HMM-based Speech Synthesis System (HTS) - Reference of hts_engine API

Preprocessor symbols for conditional compilation ↑

- For embedded device

```
HTS_EMBEDDED (slightly faster setting is used)
```

- Audio device setting

```
AUDIO_PLAY_WIN32 (for Windows 2000/XP/Vista C++ compiler)
AUDIO_PLAY_WINCE (for Windows Mobile C++ compiler)
AUDIO_PLAY_NONE  (default)
```

- For Festival speech synthesis system

```
FESTIVAL
```

- Endian definition

```
WORDS_BIGENDIAN (e.g. PowerPC, Cell BE, SPARC, 680x0)
WORDS_LITTLEENDIAN (e.g. x86, Alpha AXP)
```

- Boolean definition (HTS_Boolean)

```
TRUE
FALSE
```

- Vocoder type definition (HTS_VocoderType?)

```
HTS_VOCODER_TYPE_MCP (MLSA filter : if Gamma!=0 then MGLSA filter)
HTS_VOCODER_TYPE_LSP
```

Engine structures †

Model †

HTS_Window †

- Window coefficients to calculate dynamic features.

```
int size             - # of windows (static + deltas)
int *l_width         - left width of windows
int *r_width         - right width of windows
double **coefficient - window coefficients
int max_width        - maximum width of windows
```

HTS_Pattern †

- List of patterns in a question and a tree.

```
char *string      - pattern string
HTS_Pattern *next - pointer to the next pattern
```

HTS_Question †

- List of questions in a tree.

```
char *name         - name of this question
HTS_Pattern *head  - pointer to the head of pattern list
HTS_Question *next - pointer to the next question
```

HTS_Node †

- List of tree nodes in a tree.

```
int index           - index of this node
int pdf             - index of PDF for this node (leaf node only)
HTS_Node *yes       - pointer to its child node (yes)
HTS_Node *no        - pointer to its child node (no)
HTS_Node *next      - pointer to the next node
HTS_Question *quest - question applied at this node
```

HTS_Tree †

- List of decision trees in a model.

```
HTS_Pattern *head - pointer to the head of pattern list for this tree
HTS_Tree *next    - pointer to the next tree
HTS_Node *root    - root node of this tree
```

HTS_Model †

- Set of PDFs, decision trees and questions.

```
int nstate             - # of HMM states
int vector_length      - vector length (include static and dynamic features)
int *npdf              - # of PDFs at each state position
double ***pdf          - PDFs
HTS_Tree *tree         - pointer to the list of trees
HTS_Question *question - pointer to the list of questions
```

HTS_Stream †

- Set of models and a window.

```
int vector_length        - vector_length (include static and dynamic features)
HTS_Model *model         - models
HTS_DynamicWindow window - window coefficients
HTS_Boolean msd_flag     - flag for MSD
int interpolation_size   - # of models for interpolation
```

HTS_ModelSet? †

- Set of duration models, HMMs and GV models.

```
HTS_Stream duration - duration PDFs and trees
HTS_Stream *stream  - parameter PDFs, trees and windows
HTS_Stream *gv      - GV PDFs
int nstate          - # of HMM states
int nstream         - # of stream
```

↑

Label ↑

↑

HTS_LabelString? ↑

- Individual label string with time infomation.

```
HTS_LabelString *next  - pointer to the next label string
char *name             - label string
HTS_Boolean frame_flag - flag for frame length modification
int frame              - frame length specified in the given label
HTS_Boolean speed_flag - flag for speech speed modification
double speed           - speech speed specified in the given label
```

↑

HTS_Label ↑

- List of label strings.

```
HTS_LabelString *head - pointer to the head of label string
int size              - # of label strings
```

↑

State stream ↑

↑

HTS_SStream ↑

- Individual state stream.

```
int vector_length        - vector length (include static and dynamic features)
double **mean            - mean vector sequence
double **vari            - variance vector sequence
double *msd              - MSD parameter sequence
int win_size             - # of windows (static + deltas)
int *win_l_width         - left width of windows
int *win_r_width         - right width of windows
double **win_coefficient - window coefficients
int win_max_width        - maximum width of windows
double *gv_mean          - mean vector of GV
double *gv_vari          - variance vector of GV
```

↑

HTS_SStreamSet? ↑

- Set of state stream.

```
HTS_SStream *sstream - state streams
int nstream          - # of streams
int nstate           - # of states
int *duration        - duration sequence
int total_state      - total state
int total_frame      - total frame
```

