

## Previous topic

[gnuradio.video\\_sdl](#)

## Next topic

[gnuradio.wavelet](#)

## This Page

[Show Source](#)

## Quick search

Go

Enter search terms or a module, class or function name.

## gnuradio.vocoder

This is the gr-vocoder package. This package includes the various vocoder blocks in GNU Radio.

`gnuradio.vocoder.alaw_decode_bs()` → `alaw_decode_bs_sptr`

This block performs alaw audio decoding.

Constructor Specific Documentation:

Make alaw decoder block.

`alaw_decode_bs_sptr.active_thread_priority(alaw_decode_bs_sptr self) → int`

`alaw_decode_bs_sptr.declare_sample_delay(alaw_decode_bs_sptr self, int which, int delay)`

`declare_sample_delay(alaw_decode_bs_sptr self, unsigned int delay)`

`alaw_decode_bs_sptr.message_subscribers(alaw_decode_bs_sptr self, swig_int_ptr which_port) → swig_int_ptr`

`alaw_decode_bs_sptr.min_noutput_items(alaw_decode_bs_sptr self) → int`

`alaw_decode_bs_sptr.pc_input_buffers_full_avg(alaw_decode_bs_sptr self, int which) → float`

`pc_input_buffers_full_avg(alaw_decode_bs_sptr self) → pmt_vector_float`

`alaw_decode_bs_sptr.pc_noutput_items_avg(alaw_decode_bs_sptr self) → float`

`alaw_decode_bs_sptr.pc_nproduced_avg(alaw_decode_bs_sptr self) → float`

`alaw_decode_bs_sptr.pc_output_buffers_full_avg(alaw_decode_bs_sptr self, int which) → float`

`pc_output_buffers_full_avg(alaw_decode_bs_sptr self) → pmt_vector_float`

`alaw_decode_bs_sptr.pc_throughput_avg(alaw_decode_bs_sptr self) → float`

`alaw_decode_bs_sptr.pc_work_time_avg(alaw_decode_bs_sptr self) → float`

`alaw_decode_bs_sptr.pc_work_time_total(alaw_decode_bs_sptr self) → float`

`alaw_decode_bs_sptr.sample_delay(alaw_decode_bs_sptr self, int which) → unsigned int`

`alaw_decode_bs_sptr.set_min_noutput_items(alaw_decode_bs_sptr self, int m)`

`alaw_decode_bs_sptr.set_thread_priority(alaw_decode_bs_sptr self, int priority) → int`

`alaw_decode_bs_sptr.thread_priority(alaw_decode_bs_sptr self) → int`

`gnuradio.vocoder.alaw_encode_sb()` → `alaw_encode_sb_sptr`

This block performs g.711 alaw audio encoding.

Constructor Specific Documentation:

Make alaw encoder block.

`alaw_encode_sb_sptr.active_thread_priority(alaw_encode_sb_sptr self) → int`

```
alaw_encode_sb_sptr.declare_sample_delay(alaw_encode_sb_sptr self, int which, int delay)
```

```
declare_sample_delay(alaw_encode_sb_sptr self, unsigned int delay)
```

```
alaw_encode_sb_sptr.message_subscribers(alaw_encode_sb_sptr self, swig_int_ptr which_port) → swig_int_ptr
```

```
alaw_encode_sb_sptr.min_noutput_items(alaw_encode_sb_sptr self) → int
```

```
alaw_encode_sb_sptr.pc_input_buffers_full_avg(alaw_encode_sb_sptr self, int which) → float
```

```
pc_input_buffers_full_avg(alaw_encode_sb_sptr self) -> pmt_vector_float
```

```
alaw_encode_sb_sptr.pc_noutput_items_avg(alaw_encode_sb_sptr self) → float
```

```
alaw_encode_sb_sptr.pc_nproduced_avg(alaw_encode_sb_sptr self) → float
```

```
alaw_encode_sb_sptr.pc_output_buffers_full_avg(alaw_encode_sb_sptr self, int which) → float
```

```
pc_output_buffers_full_avg(alaw_encode_sb_sptr self) -> pmt_vector_float
```

```
alaw_encode_sb_sptr.pc_throughput_avg(alaw_encode_sb_sptr self) → float
```

```
alaw_encode_sb_sptr.pc_work_time_avg(alaw_encode_sb_sptr self) → float
```

```
alaw_encode_sb_sptr.pc_work_time_total(alaw_encode_sb_sptr self) → float
```

```
alaw_encode_sb_sptr.sample_delay(alaw_encode_sb_sptr self, int which) → unsigned int
```

```
alaw_encode_sb_sptr.set_min_noutput_items(alaw_encode_sb_sptr self, int m)
```

```
alaw_encode_sb_sptr.set_thread_priority(alaw_encode_sb_sptr self, int priority) → int
```

```
alaw_encode_sb_sptr.thread_priority(alaw_encode_sb_sptr self) → int
```

```
gnuradio.vocoder.codec2_decode_ps(int mode) → codec2_decode_ps_sptr
```

CODEC2 Vocoder Decoder

Input: A vector of unpacked bits forming a Codec2 frame.

Output: 16-bit short values of an audio signal with sampling rate 8 kHz.

See also `gr::vocoder::codec2_encode_sp`.

Constructor Specific Documentation:

Make Codec2 decoder block.

