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)
```

↥

Structures ↥

↥

Models ↥

↥

HTS_ModelSet? ↥

Set of HMMs and duration models.

```
int nstate           - # of HMM states
int lf0stream        - # of stream for F0
int mcpvsize         - vector size for spectrum
int *nlf0pdf         - # of PDFs at each state position (F0)
int *nmcppdf         - # of PDFs at each state position (spectrum)
int ndurpdf          - # of PDFs (duration)
double **durpdf      - array of PDFs (duration)
double ***mcppdf     - array of PDFs (spectrum)
double ****lf0pdf    - array of PDFs (F0)
double weight_interp - weight for model interpolation
```

↥

HTS_Model ↥

A subword HMM in an utterance HMM.

```
char *name                  - name of this HMM (name only)
char *lab                   - label of this HMM (includes other information)
int durpdf                  - duration PDF index
int *lf0pdf                 - F0 PDF indexes
int *mcppdf                 - spectrum PDF indexes
int *dur                    - state durations (frame)
int totaldur                - total duration in this HMM (frame)
double **lf0mean            - mean vectors of F0 PDFs
double **lf0variance        - diag variances of F0 PDFs
double **mcpmean            - mean vectors of spectrum PDFs
double **mcpvariance        - diag variances of spectrum PDFs
HTS_Boolean *voiced         - voiced/unvoiced flags in this HMM
struct _HTS_Model *next     - pointer to the next subword HMM
HTS_Boolean bool_rate       - flag for speaking rate modification
HTS_Boolean bool_dur        - flag for duration modification
HTS_Boolean bool_f0_level - flag for f0 level modification
HTS_Boolean bool_f0_range - flag for f0 range modification
HTS_Boolean bool_volume     - flag for volume modification
HTS_Boolean bool_alpha      - flag for frequency warping modification
double lab_rate             - speaking rate specified in the given label
int lab_dur                 - # of frames specified in the given label
double lab_f0_level         - f0 level specified in the given label
double lab_f0_range         - f0 range specified in the given label
double lab_volume           - volume specified in the given label
double lab_alpha            - frequency warping specified in the given label
```

↥

HTS_UttModel? ↥

An utterance HMM.

```
HTS_Model *mhead - list of subword HMMs (head)
HTS_Model *mtail - list of subword HMMs (tail)
int nModel       - # of subword HMMs in this utterance HMM
int nState       - total # of HMM states in this utterance HMM
int totalframe   - total # of frames in this utterance
```

↥

Trees ↥

↥

HTS_Pattern ↥

List of patterns in a question.

```
char *pat                   - pattern string
struct _HTS_Pattern *next - pointer to the next pattern
```

↥

HTS_Question ↥

List of questions in HTS_TreeSet?.

```
char *qName                  - name of this question
HTS_Pattern *phead           - list of patterns (head)
HTS_Pattern *ptail           - list of patterns (tail)
struct _HTS_Question *next - pointer to the next question
```

↥

HTS_Node ↥

List of tree nodes in a decision tree.

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

↥

HTS_Tree †

List of decision trees in HTS_TreeSet?.

```
int state               - state position of this tree
HTS_Pattern *phead      - list of patterns used in this tree (head)
HTS_Pattern *ptail      - list of patterns used in this tree (tail)
struct _HTS_Tree *next  - pointer to the next tree
HTS_Node *root          - root node of this tree
HTS_Node *leaf          - list of leaf nodes in this tree
```

↥

HTS_TreeSet? †

Set of decision trees.

```
HTS_Question *qhead[HTS_NUMMTYPE] - lists of questions for spectrum, F0 & duration (head)
HTS_Question *qtail[HTS_NUMMTYPE] - lists of questions for spectrum, F0 & duration (tail)
HTS_Tree *thead[HTS_NUMMTYPE]     - lists of trees for spectrum, F0 & duration (head)
HTS_Tree *ttail[HTS_NUMMTYPE]     - lists of trees for spectrum, F0 & duration (tail)
int nTrees[HTS_NUMMTYPE]          - # of trees for spectrum, F0 & duration
```

↥

Parameter generation †

↥

HTS_DWin †

- Window coefficients to calculate dynamic features.

