 1

3.1 Definitions, symbols and abbreviations

3.1.1 Definitions

For the purposes of this TS, the following definitions apply:

frame: Time interval of 20 ms corresponding to the time segmentation of the speech transcoder.

3.1.2 Symbols

For the purposes of this TS, the following symbols apply.

3.1.2.1 Variables

bckr_est[n]	background noise estimate
burst_count	counts length of a speech burst, used by VAD hangover addition
hang_count	hangover counter, used by VAD hangover addition
complex_hang_count	hangover counter, used by CAD hangover addition
complex_hang_timer	hangover initator, used fo Complex Activity Estimation
lagcount	pitch detection counter
level[n]	signal level
new_speech	pointer of the speech encoder, points a buffer containing last received samples of a speech frame [2]
noise_level	average level of the background noise estimate
oldlagcount	lagcount of the previous frame
pitch	flag indicating presence of a periodic signal
complex_warning	flag indicating the presence of a complex signal.

best_corr_hp	normalized and limited value from maximum HP filtered correlation vector
corr_hp	filtered best_corr_hp values
pow_sum	power of the input frame
s(i)	samples of the input framer
snr_sum	measure between input frame and noise estimate
stat_count	stationarity counter
stat_rat	measure indicating stationary
T_op[n]	open-loop lags [2]
t0	autocorrelation maxima calculated by the open-loop pitch analysis [2]
t1	signal power related to the autocorrelation maxima t0 [2]
tone	flag indicating the presence of a tone
vad_thr	VAD threshold
VAD_flag	boolean VAD flag
vadreg	intermediate VAD decision
complex_low	intermediate complex signal decisions
complex_high	intermediate complex signal decisions

3.1.2.2 Constants

ALPHA_UP1	constant for updating noise estimate (see subclause 3.3.5.2)
ALPHA_DOWN1	constant for updating noise estimate (see subclause 3.3.5.2)
ALPHA_UP2	constant for updating noise estimate (see subclause 3.3.5.2)
ALPHA_DOWN2	constant for updating noise estimate (see subclause 3.3.5.2)
ALPHA3	constant for updating noise estimate (see subclause 3.3.5.2)
ALPHA4	constant for updating average signal level (see subclause 3.3.5.2)
ALPHA5	constant for updating average signal level (see subclause 3.3.5.2)
BURST_LEN_HIGH_NOISE	constant for controlling VAD hangover addition (see subclause 3.3.5.1)
BURST_LEN_LOW_NOISE	constant for controlling VAD hangover addition (see subclause 3.3.5.1)
COEFF3	coefficient for the filter bank (see subclause 3.3.1)
COEFF5_1	coefficient for the filter bank (see subclause 3.3.1)

COEFF5_2	coefficient for the filter bank (see subclause 3.3.1)
HANG_LEN_HIGH_NOISE	constant for controlling VAD hangover addition (see subclause 3.3.5.1)
HANG_LEN_LOW_NOISE	constant for controlling VAD hangover addition (see subclause 3.3.5.2)
HANG_NOISE_THR	constant for controlling VAD hangover addition (see subclause 3.3.5.2)
L_FRAME	size of a speech frame, 160
L_NEXT	length for the lookahead of the speech encoder, 40
LTHRESH	threshold for pitch detection (see subclause 3.3.2)
NOISE_MAX	maximum value for noise estimate (see subclause 3.3.5.2)
NOISE_MIN	minimum value for noise estimate (see subclause 3.3.5.2)
NTHRESH	threshold for pitch detection (see subclause 3.3.2)
POW_PITCH_THR	threshold for pitch detection (see subclause 3.3.5)
POW_COMPLEX_THR	threshold for complex detection (see subclause 3.3.5)
STAT_COUNT	threshold for stationary detection (see subclause 3.3.5.2)
CAD_MIN_STAT_COUNT	minimum threshold after complex warning
STAT_THR	threshold for stationary detection (see subclause 3.3.5.2)
STAT_THR_LEVEL	threshold for stationary detection (see subclause 3.3.5.2)
TONE_THR	threshold for tone detection (see subclause 3.3.3)
VAD_P1	constant of computation for VAD threshold (see subclause 3.3.5.2)
VAD_POW_LOW	constant for controlling VAD hangover addition (see subclause 3.3.5.1)
VAD_SLOPE	constant of computation for VAD threshold (see subclause 3.3.5)
VAD_THR_HIGH	constant of computation for VAD threshold (see subclause 3.3.5)
CVAD_THRESH_ADAPT_HIGH	constant for updating complex_high
CVAD_THRESH_ADAPT_LOW	constant for updating complex_low
CVAD_THRESH_HANG	constant for updating complex_hang_timer
CVAD_HANG_LIMIT	constant for initiating complex_hang_count
CVAD_HANG_LENGTH	constant for resetting complex_hang_count

3.1.2.3 Functions

+	addition

-	subtraction
*****	multiplication
/	division
| x |	absolute value of x
AND	Boolean AND
OR	Boolean OR
$\sum_{n=a}^{b} x(n)$	$= x(a) + x(a+1) + \ldots + x(b-1) + x(b)$
MIN(x,y)	$= \begin{cases} x, x \le y \\ y, y < x \end{cases}$
MAX(x,y)	$= \begin{cases} x, x \ge y \\ y, y > x \end{cases}$

3.1.3 Abbreviations

ANSI	American National Standards Institute
DTX	Discontinuous Transmission
VAD	Voice Activity Detector
CAD	Complex Activity Detection
CNG	Comfort Noise Generation

3.2 General

The function of the VAD algorithm is to indicate whether each 20 ms frame contains signals that should be transmitted, i.e. speech, music or information tones. The output of the VAD algorithm is a Boolean flag (VAD_flag) indicating presence of such signals.

3.3 Functional description

The block diagram of the VAD algorithm is depicted in figure 1. The VAD algorithm uses parameters of the speech encoder to compute the Boolean VAD flag (VAD_flag). Samples of the Input frame (s(i)) are divided into sub-bands and level of the signal in each band (level[n]) is calculated. Input for the pitch detection function are open-loop lags (T_op[n]), which are calculated by open-loop pitch analysis of the speech encoder. The pitch detection function computes a flag (pitch) which indicates presence of pitch. Tone detection function calculates a flag (tone), which indicates presence of an information tone. Tones are detected based on pitch gain of the open-loop pitch analysis The pitch gain is estimated using autocorrelation values (t0 and t1) received from the pitch analysis. Complex Signal Detection function calculates a flag (complex_warning), which indicates presence of a correlated complex signal such as music. Correlate complex signals are detected based on analysis of the correlation vector available in the open-loop pitch analysis.The VAD decision function estimates background noise levels. Intermediate VAD decision is calculated based on the comparison of the background noise estimate and levels of the input frame (level[n]). Finally, the VAD flag is calculated by adding hangover to the intermediate VAD decision.