**Parameters:** `mode` – Encoded bit rate/mode

```
codec2_decode_ps_sptr.active_thread_priority(codec2_decode_ps_sptr self) → int
```

```
codec2_decode_ps_sptr.declare_sample_delay(codec2_decode_ps_sptr self, int which, int delay)
```

```
declare_sample_delay(codec2_decode_ps_sptr self, unsigned int delay)
```

```
codec2_decode_ps_sptr.message_subscribers(codec2_decode_ps_sptr self, swig_int_ptr which_port) → swig_int_ptr
```

```

codec2_decode_ps_sptr.min_noutput_items(codec2_decode_ps_sptr self) →
int

codec2_decode_ps_sptr.pc_input_buffers_full_avg(codec2_decode_ps_sptr
self, int which) → float
    pc_input_buffers_full_avg(codec2_decode_ps_sptr self) -> pmt_vector_float

codec2_decode_ps_sptr.pc_noutput_items_avg(codec2_decode_ps_sptr self) →
float

codec2_decode_ps_sptr.pc_nproduced_avg(codec2_decode_ps_sptr self) →
float

codec2_decode_ps_sptr.pc_output_buffers_full_avg(codec2_decode_ps_sptr
self, int which) → float
    pc_output_buffers_full_avg(codec2_decode_ps_sptr self) -> pmt_vector_float

codec2_decode_ps_sptr.pc_throughput_avg(codec2_decode_ps_sptr self) →
float

codec2_decode_ps_sptr.pc_work_time_avg(codec2_decode_ps_sptr self) →
float

codec2_decode_ps_sptr.pc_work_time_total(codec2_decode_ps_sptr self) →
float

codec2_decode_ps_sptr.sample_delay(codec2_decode_ps_sptr self, int which) →
unsigned int

codec2_decode_ps_sptr.set_min_noutput_items(codec2_decode_ps_sptr self,
int m)

codec2_decode_ps_sptr.set_thread_priority(codec2_decode_ps_sptr self, int
priority) → int

codec2_decode_ps_sptr.thread_priority(codec2_decode_ps_sptr self) → int

```

```

gnuradio.vocoder.codec2_encode_sp(int mode) → codec2_encode_sp_sptr

```

CODEC2 Vocoder Encoder

Input: Speech (audio) signal as 16-bit shorts, sampling rate 8 kHz.

Output: Vector of unpacked bits, forming one Codec2 frame, per 160 input samples (in 2400 and 3200 bps modes) or per 320 input samples (in 1200, 1300, 1400 and 1600 bps modes).

Constructor Specific Documentation:

Make Codec2 encoder block.

**Parameters:** **mode** – Encoded bit rate/mode

```

codec2_encode_sp_sptr.active_thread_priority(codec2_encode_sp_sptr self)
→ int

codec2_encode_sp_sptr.declare_sample_delay(codec2_encode_sp_sptr self, int
which, int delay)
    declare_sample_delay(codec2_encode_sp_sptr self, unsigned int delay)

codec2_encode_sp_sptr.message_subscribers(codec2_encode_sp_sptr self,
swig_int_ptr which_port) → swig_int_ptr

codec2_encode_sp_sptr.min_noutput_items(codec2_encode_sp_sptr self) →
int

```

```

codec2_encode_sp_sptr.pc_input_buffers_full_avg(codec2_encode_sp_sptr
self, int which) → float
    pc_input_buffers_full_avg(codec2_encode_sp_sptr self) -> pmt_vector_float

codec2_encode_sp_sptr.pc_noutput_items_avg(codec2_encode_sp_sptr self) →
float

codec2_encode_sp_sptr.pc_nproduced_avg(codec2_encode_sp_sptr self) →
float

codec2_encode_sp_sptr.pc_output_buffers_full_avg(codec2_encode_sp_sptr
self, int which) → float
    pc_output_buffers_full_avg(codec2_encode_sp_sptr self) -> pmt_vector_float

codec2_encode_sp_sptr.pc_throughput_avg(codec2_encode_sp_sptr self) →
float

codec2_encode_sp_sptr.pc_work_time_avg(codec2_encode_sp_sptr self) →
float

codec2_encode_sp_sptr.pc_work_time_total(codec2_encode_sp_sptr self) →
float

codec2_encode_sp_sptr.sample_delay(codec2_encode_sp_sptr self, int which) →
unsigned int

codec2_encode_sp_sptr.set_min_noutput_items(codec2_encode_sp_sptr self,
int m)

codec2_encode_sp_sptr.set_thread_priority(codec2_encode_sp_sptr self, int
priority) → int

codec2_encode_sp_sptr.thread_priority(codec2_encode_sp_sptr self) → int

```

```

gnuradio.vocoder.cvsd_decode_bs(short min_step=10, short max_step=1280, double
step_decay=0.9990234375, double accum_decay=0.96875, int K=32, int J=4, short
pos_accum_max=32767, short neg_accum_max=-32767) → cvsd_decode_bs_sptr

```

This block performs CVSD audio decoding. Its design and implementation is modeled after the CVSD encoder/decoder specifications defined in the Bluetooth standard.

CVSD is a method for encoding speech that seeks to reduce the bandwidth required for digital voice transmission. CVSD takes advantage of strong correlation between samples, quantizing the difference in amplitude between two consecutive samples. This difference requires fewer quantization levels as compared to other methods that quantize the actual amplitude level, reducing the bandwidth. CVSD employs a two level quantizer (one bit) and an adaptive algorithm that allows for continuous step size adjustment.

The coder can represent low amplitude signals with accuracy without sacrificing performance on large amplitude signals, a trade off that occurs in some non-adaptive modulations.

The CVSD decoder effectively provides 1-to-8 decompression. More specifically, for each incoming input bit, the decoder outputs one audio sample. If the input is a "1" bit, the internal reference is increased appropriately and then outputted as the next estimated audio sample. If the input is a "0" bit, the internal reference is decreased appropriately and then likewise outputted as the next estimated audio sample. Grouping 8 input bits together, the encoder essentially produces 8 output audio samples for every one input byte.

This decoder requires that output audio samples are 2-byte short signed integers. The result bandwidth conversion, therefore, is 1 byte of encoded audio data to 16 output bytes of raw audio data.

The CVSD decoder module must be post-fixed by a down-converter to under-sample

the audio data after decoding. The Bluetooth standard specifically calls for a 8-to-1 decimating down-converter. This is required so that so that output sampling rate equals the original input sampling rate present before the encoder. In all cases, the output down-converter rate must be the inverse of the input up-converter rate before the CVSD encoder.