```
int num       - # of windows (static, delta, delta-delta -> 3)
int **width   - width of windows [0..num-1][0(left) 1(right)]
double **coef - window coefficients [0..num-1][width[0]..width[1]]
int maxw[2]   - maximum width [0(left) 1(right)]
int max_L     - maximum width {maxw[0], maxw[1]}
```

↥

HTS_SMatrices †

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

```
double **mseq  - mean vector sequence
double **ivseq - 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 †

- PDF stream used in the speech parameter generation algorithm.

```
int vSize           - vector size of an observation vector (includes static & dynamic features)
int order           - vector size of static features
int T               - vector length (# of frames)
int width           - maximum width of dynamic feature windows
HTS_DWin dw         - dynamic feature windows
double **par        - output parameter vector
HTS_SMatrices sm    - matrices/vectors for parameter generation
```

```
HTS_Boolean *voiced - voiced/unvoiced decision
```

Global settings †

HTS_globalP †

- Global settings.

```
int rate                     - sampling rate (Hz)
int fperiod                  - frame shift (points)
double rho                   - speaking rate
double alpha                 - frequency warping
double beta                  - postfiltering coefficient
double f0_std                - F0 multiply
double f0_mean               - F0 bias
double uv                    - voiced/unvoiced threshold
double length                - total number of frames
HTS_Boolean algnst           - use state-level alignments from labels
HTS_Boolean algnph           - use phone-level alignments from labels
int totaldur                 - total frame
int totalframe               - total frame
int nsample                  - # of samples in a synthesized waveform
int buff_size                - buffer size of audio output device
short *raw_data              - synthesized waveform
HTS_Boolean stored_raw_data - flag to store a synthesized waveform in raw_data
```

HTS_AudioSet? †

- 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
```

HTS_VocoderSet? †

- MLSA filter settings. Usually you don't need to set this structure manually.

```
int fprd                    - frame shift
int iprd                    - interpolation period
int seed                    - seed of random generator
int pd                      - Pade approximation order (4 or 5)
unsigned long next          - temporary variable for random generator
HTS_Boolean gauss           - flag to use Gaussian noise
double p1                   - used in excitation generation
double pc                   - used in excitation generation
double pade[21]             - Pade coefficients
double *ppade               - Pade array
double *c, *cc, *cinc, *d1 - used in the MLSA filter
double rate                 - sampling rate
int sw                      - switch used in random generator
double r1, r2, s            - used in random generator
int x                       - excitation signal
HTS_AudioSet *as            - pointer for audio device
int size                    - buffer size for postfiltering
double *d                   - used in postfiltering
double *g                   - used in postfiltering
double *mc                  - mel-cepstral coefficients
double *cep                 - cepstral coefficients
double *ir                  - impulse response
int o                       - used in postfiltering
int irleng                  - length of impulse response
```

↥

HTS_Engine †

- HTS_Engine itself.

```
HTS_ModelSet *ms    - sets of HMMs and duration models
HTS_TreeSet *ts     - sets of decision trees
HTS_PStream lf0pst - PDF stream for F0
HTS_PStream mcppst - PDF stream for spectrum
HTS_globalP gp      - global settings
int num_interp      - # of models for interpolation
```

↥

Functions †

↥

HTS_Engine functions †

↥

HTS_Engine_initialize †

```
void HTS_Engine_initialize(HTS_Engine *engine);
```

- Use: Initialize HTS_Engine structure.
- Arguments:

```
HTS_Engine *engine - Pointer for HTS_Engine structure
```

- **Attention!!:** To start HTS_Engine module, first you should call this function.

↥

HTS_Engine_load_fn †