↑

PDF stream ↑

↑

HTS_SMatrices ↑

- Matrices/Vectors used in the speech parameter generation algorithm.

```
double **mean - mean vector sequence
double **ivar - inverse diag variance sequence
double *g     - vector used in the forward substitution
double **wuw  - W' U^-1 W
double *wum   - W' U^-1 mu
```

↑

HTS_PStream ↑

- Individual PDF stream.

```
int vector_length        - vector length (include static and dynamic features)
int static_length        - static features length
int length               - stream length
int width                - width of dynamic window
double **par             - output parameter vector
HTS_SMatrices sm         - matrices for parameter generation
int win_size             - # of windows (static + deltas)
int *win_l_width         - left width of windows
int *win_r_width         - right width of windows
double **win_coefficient - window coefficients
HTS_Boolean *msd_flag    - Boolean sequence for MSD
double *gv_buff          - buffer for GV calculation
double *gv_mean          - mean vector of GV
double *gv_vari          - variance vector of GV
double gv_weight         - GV weight
```

↑

HTS_PStreamSet? ↑

- Set of PDF streams.

```
HTS_PStream *pstream - PDF streams
int nstream          - # of PDF streams
int total_frame      - total frame
```

↑

Generated parameter stream ↑

↑

HTS_GStream ↑

- Generated parameter stream.

```
int static_length - static features length
double **par      - generated parameter
```

↑

HTS_GStreamSet? ↑

- Set of generated parameter stream.

```
int total_nsample    - total sample
int total_frame      - total frame
int nstream          - # of streams
HTS_GStream *gstream - generated parameter streams
short *gspeech       - generated speech
```

↑

Engine ↑

↑

HTS_Global ↑

- Global settings.

```
int stage             - Gamma=-1/stage : if stage=0 then Gamma=0
int sampling_rate     - sampling rate
int fperiod           - frame period
double rho            - control duration parameter
double alpha          - all-pass constant
double beta           - postfiltering coefficient
int audio_buff_size   - audio buffer size (for audio device)
HTS_VocoderType type  - vocoder type
double *msd_threshold - MSD thresholds
double *duration_iw   - weights for duration interpolation
double **parameter_iw - weights for parameter interpolation
double **gv_iw        - weights for GV interpolation
double *gv_weight     - GV weights
```

↑

HTS_Engine ↑

- Engine itself.

```
HTS_Global global  - global settings
HTS_ModelSet ms    - set of duration models, HMMs and GV models
HTS_Label label    - label
HTS_SStreamSet sss - set of state streams
HTS_PStreamSet pss - set of PDF streams
HTS_GStreamSet gss - set of generated parameter streams
```

↑

Vocoder structures ↑

↑

Audio ↑

↑

HTS_Audio ↑

- For MS Windows (Windows Mobile) audio output device.

```
HWAVEOUT hwaveout         - audio device handle
WAVEFORMATEX waveformatex - wave formatex
short *buff               - current buffer
int buff_size             - current buffer size
int which_buff            - double buffering flag
HTS_Boolean now_buff_1    - double buffering flag
HTS_Boolean now_buff_2    - double buffering flag
WAVEHDR buff_1            - buffer
WAVEHDR buff_2            - buffer
int max_buff_size         - buffer size of audio output device
```

- For Linux, etc.

```
int i                     - make compiler happy
```

↑

Vocoder ↑

↑

HTS_Vocoder ↑

- MLSA/MGLSA filter settings.

```
HTS_VocoderType type       - vocoder type
int stage                  - Gamma=-1/stage : if stage=0 then Gamma=0
double gamma               - Gamma
int fprd                   - frame shift
int iprd                   - interpolation period
int seed                   - seed of random generator
unsigned long next         - temporary variable for random generator
HTS_Boolean gauss          - flag to use Gaussian noise
double rate                - sampling rate
double p1                  - used in excitation generation
double pc                  - used in excitation generation
int sw                     - switch used in random generator
int x                      - excitation signal
HTS_Audio *audio           - pointer for audio device
double *freqt_buff         - used in freqt
int freqt_size             - buffer size for freqt
double *spectrum2en_buff   - used in spectrum2en
int spectrum2en_size       - buffer size for spectrum2en
double r1, r2, s           - used in random generator
double *postfilter_buff    - used in postfiltering
int postfilter_size        - buffer size for postfiltering
double *c, *cc, *cinc, *d1 - used in the MLSA/MGLSA filter
double *pade               - used in mlsadf
double *lsp2lpc_buff       - used in lsp2lpc
int lsp2lpc_size           - buffer size of lsp2lpc
double *gc2gc_buff         - used in gc2gc
int gc2gc_size             - buffer size for gc2gc
```