References:

Constructor Specific Documentation:

Constructor parameters to initialize the CVSD decoder. The default values are modeled after the Bluetooth standard and should not be changed, except by an advanced user.

**Parameters:**

- **min\_step** – Minimum step size used to update the internal reference. Default: “10”
- **max\_step** – Maximum step size used to update the internal reference. Default: “1280”
- **step\_decay** – Decay factor applied to step size when there is not a run of J output 1s or 0s. Default: “0.9990234375” (i.e. 1-1/1024)
- **accum\_decay** – Decay factor applied to the internal reference during every iteration of the codec. Default: “0.96875” (i.e. 1-1/32)
- **K** – Size of shift register; the number of output bits remembered by codec (must be  $\leq 32$ ). Default: “32”
- **J** – Number of bits in the shift register that are equal; i.e. the size of a run of 1s, 0s. Default: “4”
- **pos\_accum\_max** – Maximum integer value allowed for the internal reference. Default: “32767” ( $2^{15} - 1$  or MAXSHORT)
- **neg\_accum\_max** – Minimum integer value allowed for the internal reference. Default: “-32767” ( $-2^{15} + 1$  or MINSHORT+1)

`cvsd_decode_bs_sptr.j(cvsd_decode_bs_sptr self) → int`

`cvsd_decode_bs_sptr.k(cvsd_decode_bs_sptr self) → int`

`cvsd_decode_bs_sptr.accum_decay(cvsd_decode_bs_sptr self) → double`

`cvsd_decode_bs_sptr.active_thread_priority(cvsd_decode_bs_sptr self) → int`

`cvsd_decode_bs_sptr.declare_sample_delay(cvsd_decode_bs_sptr self, int which, int delay)`

`declare_sample_delay(cvsd_decode_bs_sptr self, unsigned int delay)`

`cvsd_decode_bs_sptr.max_step(cvsd_decode_bs_sptr self) → short`

`cvsd_decode_bs_sptr.message_subscribers(cvsd_decode_bs_sptr self, swig_int_ptr which_port) → swig_int_ptr`

`cvsd_decode_bs_sptr.min_noutput_items(cvsd_decode_bs_sptr self) → int`

`cvsd_decode_bs_sptr.min_step(cvsd_decode_bs_sptr self) → short`

`cvsd_decode_bs_sptr.neg_accum_max(cvsd_decode_bs_sptr self) → short`

`cvsd_decode_bs_sptr.pc_input_buffers_full_avg(cvsd_decode_bs_sptr self, int which) → float`

`pc_input_buffers_full_avg(cvsd_decode_bs_sptr self) → pmt_vector_float`

`cvsd_decode_bs_sptr.pc_noutput_items_avg(cvsd_decode_bs_sptr self) → float`

`cvsd_decode_bs_sptr.pc_nproduced_avg(cvsd_decode_bs_sptr self) → float`

`cvsd_decode_bs_sptr.pc_output_buffers_full_avg(cvsd_decode_bs_sptr self) → float`

*self, int which*) → float

`pc_output_buffers_full_avg(cvsd_decode_bs_sptr self) -> pmt_vector_float`

`cvsd_decode_bs_sptr.pc_throughput_avg(cvsd_decode_bs_sptr self) → float`

`cvsd_decode_bs_sptr.pc_work_time_avg(cvsd_decode_bs_sptr self) → float`

`cvsd_decode_bs_sptr.pc_work_time_total(cvsd_decode_bs_sptr self) → float`

`cvsd_decode_bs_sptr.pos_accum_max(cvsd_decode_bs_sptr self) → short`

`cvsd_decode_bs_sptr.sample_delay(cvsd_decode_bs_sptr self, int which) → unsigned int`

`cvsd_decode_bs_sptr.set_min_noutput_items(cvsd_decode_bs_sptr self, int m)`

`cvsd_decode_bs_sptr.set_thread_priority(cvsd_decode_bs_sptr self, int priority) → int`

`cvsd_decode_bs_sptr.step_decay(cvsd_decode_bs_sptr self) → double`

`cvsd_decode_bs_sptr.thread_priority(cvsd_decode_bs_sptr self) → int`

`gnuradio.vocoder.cvsd_encode_sb(short min_step=10, short max_step=1280, double step_decay=0.9990234375, double accum_decay=0.96875, int K=32, int J=4, short pos_accum_max=32767, short neg_accum_max=-32767) → cvsd_encode_sb_sptr`

This block performs CVSD audio encoding. Its design and implementation is modeled after the CVSD encoder/decoder specifications defined in the Bluetooth standard.

CVSD is a method for encoding speech that seeks to reduce the bandwidth required for digital voice transmission. CVSD takes advantage of strong correlation between samples, quantizing the difference in amplitude between two consecutive samples. This difference requires fewer quantization levels as compared to other methods that quantize the actual amplitude level, reducing the bandwidth. CVSD employs a two level quantizer (one bit) and an adaptive algorithm that allows for continuous step size adjustment.

The coder can represent low amplitude signals with accuracy without sacrificing performance on large amplitude signals, a trade off that occurs in some non-adaptive modulations.

The CVSD encoder effectively provides 8-to-1 compression. More specifically, each incoming audio sample is compared to an internal reference value. If the input is greater or equal to the reference, the encoder outputs a “1” bit. If the input is less than the reference, the encoder outputs a “0” bit. The reference value is then updated accordingly based on the frequency of outputted “1” or “0” bits. By grouping 8 outputs bits together, the encoder essentially produce one output byte for every 8 input audio samples.

This encoder requires that input audio samples are 2-byte short signed integers. The result bandwidth conversion, therefore, is 16 input bytes of raw audio data to 1 output byte of encoded audio data.