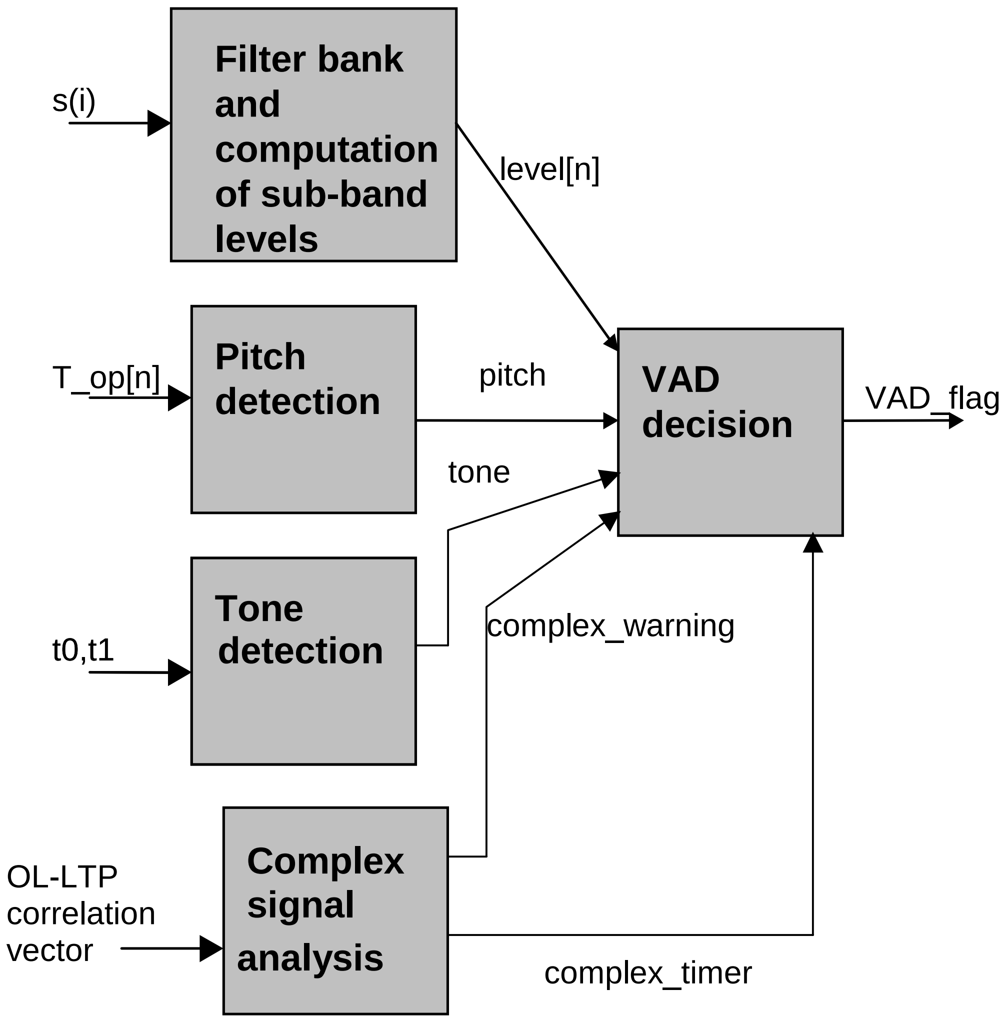

Figure 3.1. Simplified block diagram of the VAD algorithm: Option 1

3.3.1 Filter bank and computation of sub-band levels

The input signal is divided into frequency bands using a 9-band filter bank (figure 3.2). Cut-off frequencies for the filter bank are shown in table 3.1.

Table 3.1. Cut-off frequencies for the filter bank

Band number	Frequencies
1	0 - 250 Hz
2	250 - 500 Hz
3	500 - 750 Hz
4	750 - 1000 Hz
5	1000 - 1500 Hz
6	1500 - 2000 Hz
7	2000 - 2500 Hz
8	2500 - 3000 Hz
9	3000 - 4000 Hz

Input for the filter bank is the speech frame pointed by the new_speech pointer of the speech encoder [1]. Input values for the filter bank are scaled down by one bit. This ensures safe scaling, i.e. saturation can not occur during calculation of the filter bank.

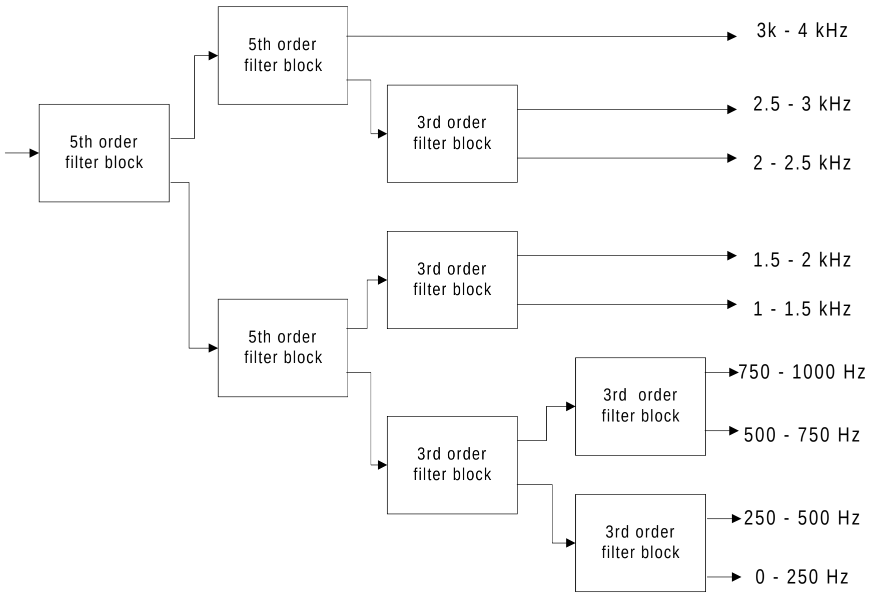

Figure 3.2. Filter bank

The filter bank consists of 5th and 3rd order filter blocks. Each filter block divides the input into high-pass and low-pass parts and decimates the sampling frequency by 2. The 5th order filter block is calculated as follows:

$$x_{lp}(i) = 0.5*(A_1(x(i-1)) + A_2(x(i))) \qquad (3.1a)$$

$$x_{hp}(i) = 0.5*(A_1(x(i-1)) - A_2(x(i))) \qquad (3.1b)$$

where

x(i)	input signal for a filter block
$x_{lp}(i)$	low-pass component
$x_{hp}(i)$	high-pass component