```
void HTS_Engine_load_fn(HTS_Engine *engine,
                        char **fn_ms_lf0,char **fn_ms_mcp,char **fn_ms_dur,
                        char **fn_ts_lf0,char **fn_ts_mcp,char **fn_ts_dur,
                        int num_ws_lf0,char **fn_ws_lf0,
                        int num_ws_mcp,char **fn_ws_mcp,
                        double *rate_interp,int num_interp);
```

- Use: Load models/trees/windows from files using given filenames.
- Arguments:

```
HTS_Engine *engine      - HTS_Engine structure
char       **fn_ms_lf0  - F0 PDF file names
char       **fn_ms_mcp  - spectrum PDF file names
char       **fn_ms_dur  - duration PDF file names
char       **fn_ts_lf0  - F0 tree file names
char       **fn_ts_mcp  - spectrum tree file names
char       **fn_ts_dur  - duration tree file names
int        num_ws_lf0   - # of dynamic feature windows for F0
char       **fn_ws_lf0  - dynamic feature window file names for F0
int        num_ws_mcp   - # of dynamic feature windows for spectrum
char       **fn_ws_mcp  - dynamic feature window file names for spectrum
double     *rate_interp - model interpolation rates
int        num_interp   - # of models to be interpolated
```

- **Attention!!**: You should initialize variable *engine* using HTS_Engine_initialize before calling this function. If rate_interp==NULL, interpolation rates of all models are set to the same value.

↥

HTS_Engine_load_fp †

```
void HTS_Engine_load_fp(HTS_Engine *engine,
                        FILE **fp_ms_lf0,FILE **fp_ms_mcp,FILE **fp_ms_dur,
                        FILE **fp_ts_lf0,FILE **fp_ts_mcp,FILE **fp_ts_dur,
                        int num_ws_lf0,FILE **fp_ws_lf0,
                        int num_ws_mcp,FILE **fp_ws_mcp,
                        double *rate_interp,int num_interp);
```

- Use: Load models, trees & windows from files using given file pointers.
- Arguments:

```
HTS_Engine *engine      - HTS_Engine structure
FILE       **fp_ms_lf0  - F0 PDF file pointers
FILE       **fp_ms_mcp  - spectrum PDF file pointers
FILE       **fp_ms_dur  - duration PDF file pointers
FILE       **fp_ts_lf0  - F0 tree file pointers
FILE       **fp_ts_mcp  - spectrum tree file pointers
FILE       **fp_ts_dur  - duration tree file pointers
int        num_ws_lf0   - # of dynamic feature windows for F0
FILE       **fp_ws_lf0  - dynamic feature window file pointers for F0
int        num_ws_mcp   - # of dynamic feature windows for spectrum
FILE       **fp_ws_mcp  - dynamic feature window file pointers for spectrum
double     *rate_interp - model interpolation rates
int        num_interp   - # of models to be interpolated
```

- **Attention!!**: You should initialize variable *engine* using HTS_Engine_initialize before calling this function. If rate_interp==NULL, interpolation rates of all models are set to the same value.

↥

HTS_Engine_process ↑

```
void HTS_Engine_process(HTS_Engine *engine,HTS_Model *model,
                        FILE *wavfp, FILE *rawfp, FILE *lf0fp,
                        FILE *mcpfp, FILE *durfp, FILE *tracefp,
                        double *fr_f0, double *fr_power);
```

- Use: run HMM-based speech synthesis.
- Arguments:

```
HTS_Engine  *engine    - HTS_Engine structure
HTS_Model   *model     - HTS_Model structure
FILE        *wavfp     - file pointer for RIFF waveform output
FILE        *rawfp     - file pointer for raw audio output
FILE        *lf0fp     - file pointer for generated (log) F0 sequence
FILE        *mcpfp     - file pointer for generated spectrum (mel-cepstrum)
FILE        *durfp     - file pointer for predicted durations
FILE        *tracefp   - file pointer for trace information
double      *fr_f0     - Given F0 values
double      *fr_power  - Given powers
```

- **Attention!!**: You can give F0 values or powers predicted by other modules via fr_f0 and fr_power.

↥

HTS_Engine_refresh ↑