The CVSD encoder module must be prefixed by an up-converter to over-sample the audio data prior to encoding. The Bluetooth standard specifically calls for a 1-to-8 interpolating up-converter. While this reduces the overall compression of the codec, this is required so that the encoder can accurately compute the slope between adjacent audio samples and correctly update its internal reference value.

References:

Constructor Specific Documentation:

Constructor parameters to initialize the CVSD encoder. The default values are modeled after the Bluetooth standard and should not be changed except by an

advanced user.

- Parameters:**
- **min\_step** – Minimum step size used to update the internal reference. Default: “10”
  - **max\_step** – Maximum step size used to update the internal reference. Default: “1280”
  - **step\_decay** – Decay factor applied to step size when there is not a run of J output 1s or 0s. Default: “0.9990234375” (i.e. 1-1/1024)
  - **accum\_decay** – Decay factor applied to the internal reference during every iteration of the codec. Default: “0.96875” (i.e. 1-1/32)
  - **K** – Size of shift register; the number of output bits remembered by codec (must be  $\leq$  to 32). Default: “32”
  - **J** – Number of bits in the shift register that are equal; i.e. the size of a run of 1s, 0s. Default: “4”
  - **pos\_accum\_max** – Maximum integer value allowed for the internal reference. Default: “32767” ( $2^{15} - 1$  or MAXSHORT)
  - **neg\_accum\_max** – Minimum integer value allowed for the internal reference. Default: “-32767” ( $-2^{15} + 1$  or MINSHORT+1)

`cvsd_encode_sb_sptr.J(cvsd_encode_sb_sptr self) → int`

`cvsd_encode_sb_sptr.K(cvsd_encode_sb_sptr self) → int`

`cvsd_encode_sb_sptr.accum_decay(cvsd_encode_sb_sptr self) → double`

`cvsd_encode_sb_sptr.active_thread_priority(cvsd_encode_sb_sptr self) → int`

`cvsd_encode_sb_sptr.declare_sample_delay(cvsd_encode_sb_sptr self, int which, int delay)`

`declare_sample_delay(cvsd_encode_sb_sptr self, unsigned int delay)`

`cvsd_encode_sb_sptr.max_step(cvsd_encode_sb_sptr self) → short`

`cvsd_encode_sb_sptr.message_subscribers(cvsd_encode_sb_sptr self, swig_int_ptr which_port) → swig_int_ptr`

`cvsd_encode_sb_sptr.min_noutput_items(cvsd_encode_sb_sptr self) → int`

`cvsd_encode_sb_sptr.min_step(cvsd_encode_sb_sptr self) → short`

`cvsd_encode_sb_sptr.neg_accum_max(cvsd_encode_sb_sptr self) → short`

`cvsd_encode_sb_sptr.pc_input_buffers_full_avg(cvsd_encode_sb_sptr self, int which) → float`

`pc_input_buffers_full_avg(cvsd_encode_sb_sptr self) → pmt_vector_float`

`cvsd_encode_sb_sptr.pc_noutput_items_avg(cvsd_encode_sb_sptr self) → float`

`cvsd_encode_sb_sptr.pc_nproduced_avg(cvsd_encode_sb_sptr self) → float`

`cvsd_encode_sb_sptr.pc_output_buffers_full_avg(cvsd_encode_sb_sptr self, int which) → float`

`pc_output_buffers_full_avg(cvsd_encode_sb_sptr self) → pmt_vector_float`

`cvsd_encode_sb_sptr.pc_throughput_avg(cvsd_encode_sb_sptr self) → float`

`cvsd_encode_sb_sptr.pc_work_time_avg(cvsd_encode_sb_sptr self) → float`

`cvsd_encode_sb_sptr.pc_work_time_total(cvsd_encode_sb_sptr self) → float`

`cvsd_encode_sb_sptr.pos_accum_max(cvsd_encode_sb_sptr self) → short`

`cvsd_encode_sb_sptr.sample_delay(cvsd_encode_sb_sptr self, int which) →`

unsigned int

`cvsd_encode_sb_sptr.set_min_noutput_items(cvsd_encode_sb_sptr self, int m)`

`cvsd_encode_sb_sptr.set_thread_priority(cvsd_encode_sb_sptr self, int priority) → int`

`cvsd_encode_sb_sptr.step_decay(cvsd_encode_sb_sptr self) → double`

`cvsd_encode_sb_sptr.thread_priority(cvsd_encode_sb_sptr self) → int`

`gnuradio.vocoder.g721_decode_bs() → g721_decode_bs_sptr`

This block performs g721 audio decoding.

Constructor Specific Documentation:

Make G721 decoder block.

`g721_decode_bs_sptr.active_thread_priority(g721_decode_bs_sptr self) → int`

`g721_decode_bs_sptr.declare_sample_delay(g721_decode_bs_sptr self, int which, int delay)`

`declare_sample_delay(g721_decode_bs_sptr self, unsigned int delay)`

`g721_decode_bs_sptr.message_subscribers(g721_decode_bs_sptr self, swig_int_ptr which_port) → swig_int_ptr`

`g721_decode_bs_sptr.min_noutput_items(g721_decode_bs_sptr self) → int`

`g721_decode_bs_sptr.pc_input_buffers_full_avg(g721_decode_bs_sptr self, int which) → float`

`pc_input_buffers_full_avg(g721_decode_bs_sptr self) → pmt_vector_float`

`g721_decode_bs_sptr.pc_noutput_items_avg(g721_decode_bs_sptr self) → float`

`g721_decode_bs_sptr.pc_nproduced_avg(g721_decode_bs_sptr self) → float`

`g721_decode_bs_sptr.pc_output_buffers_full_avg(g721_decode_bs_sptr self, int which) → float`

`pc_output_buffers_full_avg(g721_decode_bs_sptr self) → pmt_vector_float`

`g721_decode_bs_sptr.pc_throughput_avg(g721_decode_bs_sptr self) → float`

`g721_decode_bs_sptr.pc_work_time_avg(g721_decode_bs_sptr self) → float`

`g721_decode_bs_sptr.pc_work_time_total(g721_decode_bs_sptr self) → float`

`g721_decode_bs_sptr.sample_delay(g721_decode_bs_sptr self, int which) → unsigned int`

`g721_decode_bs_sptr.set_min_noutput_items(g721_decode_bs_sptr self, int m)`

`g721_decode_bs_sptr.set_thread_priority(g721_decode_bs_sptr self, int priority) → int`

`g721_decode_bs_sptr.thread_priority(g721_decode_bs_sptr self) → int`

`gnuradio.vocoder.g721_encode_sb() → g721_encode_sb_sptr`

This block performs g721 audio encoding.