The 3rd order filter block is calculated as follows:

$$x_{lp}(i) = 0.5*(x(i) + A_3(x(i-1))) \qquad (3.2a)$$

$$x_{hp}(i) = 0.5*(x(i) - A_3(x(i-1))) \qquad (3.2b)$$

The filters $A_1()$, $A_2()$, and $A_3()$ are first order direct form all-pass filters, whose transfer function is given by:

$$A(z) = \frac{C + z^{-1}}{1 + C*z^{-1}}, \qquad (3.3)$$

where C is the filter coefficient.

Coefficients for the all-pass filters $A_1()$, $A_2()$, and $A_3()$ are COEFF5_1, COEFF5_2, and COEFF3, respectively.

Signal level is calculated at the ouput of the filter bank at each frequency band as follows:

$$level(n) = \sum_{i=START_n}^{END_n} |x_n(i)|, \qquad (3.4)$$

where:

n	index for the frequency band
$x_n(i)$	sample i at the output of the filter bank at frequency band n

$$START_n = \begin{cases} -2, & n \le 4 \\ -4, & 5 \le n \le 8 \\ -8, & n = 9 \end{cases}$$

$$END_n = \begin{cases} 9, & n \le 4 \\ 19, & 5 \le n \le 8 \\ 39, & n = 9 \end{cases}$$

Negative indices of $x_n(i)$ refer to the previous frame.

3.3.2 Pitch detection

The purpose of the pitch detection function is to detect vowel sounds and other periodic signals. The pitch detection is based on comparison of open-loop lags (T_op[n]), which are calculated by the speech encoder [2]. If the difference of consecutive open-loop lags (T_op[n]) is smaller than a threshold, lagcount is incremented. If the sum of the lagcounts of two consecutive frames is high enough, the pitch flag is set. For 5.15 and 4.75 kbit/s rates, only one open-loop lag is calculated, and therfore only the first lag-comparison is made every frame. The pitch flag is calculated as follows:

```
Lagcount = 0;
If ( | T_op[-1] - T_op[0] | < LTHRESH)
        Lagcount = Lagcount + 1
If ( | T_op[0] - T_op[1] |  < LTHRESH)
        Lagcount = Lagcount + 1
If (Lagcount + oldlagcount > NTHRESH)
        pitch = 1
else
        pitch = 0
oldlagcount = Lagcount
```

T_op[-1] refers to the open-loop lag of the previous frame.

3.3.3 Tone detection

Tone detection is used to detect information tones, since the pitch detection function can not always detect these signals. Also, other signals which contain very strong periodic component are detected, because it may sound annoying if these signals are replaced by comfort noise. If the open-loop pitch gain is higher than the constant TONE_THR, tone is detected and tone flag is set. The pitch gain can be tested by comparing variables t0 and t1 as follows:

```
if (t0 > TONE_THR * t1)
      tone = 1
```

The speech encoder calculates the pitch in three delay ranges, except for mode 10.2 kbit/s, where only one range is used. The above comparison is made once for each delay range and the tone flag should be set if the condition is true at least in one range. Otherwise, the tone flag should be set to zero.
The variables t0 and t1 are calculated by the open-loop pitch analysis of the speech encoder [2]. The variable t0 is autocorrelation maxima given by:

$$t0 = \sum_n s_w(n)s_w(n-k) \qquad (3.5)$$

The variable t1 is the signal power related to the autocorrelation maxima t0 at the delay value k:

$$t1 = \sum_n s_w^2(n-k) \qquad (3.6)$$

The open-loop pitch search and correspondingly the tone flag is computed twice in each frame, except for modes 5.15 kbit/s and 4.75 kbit/s, where it is computed only once.

3.3.4 Correlated Complex Signal Analysis (and detection)

Correlated complex signal detection is used to detect correlated signals in the highpass filtered weighted speech domain, since the pitch and tone detection functions can not always detect these signals. Signals which contain very strong correlation values in the high pass filtered domain are taken care of, because it may sound really annoying if these signals are replaced by comfort noise. If the statistics of the maximum normalized correlation value of a high pass filtered input signal indicates the presence of a correlated complex signal a flag ***complex_warning*** is set. To reduce complexity the high band correlation analysis is performed in a simplified manner by analysing the high pass filtered fullband correlation vector which is available from the OL-LTP analysis performed by the speech encoder at least once in each frame.
best_corr_hp$_m$ is the maximum normalized value of the high pass filtered correlation in the range 19-146 limited to be in the range [1.0, 0.0]. (Note that the *best_corr_hp* value is delayed one frame). The high pass filter is a simple first order filter with coefficients [1, -1] The *best_corr_hp* value is filtered according to :

$$corr\_hp_{m+1} = (alpha) * corr\_hp_m + (1 - alpha) * best\_corr\_hp_m,$$

where alpha is varied between 0.98 and 0.8 as a function of *corr_hp*$_m$ and *best_corr_hp*$_m$
The *corr_hp* output value is thresholded into two to registers *complex_high*, *complex_low* and one *counter complex_hang_timer*.
complex_low is set to 1 if the *corr_hp* value is greater than CVAD_THRESH_ADAPT_LOW.
complex_high is set to 1 if the *corr_hp* value is greater than CVAD_THRESH_ADAPT_HIGH.
complex_hang_timer is increased by 1 if the *corr_hp* value is greater than CVAD_THRESH_HANG. If the *corr_hp* value is lower than or equal to CVAD_THRESH_HANG the *complex_hang_timer* value is set to 0.
The flag ***complex_warning*** is set if *complex_low have been set for 15 consecutive frames* or *complex_high* has been set for 8 consecutive frames.