↑

Engine functions ↑

↑

Initialize engine ↑

↑

HTS_Engine_initialize ↑

```
void HTS_Engine_initialize(HTS_Engine *engine, int nstream)
```

- Use: Initialize engine.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
```

- **Attention!!:** To start engine, first you must call this function.

↑

Load models ↑

↑

HTS_Engine_load_duration_from_fn ↑

```
void HTS_Engine_load_duration_from_fn(HTS_Engine *engine, char **pdf_fn, char **tree_fn, int interpolation_size)
```

- Use: Load duration PDFs and trees from files using given file names.
- Arguments:

```
HTS_Engine *engine     - pointer to HTS_Engine structure
char **pdf_fn          - duration PDF file names
char **tree_fn         - duration tree file names
int interpolation_size - # of duration models to be interpolated
```

- **Attention!!**: You must initialize variable *engine* using HTS_Engine_initialize before calling this function.

↑

HTS_Engine_load_duration_from_fp ↑

void HTS_Engine_load_duration_from_fp(HTS_Engine *engine, FILE **pdf_fp, FILE **tree_fp, int interpolation_size)

- Use: load duration PDFs and trees from files using given file pointers.
- Arguments:

```
HTS_Engine *engine     - pointer to HTS_Engine structure
FILE **pdf_fp          - duration PDF file pointers
FILE **tree_fp         - duration tree file pointers
int interpolation_size - # of duration models to be interpolated
```

- **Attention!!**: You must initialize variable *engine* using HTS_Engine_initialize before calling this function.

↑

HTS_Engine_load_parameter_from_fn ↑

```
void HTS_Engine_load_parameter_from_fn(HTS_Engine *engine, char **pdf_fn, char **tree_fn, char **win_fn, int stream_index, HTS_Boolean msd_flag, int window_size, int interpolation_size)
```

- Use: load parameter PDFs, trees and windows from files using given file names.
- Arguments:

```
HTS_Engine *engine     - pointer to HTS_Engine structure
char **pdf_fn          - parameter PDF file pointers
char **tree_fn         - parameter tree file pointers
char **win_fn          - parameter window file pointers
int stream_index       - index of stream
HTS_Boolean msd_flag   - flag for MSD
int window_size        - # of windows
int interpolation_size - # of parameter models to be interpolated
```

- **Attention!!**: You must load duration models using HTS_Engine_load_duration_from_fn/fp before calling this function.

↑

HTS_Engine_load_parameter_from_fp ↑

```
void HTS_Engine_load_parameter_from_fp(HTS_Engine *engine, FILE **pdf_fp, FILE **tree_fp, FILE **win_fp, int stream_index, HTS_Boolean msd_flag, int window_size, int interpolation_size)
```

- Use: load parameter PDFs, trees and windows from files using given file pointers.
- Arguments:

```
HTS_Engine *engine     - pointer to HTS_Engine structure
FILE **pdf_fp          - parameter PDF file pointers
FILE **tree_fp         - parameter tree file pointers
FILE **win_fp          - parameter window file pointers
int stream_index       - index of stream
HTS_Boolean msd_flag   - MSD flag
int window_size        - # of windows
int interpolation_size - # of parameter models to be interpolated
```

- **Attention!!**: You must load duration models using HTS_Engine_load_duration_from_fn/fp before calling this function.

↑

HTS_Engine_load_gv_from_fn ↑

```
void HTS_Engine_load_gv_from_fn(HTS_Engine *engine, char **pdf_fn, int stream_index, int interpolation_size)
```

- Arguments:

```
HTS_Engine *engine     - pointer to HTS_Engine structure
char **pdf_fn          - GV PDF file names
int stream_index       - index of streams
int interpolation_size - # of GV models to be interpolated
```

- **Attention!!**: You must load parameter models using HTS_Engine_load_parameter_from_fn/fp before calling this function.

↑

HTS_Engine_load_gv_from_fp ↑

```
void HTS_Engine_load_gv_from_fp(HTS_Engine *engine, FILE **pdf_fn, int stream_index, int interpolation_size)
```

- Arguments:

```
HTS_Engine *engine     - pointer to HTS_Engine structure
FILE **pdf_fn          - GV PDF file pointers
int stream_index       - index of streams
int interpolation_size - # of GV models to be interpolated
```

- **Attention!!**: You must load parameter models using HTS_Engine_load_parameter_from_fn/fp before calling this function.

↑

Set parameter ↑

↑

HTS_Engine_set_sampling_rate ↑

```
void HTS_Engine_set_sampling_rate(HTS_Engine *engine,int i)
```

- Use: set sampling frequency.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
int i              - sampling frequency (Hz), 1= < i <= 48000
```

- **Attention!!:** Default value is 16000.