Constructor Specific Documentation:

Make G721 encoder block.

```
g721_encode_sb_sptr.active_thread_priority(g721_encode_sb_sptr self) → int  
  
g721_encode_sb_sptr.declare_sample_delay(g721_encode_sb_sptr self, int which, int delay)  
    declare_sample_delay(g721_encode_sb_sptr self, unsigned int delay)  
  
g721_encode_sb_sptr.message_subscribers(g721_encode_sb_sptr self, swig_int_ptr which_port) → swig_int_ptr  
  
g721_encode_sb_sptr.min_noutput_items(g721_encode_sb_sptr self) → int  
  
g721_encode_sb_sptr.pc_input_buffers_full_avg(g721_encode_sb_sptr self, int which) → float  
    pc_input_buffers_full_avg(g721_encode_sb_sptr self) → pmt_vector_float  
  
g721_encode_sb_sptr.pc_noutput_items_avg(g721_encode_sb_sptr self) → float  
  
g721_encode_sb_sptr.pc_nproduced_avg(g721_encode_sb_sptr self) → float  
  
g721_encode_sb_sptr.pc_output_buffers_full_avg(g721_encode_sb_sptr self, int which) → float  
    pc_output_buffers_full_avg(g721_encode_sb_sptr self) → pmt_vector_float  
  
g721_encode_sb_sptr.pc_throughput_avg(g721_encode_sb_sptr self) → float  
  
g721_encode_sb_sptr.pc_work_time_avg(g721_encode_sb_sptr self) → float  
  
g721_encode_sb_sptr.pc_work_time_total(g721_encode_sb_sptr self) → float  
  
g721_encode_sb_sptr.sample_delay(g721_encode_sb_sptr self, int which) → unsigned int  
  
g721_encode_sb_sptr.set_min_noutput_items(g721_encode_sb_sptr self, int m)  
  
g721_encode_sb_sptr.set_thread_priority(g721_encode_sb_sptr self, int priority) → int  
  
g721_encode_sb_sptr.thread_priority(g721_encode_sb_sptr self) → int
```

gnuradio.vocoder.**g723\_24\_decode\_bs**() → g723\_24\_decode\_bs\_sptr

This block performs g723\_24 audio decoding.

Constructor Specific Documentation:

Make G722\_24 decoder block.

```
g723_24_decode_bs_sptr.active_thread_priority(g723_24_decode_bs_sptr self) → int  
  
g723_24_decode_bs_sptr.declare_sample_delay(g723_24_decode_bs_sptr self, int which, int delay)  
    declare_sample_delay(g723_24_decode_bs_sptr self, unsigned int delay)  
  
g723_24_decode_bs_sptr.message_subscribers(g723_24_decode_bs_sptr self, swig_int_ptr which_port) → swig_int_ptr  
  
g723_24_decode_bs_sptr.min_noutput_items(g723_24_decode_bs_sptr self) → int  
  
g723_24_decode_bs_sptr.pc_input_buffers_full_avg(g723_24_decode_bs_sptr self) → pmt_vector_float
```

*self, int which*) → float

pc\_input\_buffers\_full\_avg(g723\_24\_decode\_bs\_sptr self) -> pmt\_vector\_float

g723\_24\_decode\_bs\_sptr.**pc\_noutput\_items\_avg**(g723\_24\_decode\_bs\_sptr self)  
→ float

g723\_24\_decode\_bs\_sptr.**pc\_nproduced\_avg**(g723\_24\_decode\_bs\_sptr self) →  
float

g723\_24\_decode\_bs\_sptr.**pc\_output\_buffers\_full\_avg**(g723\_24\_decode\_bs\_sptr  
*self, int which*) → float

pc\_output\_buffers\_full\_avg(g723\_24\_decode\_bs\_sptr self) -> pmt\_vector\_float

g723\_24\_decode\_bs\_sptr.**pc\_throughput\_avg**(g723\_24\_decode\_bs\_sptr self) →  
float

g723\_24\_decode\_bs\_sptr.**pc\_work\_time\_avg**(g723\_24\_decode\_bs\_sptr self) →  
float

g723\_24\_decode\_bs\_sptr.**pc\_work\_time\_total**(g723\_24\_decode\_bs\_sptr self) →  
float

g723\_24\_decode\_bs\_sptr.**sample\_delay**(g723\_24\_decode\_bs\_sptr self, *int which*)  
→ unsigned int

g723\_24\_decode\_bs\_sptr.**set\_min\_noutput\_items**(g723\_24\_decode\_bs\_sptr  
*self, int m*)

g723\_24\_decode\_bs\_sptr.**set\_thread\_priority**(g723\_24\_decode\_bs\_sptr *self*,  
*int priority*) → int

g723\_24\_decode\_bs\_sptr.**thread\_priority**(g723\_24\_decode\_bs\_sptr self) → int

gnuradio.vocoder.**g723\_24\_encode\_sb**() → g723\_24\_encode\_sb\_sptr

This block performs g723\_24 audio encoding.