The open-loop pitch search and correspondingly the tone flag is computed twice in each frame, except for modes 5.15 kbit/s and 4.75 kbit/s, where it is computed only once. The computation of the corr_hp value is however always done only once per frame using the newest correlation vector available.

3.3.5 VAD decision

Power of the input frame is calculated as follows:

$$pow\_sum = \sum_{i=-L\_NEXT}^{L\_FRAME-L\_NEXT-1} s(i) * s(i)\,, \tag{3.7}$$

where samples s(i) of the input frame are pointed by the new_speech pointer of the speech encoder. If the power of the input frame (pow_sum) is lower than the constant POW_PITCH_THR, last pitch flag is set to zero. If the power of the input frame (pow_sum) is lower than the constant POW_COMPLEX_THR, last complex_low flag is set to zero.
The difference between the signal levels of the input frame and background noise estimate is calculated as follows:

$$snr\_sum = \sum_{n=1}^{9} MAX(1.0, \frac{level[n]}{bckr\_est[n]})^2\ , \tag{3.8}$$

where:

level[n]	signal level at band n
bckr_est[n]	level of background noise estimate at band n

VAD decision is made by comparing the variable snr_sum to a threshold. The threshold (vad_thr) is tuned to get desired sensitivity at each background noise level. The higher the noise level the lower is the threshold. Specially, a low threshold at high-level background noise is needed to detect speech reliably enough, although probability of detecting noise as speech also increases.
Average level of background noise is calculated by adding noise estimates at each band:

$$noise\_level = \sum_{n=1}^{9} bckr\_est[n] \tag{3.9}$$

Threshold is calculated using average noise level as follows:

$$vad\_thr = VAD\_SLOPE * (noise\_level - VAD\_PI) + VAD\_THR\_HIGH\ , \tag{3.10}$$

where VAD_SLOPE, VAD_P1, and VAD_THR_HIGH are constants.
The variable vadreg indicates intermediate VAD decision and it is calculated as follows:

```
if (snr_sum > vad_thr)
        vadreg = 1
else
          vadreg = 0
```

3.3.5.1 Hangover addition

Before the final VAD flag is given, a hangover is added. The hangover addition helps to detect low power endings of speech bursts, which are subjectively important but difficult to detect. Also a long hangover is added if the signal has been found to be of very complex nature for a long time (2 seconds) since the VAD is not likely to work reliably for such a complex signal.
VAD flag is set to “1” if less that hang_len frames with “0” decision have been elapsed since burst_len consecutive “1” decisions have been detected. The variables hang_len and burst_len are set depending on the average noise level (noise_level). The ***vad_flag*** is also controlled by the ***complex_hang_count*** which indicates that the signal is too complex for the VAD and should not be used with a Comfort noise generation algorithm. The filtered correlation value ***corr_hp*** is also used as an activity indication after the VAD has indicated noise for a while (during 200 ms), this will aid in situations where the VAD noise estimate has adapted to a rather stationary but still all to complex signal to make it sound well with CNG.
The power of the input frame is compared to a threshold (VAD_POW_LOW). If the power is lower, the VAD flag is set to “0” and no hangover is added. The VAD_flag is calculated as follows:

```
if (noise_level > HANG_NOISE_THR)
        burst_len = BURST_LEN_HIGH_NOISE
        hang_len = HANG_LEN_HIGH_NOISE
else
```

```
          burst_len = BURST_LEN_LOW_NOISE
          hang_len = HANG_LEN_LOW_NOISE
if(complex_hang_timer > CVAD_HANG_LIMIT) {
      if(complex_hang_count < CVAD_HANG_LENGTH  {
         complex_hang_count = CVAD_HANG_LENGTH;
      }
}

if (powsum < VAD_POW_LOW){
          burst_count = 0
          hang_count = 0
          complex_hang_count = 0;
          complex_hang_timer = 0;
          Vad_flag=0;
        Goto **Exit**;
}
VAD_flag=0;
if(complex_hang_count != 0){
      burst_count = BURST_LEN_HIGH_NOISE;
      complex_hang_count =  complex_hang_count – 1 ;
       VAD_flag=1;
       goto **Exit**
} else {
      if ( (the 10 last out of 11 vadreg values all are zero)  AND
          (corr_hp > CVAD_THRESH_IN_NOISE ) ) {
        VAD_flag = 1;
        Goto **Exit**
      }
   }

if (vadreg = 1){
             burst_count = burst_count + 1}
          if (burst_count >= burst_len){
                 hang_count = hang_len
             }
          VAD_flag = 1
} else {
             burst_count = 0
          if (hang_count > 0){
                 hang_count = hang_count - 1
                 VAD_flag=1
          }
}
Label **Exit**
```

3.3.5.2 Background noise estimation

Background noise estimate (bckr_est[n]) is updated using amplitude levels of the previous frame. Thus, the update is delayed by one frame to avoid undetected start of speech bursts to corrupt the noise estimate. If the internal VAD decision is “1” or if pitch has been detected, the noise estimate is not updated upwards. The update speed for the current frame is selected as follows:

```
if ((vadreg for the last 4 frames has been zero) AND
    (pitch for the last 4 frames has been zero) AND
     (we are not in complex signal hangover))
          alpha_up = ALPHA_UP1
          alpha_down = ALPHA_DOWN1
  else
             if ((stat_count = 0 ) AND (not in complex_signal hangover))
                        alpha_up = ALPHA_UP2
          alpha_down = ALPHA_DOWN2
             else
                        alpha_up = 0
```

```
        alpha_down = ALPHA3
```

The variable stat_count indicates stationary and its propose is explained later in this subclause. The variables alpha_up and alpha_down define the update speed to upwards and downwards. The update speed for each band n is selected as follows:

```
if ($bckr\_est_m[n] < level_{m-1}[n]$)
      alpha = alpha_up
else
      alpha = alpha_down
```