↑

HTS_Engine_set_fperiod †

```
void HTS_Engine_set_fperiod(HTS_Engine *engine,int i)
```

- Use: set frame shift.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
int i              - frame shift (point), 1= < i <= 2000
```

- **Attention!!:** Default value is 80.

↑

HTS_Engine_set_alpha †

```
void HTS_Engine_set_alpha(HTS_Engine *engine,double f)
```

- Use: set frequency warping parameter alpha.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
double f           - alpha, 0.0 <= f <= 1.0
```

- **Attention!!:** Default value is 0.42.

↑

HTS_Engine_set_gamma †

```
void HTS_engine_set_gamma(HTS_Engine *engine,int i)
```

- Use: set Gamma.
- Arguments:

```
HTS_engine *engine - pointer to HTS_Engine structure
int i              - Gamma=-1/i : if i=0 then Gamma=0, 0 <= i
```

- **Attention!!:** Default value is 0.

↑

HTS_Engine_set_beta †

```
void HTS_SetBeta(HTS_Engine *engine,double f)
```

- Use: set postfiltering coefficient parameter beta.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
double f           - beta, -0.8 <= f <= 0.8
```

- **Attention!!:** Default value is 0.0. If you set beta large value, formant structure will be emphasized strongly.

↑

HTS_Engine_set_rho †

```
void HTS_Engine_set_rho(HTS_Engine *engine,double f)
```

- Use: set control duration parameter rho.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
double f           - rho, -1.0 <= f <= 1.0
```

- **Attention!!:** Default value is 0.0. If you set negative value, speaking rate of synthesized speech becomes fast.

↑

HTS_Engine_set_audio_buff_size †

```
void HTS_Engine_set_audio_buff_size(HTS_Engine *engine, int i)
```

- Use: set buffer size for direct audio output.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
int i              - buffer size (sample), 0 <= i <= 48000
```

- **Attention!!:** Default value is 0. If i=0, direct audio play is turned off.

↑

HTS_Engine_set_vocoder_type †

```
void HTS_Engine_set_vocoder_type(HTS_Engine *engine, HTS_Vocode_Type type)
```

- Use: set vocoder type.
- Arguments:

```
HTS_Engine *engine   - pointer to HTS_Engine structure
HTS_Vocode_Type type - vocoder type
```

↑

HTS_Engine_set_msd_threshold †

```
void HTS_Engine_set_msd_threshold(HTS_Engine *engine, int stream_index, double f)
```

- Use: set MSD threshold.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
int stream_index   - index of streams
double f           - threshold
```

↑

HTS_Engine_set_duration_interpolation_weight †

```
void HTS_Engine_set_duration_interpolation_weight(HTS_Engine *engine, int interpolation_index, double f)
```

- Use: set weight for duration interpolation.
- Arguments:

```
HTS_Engine *engine      - pointer to HTS_Engine structure
int interpolation_index - index of duration models
double f                - interpolation weight
```

↑

HTS_Engine_set_parameter_interpolation_weight ↑

```
void HTS_Engine_set_parameter_interpolation_weight(HTS_Engine *engine, int stream_index, int interpolation_index, double f)
```

- Use: set weight for parameter interpolation.
- Arguments:

```
HTS_Engine *engine      - pointer to HTS_Engine structure
int stream_index        - index of streams
int interpolation_index - index of parameter models
double f                - interpolation weight
```

↑

HTS_Engine_set_gv_interpolation_weight ↑

```
void HTS_Engine_set_gv_interpolation_weight(HTS_Engine *engine, int stream_index, int interpolation_index, double f)
```

- Use: set weight for GV interpolation.
- Arguments:

```
HTS_Engine *engine      - pointer to HTS_Engine structure
int stream_index        - index of streams
int interpolation_index - index of GV models
double f                - interpolation weight
```

↑

HTS_Engine_set_gv_weight ↑

```
void HTS_Engine_set_gv_weight(HTS_Engine *engine, int stream_index, double f)
```

- Use: set GV weight.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
int stream_index   - index of streams
double f           - GV weight
```

↑

Synthesize speech ↑

↑

HTS_Engine_load_label_from_fn ↑

```
void HTS_Engine_load_label_from_fn(HTS_Engine *engine, char *fn)
```

- Use: load label from file using given file name.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
char *fn           - label file name
```

↑

HTS_Engine_load_label_from_fp ↑

```
void HTS_Engine_load_label_from_fp(HTS_Engine *engine, FILE *fp)
```