Constructor Specific Documentation:

Make G722\_24 encoder block.

g723\_24\_encode\_sb\_sptr.**active\_thread\_priority**(g723\_24\_encode\_sb\_sptr  
*self*) → int

g723\_24\_encode\_sb\_sptr.**declare\_sample\_delay**(g723\_24\_encode\_sb\_sptr *self*,  
*int which, int delay*)

declare\_sample\_delay(g723\_24\_encode\_sb\_sptr self, unsigned int delay)

g723\_24\_encode\_sb\_sptr.**message\_subscribers**(g723\_24\_encode\_sb\_sptr *self*,  
*swig\_int\_ptr which\_port*) → swig\_int\_ptr

g723\_24\_encode\_sb\_sptr.**min\_noutput\_items**(g723\_24\_encode\_sb\_sptr self) →  
int

g723\_24\_encode\_sb\_sptr.**pc\_input\_buffers\_full\_avg**(g723\_24\_encode\_sb\_sptr  
*self, int which*) → float

pc\_input\_buffers\_full\_avg(g723\_24\_encode\_sb\_sptr self) -> pmt\_vector\_float

g723\_24\_encode\_sb\_sptr.**pc\_noutput\_items\_avg**(g723\_24\_encode\_sb\_sptr self)  
→ float

g723\_24\_encode\_sb\_sptr.**pc\_nproduced\_avg**(g723\_24\_encode\_sb\_sptr self) →  
float

g723\_24\_encode\_sb\_sptr.**pc\_output\_buffers\_full\_avg**(g723\_24\_encode\_sb\_sptr

*self, int which*) → float

*pc\_output\_buffers\_full\_avg*(g723\_24\_encode\_sb\_sptr self) → pmt\_vector\_float

*g723\_24\_encode\_sb\_sptr.pc\_throughput\_avg*(g723\_24\_encode\_sb\_sptr self) → float

*g723\_24\_encode\_sb\_sptr.pc\_work\_time\_avg*(g723\_24\_encode\_sb\_sptr self) → float

*g723\_24\_encode\_sb\_sptr.pc\_work\_time\_total*(g723\_24\_encode\_sb\_sptr self) → float

*g723\_24\_encode\_sb\_sptr.sample\_delay*(g723\_24\_encode\_sb\_sptr self, int which) → unsigned int

*g723\_24\_encode\_sb\_sptr.set\_min\_noutput\_items*(g723\_24\_encode\_sb\_sptr self, int m)

*g723\_24\_encode\_sb\_sptr.set\_thread\_priority*(g723\_24\_encode\_sb\_sptr self, int priority) → int

*g723\_24\_encode\_sb\_sptr.thread\_priority*(g723\_24\_encode\_sb\_sptr self) → int

gnuradio.vocoder.**g723\_40\_decode\_bs**() → g723\_40\_decode\_bs\_sptr

This block performs g723\_40 audio decoding.

Constructor Specific Documentation:

Make G722\_40 decoder block.

*g723\_40\_decode\_bs\_sptr.active\_thread\_priority*(g723\_40\_decode\_bs\_sptr self) → int

*g723\_40\_decode\_bs\_sptr.declare\_sample\_delay*(g723\_40\_decode\_bs\_sptr self, int which, int delay)

*declare\_sample\_delay*(g723\_40\_decode\_bs\_sptr self, unsigned int delay)

*g723\_40\_decode\_bs\_sptr.message\_subscribers*(g723\_40\_decode\_bs\_sptr self, swig\_int\_ptr which\_port) → swig\_int\_ptr

*g723\_40\_decode\_bs\_sptr.min\_noutput\_items*(g723\_40\_decode\_bs\_sptr self) → int

*g723\_40\_decode\_bs\_sptr.pc\_input\_buffers\_full\_avg*(g723\_40\_decode\_bs\_sptr self, int which) → float

*pc\_input\_buffers\_full\_avg*(g723\_40\_decode\_bs\_sptr self) → pmt\_vector\_float

*g723\_40\_decode\_bs\_sptr.pc\_noutput\_items\_avg*(g723\_40\_decode\_bs\_sptr self) → float

*g723\_40\_decode\_bs\_sptr.pc\_nproduced\_avg*(g723\_40\_decode\_bs\_sptr self) → float

*g723\_40\_decode\_bs\_sptr.pc\_output\_buffers\_full\_avg*(g723\_40\_decode\_bs\_sptr self, int which) → float

*pc\_output\_buffers\_full\_avg*(g723\_40\_decode\_bs\_sptr self) → pmt\_vector\_float

*g723\_40\_decode\_bs\_sptr.pc\_throughput\_avg*(g723\_40\_decode\_bs\_sptr self) → float

*g723\_40\_decode\_bs\_sptr.pc\_work\_time\_avg*(g723\_40\_decode\_bs\_sptr self) → float

*g723\_40\_decode\_bs\_sptr.pc\_work\_time\_total*(g723\_40\_decode\_bs\_sptr self) →

float

`g723_40_decode_bs_sptr.sample_delay(g723_40_decode_bs_sptr self, int which)`  
→ unsigned int

`g723_40_decode_bs_sptr.set_min_noutput_items(g723_40_decode_bs_sptr self, int m)`

`g723_40_decode_bs_sptr.set_thread_priority(g723_40_decode_bs_sptr self, int priority)` → int

`g723_40_decode_bs_sptr.thread_priority(g723_40_decode_bs_sptr self)` → int

`gnuradio.vocoder.g723_40_encode_sb()` → `g723_40_encode_sb_sptr`

This block performs g723\_40 audio encoding.

Constructor Specific Documentation:

Make G722\_40 encoder block.