Finally, noise estimate is updated as follows:

$$bckr\_est_{m+1}[n] = (1.0 - alpha) * bckr\_est_m[n] + alpha * level_{m-1}[n], \quad (3.11)$$

where:

n	index of the frequency band
m	index of the frame

Level of the background estimate (bckr_est[n]) is limited between constants NOISE_MIN and NOISE_MAX. If level of background noise increases suddenly, vadreg will be set to "1" and background noise is not updated upwards. To recover from this situation, update of the background noise estimate is enabled if the intermediate VAD decision (vadreg) is “1” for enough long time and spectrum is stationary. Stationary (stat_rat) is estimated using following equation:

$$stat\_rat = \sum_{n=1}^{9} \frac{MAX(\mathrm{STAT\_THR\_LEVEL}, \mathrm{MAX}(ave\_level_m[n], level_m[n]))}{MAX(\mathrm{STAT\_THR\_LEVEL}, \mathrm{MIN}(ave\_level_m[n], level_m[n]))} \quad (3.12)$$

If the stationary estimate (stat_rat) is higher than a threshold, the stationary counter (stat_count) is set to the initial value defined by constant STAT_COUNT. The stationary counter (stat_count) is also initialised if pitch or tone or a complex_warning is detected. If the signal is not stationary but speech has been detected (VAD decision is “1”), stat_count is decreased by one in each frame until it is zero.

```
if (***complex_warning***){
    If(stat_count < CAD_MIN_STAT_COUNT)
        stat_count = < CAD_MIN_STAT_COUNT
}


if ( (8 last vadreg flags have been zero) OR (2 last pitch flags have been one) OR (5 last tone flags have been one) )
    stat_count = STAT_COUNT
else
      if (stat_rat > STAT_THR)
            stat_count = STAT_COUNT
      else
            if ((vadreg) AND (stat_count $\neq$ 0))
                  stat_count = stat_count - 1
```

The average signal levels (ave_level[n]) are calculated as follows:

$$ave\_level_{m+1}[n] = (1.0 - alpha) * ave\_level_m[n] + alpha * level_m[n] \quad (3.13)$$

The update speed (alpha) for the previous equation is selected as follows:

```
if (stat_count = STAT_COUNT)
      alpha = 1.0
else if (vadreg = 1)
      alpha=ALPHA5
else
      alpha = ALPHA4
```

4 Technical Description of VAD Option 2

4.1 Definitions, symbols and abbreviations

4.1.1 Definitions

For the purposes of the present document, the following definitions apply:
codec: The combination of an encoder and decoder in series (encoder/decoder).
compand: The process of compressing and expanding a signal. In this text, the process is described in terms of PCM [4].
Decoder: Generally, a device for the translation of a signal from a digital representation into an analog format. For this standard, a device which converts speech encoded in the format specified in this standard to analog or an equivalent PCM representation.
DFT: See Discrete Fourier Transform.
Discrete Fourier Transform (DFT): A method of transforming a time domain sequence into a corresponding frequency domain sequence.
Encoder: Generally, a device for the translation of a signal into a digital representation. For this standard, a device which converts speech from an analog or its equivalent PCM representation to the digital representation described in this standard.
Fast Fourier Transform (FFT): An efficient implementation of the Discrete Fourier Transform.
FFT: See Fast Fourier Transform.
Vocoder: Voice coder.
frame: Time interval of 20 ms corresponding to the time segmentation of the speech transcoder.

4.1.2 Symbols

For the purposes of this TS, the following symbols apply.

4.1.2.1 Variables

$\alpha_{ch}(m)$	channel energy smoothing factor
$\alpha(m)$	exponential windowing factor
$\Delta_E(m)$	estimated spectral deviation between current power spectrum and average long term power spectral estimate
$\phi(m)$	spectral peak-to-average ratio
$\sigma_q^{(i)}$	quantized channel SNR indices
$b(m)$	burst count
b_{th}	burst count threshold
$\{d(m)\}$	overlapped portion of the frame buffer of input samples
$E_{ch}(m,i)$	channel energy estimate; channel i, subframe m
$\mathbf{E}_{ch}(m)$	vector of channel energy estimates, $0 \le i < N_c$
$E_{dB}(m,i)$	estimated log power spectrum
$\mathbf{E}_{dB}(m)$	vector of log power spectrum estimates, $0 \le i < N_c$
$\overline{E}_{dB}(m,i)$	average long term power spectral estimate
$\overline{E}_{dB}(m)$	vector of average long term power spectral estimates, $0 \le i < N_c$
$E_n(m,i)$	channel noise estimate
$\mathbf{E}_n(m)$	vector of channel noise estimates, $0 \le i < N_c$

$E_{tn}(m)$	total estimated noise energy
$E_{tot}(m)$	total channel energy
$E'_{tot}(m)$	modified total channel energy
$h(m)$	hysteresis counter
h_{cnt}	hangover count
$h_o(n)$	overlap-and-add buffer of samples
hyster_cnt	hysteresis counter to avoid long term creeping of *update_cnt*
last_update_cnt	previous value of *update_cnt*
$s_{hp}(n)$	sample at the output of the speech encoder high pass filter
sinewave_flag	boolean flag, set TRUE when spectral peak-to-average ratio is greater than 10dB and the spectral deviation is less than DEV_THLD
SNR	Signal to Noise ratio
$SNR_p(m)$	long-term peak SNR
$SNR_q(m)$	quantized version of $SNR_p(m)$
update_cnt	counter gating noise estimate update process
update_flag	flag controlling noise estimate updating
VAD(m)	boolean VAD flag for subframe m
VAD_flag	boolean VAD Flag
v(m)	sum of voice metrics
v_{th}	voice metric threshold

4.1.2.2 Constants

α_H	upper limit for values of $\alpha(m)$
α_L	lower limit for values of $\alpha(m)$
α_n	channel noise smoothing factor
ζ_p	pre-emphasis factor
b_{table}	table to generate b_{th}
D	overlap (delay) in sample intervals
DEV_THLD	threshold for setting *sinewave_flag*
E_{floor}	low threshold for $E_{tot}(m)$
E_H	high energy endpoint for linear interpolation of $E_{tot}(m)$
E_{init}	minimum allowable channel noise initialisation energy
E_L	low energy endpoint for linear interpolation of $E_{tot}(m)$
E_{min}	minimum allowable channel energy
f_H	high channel combining table
f_L	low channel combining table

g(n)	trapezoidal window, n = 0 to M
G(k)	frequency domain transformation of g(n)
h_{table}	table to generate h_{cnt}
HYSTER_CNT_THLD	threshold for *hyster_cnt*
L	subframe length in samples
M	DFT sequence length
N_c	number of combined channels
NOISE_FLOOR_D	low threshold for $E_{tot}(m)$ in dB
UPDATE_CNT_THLD	threshold for *update_cnt*
UPDATE_THLD	threshold for *v(m)*
V	voice metric table
v_{table}	table to generate v_{th}