- Use: load label from file using given file pointer.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
FILE *fp           - label file pointer
```

↑

HTS_Engine_create_sstream ↑

```
void HTS_Engine_create_sstream(HTS_Engine *engine)
```

- Use: parse label, determine state duration and create state stream.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
```

↑

HTS_Engine_create_pstream ↑

```
void HTS_Engine_create_pstream(HTS_Engine *engine)
```

- Use: create PDF stream using state stream.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
```

↑

HTS_Engine_create_gstream ↑

```
void HTS_Engine_create_gstream(HTS_Engine *engine)
```

- Use: synthesis speech and store generated parameter using PDF stream.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
```

- **Attention!!:** To synthesize speech, you must set stream[0]=spectrum models and spectrum[1]=lf0 models.

↑

HTS_Engine_save_infomation ↑

```
void HTS_Engine_save_infomation(HTS_Engine *engine, FILE *fp)
```

- Use: output trace infomation.

- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
FILE *fp           - output file pointer
```

↑

HTS_Engine_save_label †

```
void HTS_Engine_save_label(HTS_Engine *engine, FILE *fp)
```

- Use: output label with time.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
FILE *fp           - output file pointer
```

↑

HTS_Engine_save_generated_parameter †

```
void HTS_Engine_save_generated_parameter(HTS_Engine *engine, FILE *fp)
```

- Use: output generated parameter.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
FILE *fp           - output file pointer
```

↑

HTS_Engine_save_generated_speech †

```
void HTS_Engine_save_generated_speech(HTS_Engine *engine, FILE *fp)
```

- Use: output generated speech.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
FILE *fp           - output file pointer
```

↑

HTS_Engine_save_riff †

```
void HTS_Engine_save_riff(HTS_Engine *engine, FILE *fp)
```

- Use: output riff format file.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
FILE *fp           - output file pointer
```

↑

HTS_Engine_refresh †

```
void HTS_Engine_refresh(HTS_Engine *engine)
```

- Use: free label, state streams, PDF streams and generated parameter streams.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
```

↑

Free engine †

↑

HTS_Engine_clear †

```
void HTS_Engine_clear(HTS_Engine *engine)
```

- Use: free engine.
- Arguments:

```
HTS_Engine *engine - pointer to HTS_Engine structure
```

↑

Vocoder functions †

↑

Initialize vocoder †

↑

HTS_Vocoder_initialize †

```
void HTS_Vocoder_initialize(HTS_Vocoder *v, const HTS_VocoderType type, const int m, const int stage, const int rate, const int fperiod, int buff_size)
```

- Use: initialize the vocoder.
- Arguments:

```
HTS_Vocoder *v       - pointer to HTS_Vocoder structure
HTS_Vocode_type type - vocoder type
int m                - order of mel-cepstral coefficients
int stage            - Gamma=-1/stage : if stage=0 then Gamma=0
int rate             - sampling frequency
int fperiod          - frame shift
int buff_size        - buffer size for direct audio output
```

↑

Synthesize speech †

↑

HTS_Vocoder_synthesize †

```
void HTS_Vocoder_synthesize(HTS_Vocoder *v, const int m, double lf0, double *spectrum, double alpha, double beta, short *rawdata)
```

- Use: run the vocoder and synthesize waveform.
- Arguments:

```
HTS_Vocoder *v   - HTS_Vocoder structure pointer
int m            - order of spectrum coefficients
```

```
double lf0       - log F0 value
double *spectrum - spectrum coefficients
double alpha     - frequency warping parameter alpha
double beta      - postfiltering parameter beta
short *rawdata   - short pointer to store synthesized waveform
```

↑

HTS_Vocoder_postfilter_mcp ↑

```
void HTS_Vocoder_postfilter_mcp(HTS_Vocoder *v, double *mcp, const int m, double alpha, double beta)
```

- Use: postfilter for mel-cepstrum.
- Arguments:

```
HTS_Vocoder *v - HTS_Vocoder structure pointer
double *mcp    - spectrum coefficients
int m          - order of mel-cepstral coefficients
double alpha   - frequency warping parameter alpha
double beta    - postfiltering parameter beta
```

↑

Free vocoder ↑

↑

HTS_Vocoder_clear ↑

```
void HTS_Vocoder_clear (HTS_Vocoder *v)
```

- Use: free the vocoder.
- Arguments:

```
HTS_Vocoder *v - pointer to HTS_Vocoder structure
```

Last-modified: 2008-02-18 (Mon) 14:23:20 (31d)

Site admin: Heiga ZEN