```
void HTS_Engine_refresh(HTS_Engine *engine, Model *mhead);
```

- Use: free model list.
- Arguments:

```
HTS_Engine *engine - HTS_Engine structure
Model      *mhead  - HTS_Model structure
```

- **Attention!!**: You should call this function if you want to run HTS_Engine_process multiple times.

↥

HTS_Engine_clear ↑

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

- Use: free memory.

- Arguments:

```
HTS_Engine *engine - HTS_Engine structure
```

↥

HTS_Engine setting function †

↥

HTS_Engine_set_sampling_rate †

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

- Use: set sampling frequency.
- Arguments:

```
HTS_Engine *engine - HTS_Engine structure
int         i       - sampling frequency (Hz), 0 < 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 - HTS_Engine structure
int         i       - frame shift (point), 0 < 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 - HTS_Engine structure
double      f       - alpha, 0.0 <= f <= 1.0
```

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

↥

HTS_Engine_set_beta †

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

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

```
HTS_Engine *engine - 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 speaking rate control parameter rho.

- Arguments:

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

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

↑

HTS_Engine_set_f0_std †

```
void HTS_Engine_set_f0_std(HTS_Engine *engine,double f);
```

- Use: set a parameter to be multiplied to generated F0 values
- Arguments:

  ```
  HTS_Engine *engine - HTS_Engine structure
  double     f       - F0 multiply value, 0.0 <= f <= 5.0
  ```

- **Attention!!:** Default value is 1.0.

↑

HTS_Engine_set_f0_mean †

```
void HTS_Engine_set_f0_mean(HTS_Engine *engine,double f);
```

- Use: set a parameter to be added to generated F0 values
- Arguments:

  ```
  HTS_Engine *engine - HTS_Engine structure
  double     f       - F0 bias value, 0.0 <= f <= 100.0
  ```

- **Attention!!:** Default value is 0.0.

↑

HTS_Engine_set_uv †

```
void HTS_Engine_set_uv(HTS_Engine *engine,double f);
```

- Use: set voiced/unvoiced threshold.
- Arguments:

  ```
  HTS_Engine *engine - HTS_Engine structure
  double     f       - voiced/unvoiced threshold, 0.0 <= f <= 1.0
  ```

- **Attention!!:** Default value 0.5.

↑

HTS_Engine_set_length †

```
void HTS_Engine_set_length(HTS_Engine *engine,double f);
```

- Use: set total length of utterance in second
- Arguments:

  ```
  HTS_Engine *engine - HTS_Engine structure
  double     f       - total length of utterance (second), 0.0 <= f <= 30.0
  ```

- **Attention!!:** Default value is 0.0 (using predicted durations by state duration models).

↑

HTS_Engine_set_algnst †

```
void HTS_Engine_set_algnst(HTS_Engine *engine,HTS_Boolean i);
```

- Use: set flag whether state-level alignments from given labels is used or not.
- Arguments:

```
HTS_Engine  *engine - HTS_Engine structure
HTS_Boolean i       - flag whether state-level alignments from given labels is used or not
```

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

↑

HTS_Engine_set_algnph †

```
void HTS_Engine_set_algnph(HTS_Engine *engine,HTS_Boolean i);
```

- Use: set flag whether phone-level alignments from given labels is used or not.
- Arguments:

```
HTS_Engine  *engine - HTS_Engine structure
HTS_Boolean i       - flag whether phone-level alignments from given labels is used or not
```

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

↑

HTS_Engine_set_buff_size †

```
void HTS_Engine_set_buff_size(HTS_Engine *engine, int i);
```