4.1.2.3 Functions

+	addition
-	subtraction
*****	multiplication
/	division
$\lfloor x \rfloor$	largest integer ≤ x
AND	Boolean AND
OR	Boolean OR
$\sum_{n=a}^{b} x(n)$	$= x(a) + x(a+1) + \ldots + x(b-1) + x(b)$

4.1.3 Abbreviations

ANSI	American National Standards Institute
DTX	Discontinuous Transmission
VAD	Voice Activity Detector
CAD	Complex Activity Detection
CNG	Comfort Noise Generation

4.2 General

The function of the VAD algorithm is to indicate whether each 20 ms frame contains signals that should be transmitted, i.e. speech, music or information tones. The output of the VAD algorithm is a Boolean flag (VAD_flag) indicating presence of such signals.

4.3 Functional description

The block diagram of the VAD algorithm is depicted in figure 4.1. The VAD algorithm uses parameters of the speech encoder to compute the Boolean VAD flag (VAD_flag).

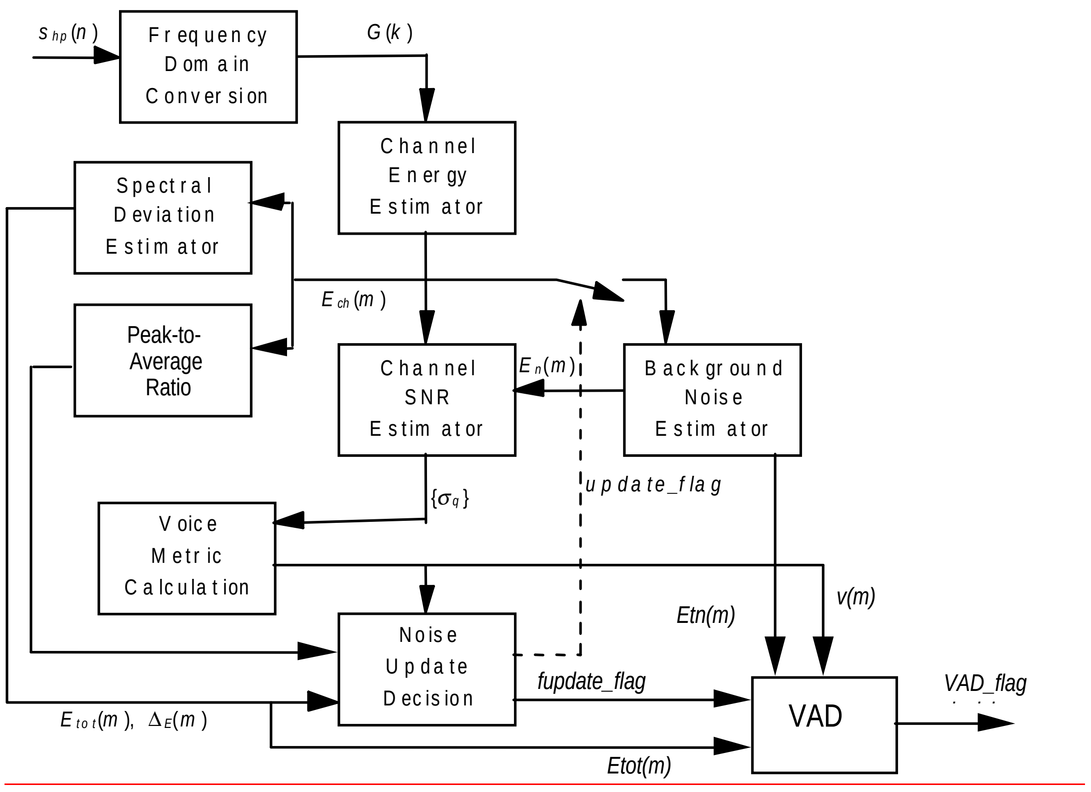

Figure 4.1. Block Diagram of the VAD algorithm: Option 2

Input:

_ The output of the High-Pass Filter, $\{s_{hp}(n)\}$

- *LTP_flag* is generated by the comparison of the long-term prediction gain to a constant threshold LTP_THLD, where the long-term prediction gain β is derived from the speech encoder[2] open-loop pitch predictor.

Output:

_ The output of the vad is designated as VAD_flag

Initialization:
The following variables shall be set to zero at initialization (frame m = 0):

_ The pre-emphasis memory

The following shall be initialized to a startup value other than zero:

_ The channel energy estimate, $\mathbf{E}_{ch}(m)$, (see Section 4.3.2)

_ The long-term power spectral estimate, $\overline{\mathrm{E}}_{dB}(m)$, (see Section 4.3.5)

_ The channel noise estimate, $\mathbf{E}_n(m)$, (see Section 4.3.8)

Processing: The following procedures shall be executed two times per 20 ms speech frame and the current 10 ms subframe shall be denoted *m*.

4.3.1 Frequency Domain Conversion

The input signal is pre-emphasised and windowed prior to frequency domain conversion. This process is defined as

$$d(n) = s_{hp}(n) + \zeta_p s_{hp}(n-1), \quad 0 \le n < L, \tag{4.1}$$

where $d(n)$ is the pre-emphasised speech buffer, ζ_p is the pre-emphasis factor, and L is the subframe length. A rectangular window is then used to frame the speech prior to frequency domain conversion, which is expressed as:

$$g(n) = \begin{cases} 0, & 0 \le n < D,\ L+D \le n < M \\ d(n-D), & D \le n < L+D \end{cases}, \tag{4.2}$$

where D is the zero-padding offset into the DFT buffer, and M is the DFT length. The transformation of $g(n)$ to the frequency domain is performed using the Discrete Fourier Transform (DFT) defined[1] as:

$$G(k) = \frac{2}{M} \sum_{n=0}^{M-1} g(n) e^{-j2\pi nk/M}, \quad 0 \le k < M \tag{4.3}$$

where $e^{j\omega}$ is a unit amplitude complex phasor with instantaneous radial position ω.