`g723_40_encode_sb_sptr.active_thread_priority(g723_40_encode_sb_sptr self)` → int

`g723_40_encode_sb_sptr.declare_sample_delay(g723_40_encode_sb_sptr self, int which, int delay)`

`declare_sample_delay(g723_40_encode_sb_sptr self, unsigned int delay)`

`g723_40_encode_sb_sptr.message_subscribers(g723_40_encode_sb_sptr self, swig_int_ptr which_port)` → `swig_int_ptr`

`g723_40_encode_sb_sptr.min_noutput_items(g723_40_encode_sb_sptr self)` → int

`g723_40_encode_sb_sptr.pc_input_buffers_full_avg(g723_40_encode_sb_sptr self, int which)` → float

`pc_input_buffers_full_avg(g723_40_encode_sb_sptr self)` → `pmt_vector_float`

`g723_40_encode_sb_sptr.pc_noutput_items_avg(g723_40_encode_sb_sptr self)` → float

`g723_40_encode_sb_sptr.pc_nproduced_avg(g723_40_encode_sb_sptr self)` → float

`g723_40_encode_sb_sptr.pc_output_buffers_full_avg(g723_40_encode_sb_sptr self, int which)` → float

`pc_output_buffers_full_avg(g723_40_encode_sb_sptr self)` → `pmt_vector_float`

`g723_40_encode_sb_sptr.pc_throughput_avg(g723_40_encode_sb_sptr self)` → float

`g723_40_encode_sb_sptr.pc_work_time_avg(g723_40_encode_sb_sptr self)` → float

`g723_40_encode_sb_sptr.pc_work_time_total(g723_40_encode_sb_sptr self)` → float

`g723_40_encode_sb_sptr.sample_delay(g723_40_encode_sb_sptr self, int which)`  
→ unsigned int

`g723_40_encode_sb_sptr.set_min_noutput_items(g723_40_encode_sb_sptr self, int m)`

`g723_40_encode_sb_sptr.set_thread_priority(g723_40_encode_sb_sptr self, int priority)` → int

`g723_40_encode_sb_sptr.thread_priority(g723_40_encode_sb_sptr self) → int`

`gnuradio.vocoder.gsm_fr_decode_ps()` → `gsm_fr_decode_ps_sptr`

GSM 06.10 Full Rate Vocoder Decoder

Input: Vector of 33 bytes per 160 input samples Output: 16-bit shorts representing speech samples.

Constructor Specific Documentation:

Make GSM decoder block.

`gsm_fr_decode_ps_sptr.active_thread_priority(gsm_fr_decode_ps_sptr self) → int`

`gsm_fr_decode_ps_sptr.declare_sample_delay(gsm_fr_decode_ps_sptr self, int which, int delay)`

`declare_sample_delay(gsm_fr_decode_ps_sptr self, unsigned int delay)`

`gsm_fr_decode_ps_sptr.message_subscribers(gsm_fr_decode_ps_sptr self, swig_int_ptr which_port) → swig_int_ptr`

`gsm_fr_decode_ps_sptr.min_noutput_items(gsm_fr_decode_ps_sptr self) → int`

`gsm_fr_decode_ps_sptr.pc_input_buffers_full_avg(gsm_fr_decode_ps_sptr self, int which) → float`

`pc_input_buffers_full_avg(gsm_fr_decode_ps_sptr self) → pmt_vector_float`

`gsm_fr_decode_ps_sptr.pc_noutput_items_avg(gsm_fr_decode_ps_sptr self) → float`

`gsm_fr_decode_ps_sptr.pc_nproduced_avg(gsm_fr_decode_ps_sptr self) → float`

`gsm_fr_decode_ps_sptr.pc_output_buffers_full_avg(gsm_fr_decode_ps_sptr self, int which) → float`

`pc_output_buffers_full_avg(gsm_fr_decode_ps_sptr self) → pmt_vector_float`

`gsm_fr_decode_ps_sptr.pc_throughput_avg(gsm_fr_decode_ps_sptr self) → float`

`gsm_fr_decode_ps_sptr.pc_work_time_avg(gsm_fr_decode_ps_sptr self) → float`

`gsm_fr_decode_ps_sptr.pc_work_time_total(gsm_fr_decode_ps_sptr self) → float`

`gsm_fr_decode_ps_sptr.sample_delay(gsm_fr_decode_ps_sptr self, int which) → unsigned int`

`gsm_fr_decode_ps_sptr.set_min_noutput_items(gsm_fr_decode_ps_sptr self, int m)`

`gsm_fr_decode_ps_sptr.set_thread_priority(gsm_fr_decode_ps_sptr self, int priority) → int`

`gsm_fr_decode_ps_sptr.thread_priority(gsm_fr_decode_ps_sptr self) → int`

`gnuradio.vocoder.gsm_fr_encode_sp()` → `gsm_fr_encode_sp_sptr`

GSM 06.10 Full Rate Vocoder Encoder

Input: 16-bit shorts representing speech samples Output: Vector of 33 bytes per 160 input samples.

Constructor Specific Documentation:

Make GSM encoder block.

```
gsm_fr_encode_sp_sptr.active_thread_priority(gsm_fr_encode_sp_sptr self)
→ int

gsm_fr_encode_sp_sptr.declare_sample_delay(gsm_fr_encode_sp_sptr self, int
which, int delay)
    declare_sample_delay(gsm_fr_encode_sp_sptr self, unsigned int delay)

gsm_fr_encode_sp_sptr.message_subscribers(gsm_fr_encode_sp_sptr self,
swig_int_ptr which_port) → swig_int_ptr

gsm_fr_encode_sp_sptr.min_noutput_items(gsm_fr_encode_sp_sptr self) → int

gsm_fr_encode_sp_sptr.pc_input_buffers_full_avg(gsm_fr_encode_sp_sptr
self, int which) → float
    pc_input_buffers_full_avg(gsm_fr_encode_sp_sptr self) -> pmt_vector_float

gsm_fr_encode_sp_sptr.pc_noutput_items_avg(gsm_fr_encode_sp_sptr self) →
float

gsm_fr_encode_sp_sptr.pc_nproduced_avg(gsm_fr_encode_sp_sptr self) →
float

gsm_fr_encode_sp_sptr.pc_output_buffers_full_avg(gsm_fr_encode_sp_sptr
self, int which) → float
    pc_output_buffers_full_avg(gsm_fr_encode_sp_sptr self) -> pmt_vector_float

gsm_fr_encode_sp_sptr.pc_throughput_avg(gsm_fr_encode_sp_sptr self) →
float

gsm_fr_encode_sp_sptr.pc_work_time_avg(gsm_fr_encode_sp_sptr self) →
float

gsm_fr_encode_sp_sptr.pc_work_time_total(gsm_fr_encode_sp_sptr self) →
float

gsm_fr_encode_sp_sptr.sample_delay(gsm_fr_encode_sp_sptr self, int which) →
unsigned int

gsm_fr_encode_sp_sptr.set_min_noutput_items(gsm_fr_encode_sp_sptr self,
int m)

gsm_fr_encode_sp_sptr.set_thread_priority(gsm_fr_encode_sp_sptr self, int
priority) → int

gsm_fr_encode_sp_sptr.thread_priority(gsm_fr_encode_sp_sptr self) → int
```

gnuradio.vocoder.**ulaw\_decode\_bs**() → ulaw\_decode\_bs\_sptr

This block performs ulaw audio decoding.

Constructor Specific Documentation:

Make ulaw decoder block.

```
ulaw_decode_bs_sptr.active_thread_priority(ulaw_decode_bs_sptr self) →
int

ulaw_decode_bs_sptr.declare_sample_delay(ulaw_decode_bs_sptr self, int
which, int delay)
    declare_sample_delay(ulaw_decode_bs_sptr self, unsigned int delay)

ulaw_decode_bs_sptr.message_subscribers(ulaw_decode_bs_sptr self,
swig_int_ptr which_port) → swig_int_ptr
```

```

ulaw_decode_bs_sptr.min_noutput_items(ulaw_decode_bs_sptr self) → int

ulaw_decode_bs_sptr.pc_input_buffers_full_avg(ulaw_decode_bs_sptr self,
int which) → float
    pc_input_buffers_full_avg(ulaw_decode_bs_sptr self) -> pmt_vector_float

ulaw_decode_bs_sptr.pc_noutput_items_avg(ulaw_decode_bs_sptr self) →
float

ulaw_decode_bs_sptr.pc_nproduced_avg(ulaw_decode_bs_sptr self) → float

ulaw_decode_bs_sptr.pc_output_buffers_full_avg(ulaw_decode_bs_sptr self,
int which) → float
    pc_output_buffers_full_avg(ulaw_decode_bs_sptr self) -> pmt_vector_float

ulaw_decode_bs_sptr.pc_throughput_avg(ulaw_decode_bs_sptr self) → float

ulaw_decode_bs_sptr.pc_work_time_avg(ulaw_decode_bs_sptr self) → float

ulaw_decode_bs_sptr.pc_work_time_total(ulaw_decode_bs_sptr self) → float

ulaw_decode_bs_sptr.sample_delay(ulaw_decode_bs_sptr self, int which) →
unsigned int

ulaw_decode_bs_sptr.set_min_noutput_items(ulaw_decode_bs_sptr self, int
m)

ulaw_decode_bs_sptr.set_thread_priority(ulaw_decode_bs_sptr self, int
priority) → int

ulaw_decode_bs_sptr.thread_priority(ulaw_decode_bs_sptr self) → int

```

gnuradio.vocoder.**ulaw\_encode\_sb**() → ulaw\_encode\_sb\_sptr

This block performs g.711 ulaw audio encoding.

Constructor Specific Documentation:

Make ulaw encoder block.

```

ulaw_encode_sb_sptr.active_thread_priority(ulaw_encode_sb_sptr self) →
int

ulaw_encode_sb_sptr.declare_sample_delay(ulaw_encode_sb_sptr self, int
which, int delay)
    declare_sample_delay(ulaw_encode_sb_sptr self, unsigned int delay)

ulaw_encode_sb_sptr.message_subscribers(ulaw_encode_sb_sptr self,
swig_int_ptr which_port) → swig_int_ptr

ulaw_encode_sb_sptr.min_noutput_items(ulaw_encode_sb_sptr self) → int

ulaw_encode_sb_sptr.pc_input_buffers_full_avg(ulaw_encode_sb_sptr self,
int which) → float
    pc_input_buffers_full_avg(ulaw_encode_sb_sptr self) -> pmt_vector_float

ulaw_encode_sb_sptr.pc_noutput_items_avg(ulaw_encode_sb_sptr self) →
float

ulaw_encode_sb_sptr.pc_nproduced_avg(ulaw_encode_sb_sptr self) → float

ulaw_encode_sb_sptr.pc_output_buffers_full_avg(ulaw_encode_sb_sptr self,
int which) → float
    pc_output_buffers_full_avg(ulaw_encode_sb_sptr self) -> pmt_vector_float

```

```
ulaw_encode_sb_sptr.pc_throughput_avg(ulaw_encode_sb_sptr self) → float  
  
ulaw_encode_sb_sptr.pc_work_time_avg(ulaw_encode_sb_sptr self) → float  
  
ulaw_encode_sb_sptr.pc_work_time_total(ulaw_encode_sb_sptr self) → float  
  
ulaw_encode_sb_sptr.sample_delay(ulaw_encode_sb_sptr self, int which) →  
unsigned int  
  
ulaw_encode_sb_sptr.set_min_noutput_items(ulaw_encode_sb_sptr self, int  
m)  
  
ulaw_encode_sb_sptr.set_thread_priority(ulaw_encode_sb_sptr self, int  
priority) → int  
  
ulaw_encode_sb_sptr.thread_priority(ulaw_encode_sb_sptr self) → int
```