- Use: set buffer size for audio device.
- Arguments:

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

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

↑

HTS_Engine_set_stored_raw_data †

```
void HTS_Engine_set_stored_raw_data(HTS_Engine *engine,HTS_Boolean i);
```

- Use: set flag whether a synthesized waveform is stored in an array.
- Arguments:

```
HTS_Engine  *engine - HTS_Engine structure
HTS_Boolean i       - if TRUE, synthesized waveform is stored
```

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

↑

HTS_Engine_get_sampling_rate †

```
int HTS_Engine_get_sampling_rate(HTS_Engine *engine);
```

- Use: get sampling frequency.
- Return value: sampling frequency (Hz)
- Arguments:

```
HTS_Engine *engine - HTS_Engine structure
```

↑

HTS_Engine_get_total_dur †

```
int HTS_Engine_get_total_dur(HTS_Engine *engine);
```

- Use: get total durations
- Return value: get total durations
- Arguments:

```
HTS_Engine *engine - HTS_Engine structure
```

↑

HTS_Engine_get_total_frame ↑

```
int HTS_Engine_get_total_frame(HTS_Engine *engine);
```

- Use: get total # of frame.
- Return value: total # of frames.
- Arguments:

```
HTS_Engine *engine - HTS_Engine structure
```

↑

HTS_Engine_get_nsample ↑

```
int HTS_Engine_get_nsample(HTS_Engine *engine);
```

- Use: get # of samples in a synthesized waveform.
- Return value: # of samples.
- Arguments:

```
HTS_Engine *engine - HTS_Engine structure
```

↑

HTS_Engine_get_pros_len ↑

```
int HTS_Engine_get_pros_len(HTS_Engine *engine);
```

- Use: get spectrum, F0 data length.
- Return value: data length.
- Arguments:

```
HTS_Engine *engine - HTS_Engine structure
```

↑

HTS_Engine_get_pros ↑

```
HTS_Boolean HTS_Engine_get_pros(HTS_Engine *engine,int len,
                                double *f0_data,double *power_data);
```

- Use: get mcp, lf0 data array.
- Return value: if len==data length, return TRUE.
- Arguments:

```
HTS_Engine *engine     - HTS_Engine structure
int        len         - data length
double     *f0_data    - generated F0 sequence
double     *power_data - generated spectrum (mel-cepstrum) sequence
```

↑

HTS_Engine_get_stored_raw_data ↑

```
short *HTS_Engine_get_stored_raw_data(HTS_Engine *engine);
```

- Use: get stored raw data.
- Return value: short array which contains a synthesized waveform.
- Arguments:

```
HTS_Engine *engine - HTS_Engine structure
```

↑

HTS_Model function ↑

↑

HTS_Model_load_from_labfp ↑

```
void HTS_Model_load_from_labfp(HTS_Model *mhead,FILE *labfp);
```

- Use: load model list from label file pointer.
- Arguments:

  ```
  HTS_Model *mhead - HTS_Model structure pointer
  FILE      *labfp - label file pointer
  ```

↥

HTS_Model_load_from_labfn †

```
void HTS_Model_load_from_labfn (HTS_Model *mhead,char *fn);
```

- Use: load model list from label file name.
- Arguments:

  ```
  HTS_Model *mhead - HTS_Model structure pointer
  char      *fn    - label file name
  ```

↥

HTS_Model_load_from_string †

```
void HTS_Model_load_from_string (HTS_Model *mhead,char *labdata);
```

- Use: load model list from string.
- Arguments:

  ```
  HTS_Model *mhead   - HTS_Model structure pointer
  char      *labdata - label string
  ```

↥

HTS_Model_load_from_string_list †

```
void HTS_Model_load_from_string_list (HTS_Model *mhead,char **labdata,int size);
```

- Use: load model list from string list.
- Arguments:

  ```
  HTS_Model *mhead    - HTS_Model structure pointer
  char      **labdata - label string list
  int       size      - label string list size
  ```

↥

HTS_Model setting function †

↥

HTS_Model_set_rate †

```
void HTS_Model_set_rate(HTS_Model *m, int i, double f);
```

- Use: set speaking rate for the i-th subword HMM.
- Arguments:

  ```
  HTS_Model *m - HTS_Model list
  int       i  - index of subword HMM to be used in HTS_Model list m
  double    f  - speaking rate
  ```

- **Attention!!:** 0.2 <= f.