4.3.2 Channel Energy Estimator

Calculate the channel energy estimate $\mathbf{E}_{ch}(m)$ for the current subframe, m, as:

$$E_{ch}(m,i) = \max\left\{ E_{\min},\ \alpha_{ch}(m) E_{ch}(m-1,i) + (1-\alpha_{ch}(m)) \frac{1}{f_H(i) - f_L(i) + 1} \sum_{k=f_L(i)}^{f_H(i)} |G(k)|^2 \right\}, \quad 0 \le i < N_c \tag{4.4}$$

where $E_{\min}$ is the minimum allowable channel energy, $\alpha_{ch}(m)$ is the channel energy smoothing factor (defined below), N_c is the number of combined channels, and $f_L(i)$ and $f_H(i)$ are the i-th elements of the respective low and high channel combining tables.
The channel energy smoothing factor, $\alpha_{ch}(m)$, is defined as:

$$\alpha_{ch}(m) = \begin{cases} 0, & m \le 1 \\ 0.45, & m > 1 \end{cases} \tag{4.5}$$

So, this means that $\alpha_{ch}(m)$ assumes a value of zero for the first frame ($m = 1$) and a value of 0.45 for all subsequent frames. This allows the channel energy estimate to be initialized to the unfiltered channel energy of the first frame.

4.3.3 Channel SNR Estimator

Estimate the channel SNR vector $\{\sigma\}$ as:

$$\sigma(i) = 10\log_{10}\left(\frac{E_{ch}(m,i)}{E_n(m,i)} \right), \quad 0 \le i < N_c \tag{4.6}$$

where $\mathbf{E}_n(m)$ is the current channel noise energy estimate (see Section 4.3.8), and then quantify the channel SNR estimate in 3/8 dB steps to yield the channel SNR indices $\{\sigma_q\}$ given as:

$$\sigma_q(i) = \max\{0, \min\{89, \mathrm{round}\{\sigma(i)/0.375\}\}\}, \quad 0 \le i < N_c \tag{4.7}$$

where the values of $\{\sigma_q\}$ are constrained to be between 0 and 89, inclusive.

[1] This atypical definition is used to exploit the efficiencies of the complex Fast Fourier Transform (FFT). The $2/M$ scale factor results from preconditioning the M point real sequence to form an $M/2$ point complex sequence that is transformed using an $M/2$ point complex FFT. Details on this technique can be found in Proakis, J. G. and Manolakis, D. G., *Introduction to Digital Signal Processing*, New York, Macmillan, 1988, pp. 721-722.

4.3.4 Voice Metric Calculation

Next, calculate the sum of voice metrics as:

$$v(m) = \sum_{i=0}^{N_c-1} V\left(\sigma_q(i)\right), \tag{4.8}$$

where $V(k)$ is the kth value of the 90 element voice metric table **V**.

4.3.5 Frame SNR and Long-Term Peak SNR Calculation

The instantaneous frame SNR, *SNR*, and long-term peak SNR, $SNR_p(m)$, are used to calibrate the responsiveness of the VAD decision. When the frame count is less than or equal to four ($m \le 4$) or the forced update flag (sec 4.3.10) is set (fupdate_flag == TRUE), then the SNR's are initialized as:

$$SNR_p(m) = SNR = 56 - 10\log_{10}\left(\sum_{i=0}^{N_c-1} E_n(m,i)\right). \tag{4.9}$$

Otherwise, the instantaneous frame SNR is generated by:

$$SNR = 10\log_{10}\left(\frac{1}{N_c}\sum_{i=0}^{N_c-1} 10^{\sigma(i)/10}\right) \tag{4.10}$$

and the long-term peak SNR is derived by the following expression:

$$SNR_p(m) = \begin{cases} 0.9SNR_p(m-1) + 0.1SNR, & SNR > SNR_p(m-1) \\ 0.998SNR_p(m-1) + 0.002SNR, & 0.625SNR_p(m-1) < SNR \le SNR_p(m-1) \\ SNR_p(m-1), & otherwise \end{cases} \tag{4.11}$$

.

The long-term peak SNR is then quantized in 3 dB steps and limited to be between 0 and 19, as follows:

$$SNR_q = \max\left\{\min\left\{\lfloor SNR_p(m)/3 \rfloor, 19\right\}, 0\right\} \tag{4.12}$$

where $\lfloor x \rfloor$ is the largest integer $\le x$ (floor function).

4.3.6 Negative SNR Sensitivity Bias

In order for the VAD decision to overcome the problem of being over-sensitive to fluctuating, non-stationary background noise conditions, a bias factor is used to increase the threshold on which the VAD decision is based. This bias factor is derived from an estimate of the variablility of the background noise estimate. The variability estimate is further based on negative values of the instantaneous SNR. It is presumed that a negative SNR can only occur as a result of fluctuating background noise, and not from the presence of voice. Therefore, the bias factor $\mu(m)$ is derived by first calculating the variability factor $\psi(m)$ as:

$$\psi(m) = \begin{cases} 0.99\psi(m-1) + 0.01SNR^2, & SNR < 0 \\ \psi(m-1) & otherwise \end{cases} \tag{4.13}$$

which is then clamped in magnitude to $0 \le \psi(m) \le 4.0$. In addition, the variability factor is reset to zero when the frame count is less than or equal to four ($m \le 4$) or the forced update flag (sec 4.3.10) is set (*fupdate_flag* == TRUE). The bias factor $\mu(m)$ is then calculated as:

$$\mu(m) = \max\left\{12.0\left(\psi(m) - 0.65\right), 0\right\} \tag{4.14}$$

4.3.7 VAD Decision

The quantized SNR SNR_q is used to determine the respective voice metric threshold v_{th}, hangover count h_{cnt}, and burst count threshold b_{th} parameters:

$$v_{th} = v_{table}\left(SNR_q\right), \quad h_{cnt} = h_{table}\left(SNR_q\right), \quad b_{th} = b_{table}\left(SNR_q\right) \tag{4.15}$$

where SNR_Q is the index of the respective table elements. The VAD decision can then be made according to the following pseudocode:

```
if ( $v(m) > v_{th} + \mu(m)$) {           /* if the voice metric > voice metric threshold*/
    $VAD(m)$ = ON
    $b(m) = b(m-1) + 1$                    /* increment burst counter */
    if ( $b(m) > b_{th}$ ) {               /* compare counter with threshold */
        $h(m) = h_{cnt}$                   /* set hangover */
    }
} else {
    $b(m) = 0$                             /* clear burst counter */
    $h(m) = h(m-1) -1$                     /* decrement hangover /
    if ( $h(m) <= 0$ ) {                   /* check for expired hangover /
        $VAD(m)$ = OFF
        $h(m) = 0$
    } else {
        $VAD(m)$ = ON                      /* hangover not yet expired */
    }
}
```

Note that two 10 ms subframes are required to determine one VAD decision. The final decision is determined by the maximum of two subframe decisions, i.e.

```
if($VAD(m)$ == ON OR $VAD(m-1)$ == ON) {
VAD_flag = TRUE
} else {
VAD_flag = FALSE
}
```

4.3.8 Spectral Deviation Estimator

The spectral deviation estimator is used as a safeguard against erroneous updates of the background noise estimate. If the spectral deviation of the input signal is too high, then the background noise estimate update may not be permitted. Calculate the estimated log power spectrum as:

$$E_{dB}(m,i) = 10\log_{10}\left(E_{ch}(m,i)\right), \quad 0 \le i < N_c \tag{4.16}$$

Then, calculate the estimated spectral deviation between the current power spectrum and the average long-term power spectral estimate:

$$\Delta_E(m) = \sum_{i=0}^{N_c-1}\left|E_{dB}(m,i) - \bar{E}_{dB}(m,i)\right| \tag{4.17}$$

where $\bar{\mathbf{E}}_{dB}(m)$ is the average long-term power spectral estimate calculated during the previous subframe, as defined in Equation 4.20. The initial value of $\bar{\mathbf{E}}_{dB}(m)$, however, is defined to be the estimated log power spectrum of subframe 1, or:

$$\bar{\mathbf{E}}_{dB}(m) = \mathbf{E}_{dB}(m), \quad m = 1 \tag{4.18}$$

The exponential windowing factor, $\alpha(m)$, is then calculated as a function of the instantaneous frame SNR SNR and the long-term peak SNR $SNR_p(m)$, as:

$$\alpha(m) = \alpha_H - \left((\alpha_H - \alpha_L) \frac{SNR_p(m) - SNR}{SNR_p(m)} \right), \qquad (4.19)$$

which is then limited to $\alpha_L \le \alpha(m) \le \alpha_H$.
The average long-term power spectral estimate is then updated for the next frame by:

$$\overline{E}_{dB}(m+1,i) = \alpha(m)\overline{E}_{dB}(m,i) + (1-\alpha(m))E_{dB}(m,i), \quad 0 \le i < N_c \qquad (4.20)$$

where all the variables are previously defined.

4.3.9 Sinewave Detection

Next the *sinewave_flag* is set TRUE when the spectral peak-to-average ratio $\phi(m)$ is greater than 10, i.e.

$$sinewave\_flag = \begin{cases} \text{TRUE}, & \phi(m) > 10 \\ \text{FALSE}, & otherwise \end{cases} \qquad (4.21)$$

where:

$$\phi(m) = 10\log_{10}\left(\frac{\max\{E_{ch}(m,i)\}}{\sum_{j=0}^{N_c-1} E_{ch}(m,j)/N_c} \right), \quad 2 \le i < N_c \qquad (4.22)$$

4.3.10 Background Noise Update Decision

The following logic, as shown in pseudo-code, demonstrates how the noise estimate update decision is ultimately made:

```
/* Normal update logic */
update_flag = fupdate_flag = FALSE
if ( $v(m) \le$ UPDATE_THLD and $b(m)$ == 0 ) {
    update_flag = TRUE
    update_cnt = 0
}
/* Forced update logic (for over-riding the normal update logic)*/
else if (( $E_{tot}$ > NOISE_FLOOR) and ( $\Delta_E(m)$ < DEV_THLD )
and ( sinewave_flag == FALSE ) and (LTP_flag == FALSE)) {
    update_cnt = update_cnt + 1
    if ( update_cnt $\ge$ UPDATE_CNT_THLD )
        update_flag = fupdate_flag = TRUE
}
/* "Hysteresis" logic to prevent long-term creeping of update_cnt */
if ( update_cnt == last_update_cnt )
    hyster_cnt = hyster_cnt + 1
else
    hyster_cnt = 0
last_update_cnt = update_cnt
if ( hyster_cnt > HYSTER_CNT_THLD )
    update_cnt = 0
```

where E_{tot} is the total channel energy defined as:

$$E_{tot} = \sum_{i=0}^{N_c-1} E_{ch}(m,i) \qquad (4.23)$$

and *LTP_flag* is generated by the comparison of the long-term prediction gain to a constant threshold LTP_THLD, i.e.:

$$LTP\_flag = \begin{cases} \text{TRUE}, & \beta > \text{LTP\_THLD} \\ \text{FALSE}, & otherwise \end{cases} \tag{4.24}$$

where the long-term prediction gain β is derived from the speech encoder [2] open-loop pitch predictor, and can be expressed as:

$$\beta = \frac{\sum_{n=0}^{N_p-1} s_w(n)s_w(n-k)}{\sum_{n=0}^{N-1} s_w^2(n)} \tag{4.25}$$

where $s_w(n)$ is the weighted speech, k is the optimal open-loop lag, and N_p is the pitch analysis frame length. This expression is calculated in the speech encoder on the previous frame.

4.3.10 Background Noise Estimate Update

If (and only if) the update flag is set (*update_flag* == TRUE), then update the channel noise estimate for the next subframe by:

$$E_n(m+1,i) = \max\{E_{\min}, \alpha_n E_n(m,i) + (1-\alpha_n)E_{ch}(m,i)\}, \quad 0 \le i < N_c \tag{4.26}$$

where $E_{\min}$ is the minimum allowable channel energy, and α_n is the channel noise smoothing factor. The channel noise estimate shall be initialized for each of the first four frames to the estimated channel energy, i.e.:

$$E_n(m,i) = \max\{E_{init}, E_{ch}(m,i)\}, \quad m \le 4, \quad 0 \le i < N_c, \tag{4.27}$$

where E_{init} is the minimum allowable channel noise initialization energy.

5 Computational details

A low level description has been prepared in form of ANSI C source code [1].

Annex A (informative) : Change history

Change history					

History

Document history		
V. 2.0.0	October 1999	3G TS 26.094 presented for approval to TSG#5 Plenary meeting