↥

HTS_Model_set_dur †

```
void HTS_Model_set_dur(Model *m, int i, int d);
```

- Use: set phone-level duration for the i-th subword HMM.

- Arguments:

```
HTS_Model *m - HTS_Model list
int       i  - index of subword HMM to be used in HTS_Model list m
int       d  - # of frames (duration)
```

- **Attention!!:** 0 < d.

↑

HTS_Model_set_f0_level ↑

```
void HTS_Model_set_f0_level (Model *m, int i, double f);
```

- Use: set F0 level for the i-th subword HMM.
- Arguments:

```
HTS_Model  *m - HTS_Model list
int         i  - index of subword HMM to be used in HTS_Model list m
double      f  - F0 bias
```

- **Attention!!:** 0.1 <= f.

↑

HTS_Model_set_f0_range ↑

```
void HTS_Model_set_f0_range(Model *m, int i, double f);
```

- Use: set F0 range for the i-th subword HMM.
- Arguments:

```
HTS_Model *m - HTS_Model list
int       i  - index of subword HMM to be used in HTS_Model list m
double    f  - F0 range
```

- **Attention!!:** 0.0 <= f. Default value is 0.0.

↑

HTS_Model_set_volume ↑

```
void HTS_Model_set_volume(Model *m, int i, double f);
```

- Use: set volume for the i-th subword HMM.
- Arguments:

```
HTS_Model *m - HTS_Model list
int       i  - index of subword HMM to be used in HTS_Model list m
double    f  - volume
```

- **Attention!!:** 0.01 <= f.

↑

HTS_Model_set_alpha ↑

```
void HTS_Model_set_alpha(Model *m, int i, double f);
```

- Use: set frequency warping parameter alpha for the i-th subword HMM.
- Arguments:

```
HTS_Model *m - HTS_Model list
int       i  - index of subword HMM to be used in HTS_Model list m
double    f  - frequency warping parameter alpha
```

- **Attention!!:** 0.0 <= f.

↑

HTS_VocoderSet? function ↑

↑

HTS_VocoderSet_initialize ↑

```
void HTS_VocoderSet_initialize (HTS_VocoderSet *vs, const int m,
                                const int rate, const int fperiod, int buff_size);
```

- Use: initialize the MLSA filter.
- Arguments:

```
HTS_VocodetSet *vs       - HTS_VocodetSet structure
const int      m         - order of mel-cepstral coefficients
const int      rate      - sampling frequency (Hz)
const int      fperiod   - frame shift (point)
int            buff_size - buffer size for direct audio output
```

↑

HTS_VocoderSet_synthesize ↑

```
void HTS_VocoderSet_synthesize (HTS_VocoderSet *vs, const int m,
                                double p, double *mc, double alpha, double beta,
                                FILE *wavfp, FILE *rawfp, short *rawdata);
```

- Use: run the MLSA filter and synthesize waveform.
- Arguments:

```
HTS_VocoderSet *vs      - HTS_VocodetSet structure
const int      m        - order of mel-cepstral coefficients
double         p        - F0 value
double         *mc      - mel-cepstral coefficients
double         alpha    - frequency warping parameter alpha
double         beta     - postfiltering parameter beta
FILE           *wavfp   - file pointer to store synthesized waveform in RIFF format
FILE           *rawfp   - file pointer to store synthesized waveform in raw audio
```