Driver for the WM8960 codec

This driver supports controlling a WM8960 codec. It is a literal Python translation of the C-Code provided by NXP/Freescale for their i.MX RT series of MCUs. Almost nothing was added, and just a few API related names were changed or added to cope with the naming style of MicroPython.

Features

The primary purpose of the driver is initialisation and setting operation modes of the codec. I does not do the audio data processing for the codec. That is the task of a separate driver built into the Operating system or firmware of a device.

Connection

The WM8960 supports next to the audio interface the I2C interface. The connection depends on the interface used and the number of devices in the system. For the I2C interface, SCL and SDA have to be connected, and of course GND and Vcc. The I2C default address is 0x1a.

Class

The driver contains the WM8960 class and quite a few name definitions.

```
wm8960=wm8960.WM8960(i2c, *,
    sample_rate=16000,
    bits=16,
    swap=swap_none,
    route=route_playback_record,
    left_input=mic_input3,
    right_input=input_mic2,
    sysclk_source=sysclk_mclk,
    mclk_freq=None,
    primary=False,
    adc_sync=sync_dac,
    protocol=bus_I2S,
    i2c_address=WM8960_I2C_ADDR
)
```

Only the first argument, i2c, is mandatory. All others are optional.
Arguments:

- *i2c* The I2C bus object. It has be be created beforehand.
- *sample_rate* The audio sample rate. Acceptable values are 8000, 11025,

12000, 16000, 22050, 24000, 32000, 44100, 48000, 96000, 192000 and 384000. Note, that not every I2S hardware will support all values.
- *bits* The number of bits per audio word. Acceptable value are 16, 20, 24, and 32.
- *swap* Swap the left & right channel, if set. For a list of options, see the table below.
- *route* Setting the audio path in the codec. For a list of options, see the table below.
- *left_input* Set the audio source for the left input channel. For a list of options, see the table below.
- *right_input* Set the audio source for the right input channel.For a list of options, see the table below.
- *sysclk_source* Control, whether the internal master clock called "sysclk" is directly taken from the MCLK input or derived from it using an internal PLL. It is usually not required to change that.
- *mclk_freq* Argument for telling the mclk frequency applied to the MCLK pin of the codec If not set, default values are used.
- *primary* Let the WM8960 act as Primary or Secondary device. The default setting is False. When set to False, sample_rate and bits are controlled by the MCU.
- *adc_sync* Tell which input is used for the ADC sync signal. The default is using the DACLRC pin.
- *protocol* Setting the communication protocol. The default is I2S. For a list of options, see the table below.
- *i2c_address* The I2C address of the WM8960. The default is 0x1a or 26.

If mclk_freq is not set, the following default values are assumed:

- sysclk_source == sysclk_PLL: 11.2896 MHz for sample rates of 44100, 22050 and 11015 Hz, and 12.288 Mhz for sample rates < 48000, otherwise sample_rate * 256.
- sysclk_source == sysclk_mclk: sample_rate * 256.

If the MCLK signal is applied using e.g. a separate oszillator, it must be specified for proper operation.

Tables of parameter constants

Swap Parameter

Value	Name
0	swap_none
1	swap_input
2	swap_output

Route parameter:

Value	Name
0	route_bypass
1	route_playback
2	route_playback_record
5	route_record

Protocol Parameter

Value	Name
2	bus_I2S
1	bus_left_justified
0	bus_right_justified
3	bus_PCMA
19	bus_PCMB

Input Source Parameter

Value	Name
0	input_closed
1	input_mic1
2	input_mic2
3	input_mic3
4	input_line2_
5	input_line3

Route Parameter

Value	Name
0	route_bypass
1	route_playback
2	route_playback_record
5	route_record

Master Clock Source Parameter

Value	Name
0	sysclk_mclk
1	sysclk_PLL

Module Names

Value	Name
0	module_ADC

Value	Name
1	module_DAC
2	module_VREF
3	module_headphone
4	module_mic_bias
5	module_mic
6	module_line_in
7	module_line_out
8	module_speaker
9	module_omix
10	module_mono_out

Play Channel Names

Value	Name
1	play_headphone_left
2	play_headphone_right
4	play_speaker_left
8	play_speaker_right

adc_sync Parameters

Value	Name
0	sync_adc
1	sync_dac

Methods

Next to the initialisation, the driver provides some useful methods for controlling the operation:

set_left_input(input source)

Specify the source for the left input. The input source names are listed above.

set_right_input(input source)

Specify the source for the left input. For a list of suitable parameter values, see the table above.

set_volume(module, value [, value_r])

Sets the volume of a certain module. If two values are given, the first one is used for the left channel, the second for the right channel. For a list of suitable modules and highest values, see the table below.

Module Names and value ranges

Value Range	Name
0-255	module_ADC
0-255	module_DAC
0-127	module_headphone
0-63	module_line_in
0-127	module_speaker

value = get_volume(module)

Get the actual volumes set for a module as a two element tuple. The module names are the same as for set_volume().

value = volume(module [, value [, value_r]])

Sets or get the volume of a certain module. If not value is supplied, the actual volume is returned. If two values are given, the first one is used for the left channel, the second for the right channel. For a list of suitable modules and highest values, see the table below.

mute(module, enable, soft=True, ramp=wm8960.mute_fast)

Mute or unmute the output. If **enable** is True, the output is muted, if False it is unmuted. If **soft** is True, mute will happen as a soft transition. The time for the transistion is defined by **ramp**, which is either mute_fast or mute_slow.

set_data_route(route)

Set the audio data route. For the parameter value/names, look at the table above.

set_module(module, True|False)

Enable or disable a module. For the list of module names, look at the table above. Note that enabling module_mono_out is different from the mono() method. The first enables output 3, while the mono method sends a mono mix to the left and right output.

enable_module(module, True|False)

Enable a module. For the list of module names, look at the table above.

disable_module(module, True|False)

Disable a module. For the list of module names, look at the table above.

expand_3d(level)

Enable Stereo 3d exansion. Level is a number between 0 and 15. A value of 0 disables the expansion.

mono(True | False)

If set to True, a Mono mix is sent to the left and right output channel. This is different from enabling module_mono_mix, which enables output 3.

alc_mode(channel, mode=alc_mode)

Enables or disables ALC mode. Parameters are:

channel Enable and set the channel for ALC. The parameter values are:

- alc_off: Switch ALC off
- als_right: Use the right input channel
- alc_left: Use the left input channel
- alc_stereo: Use both input channels.

mode Set the ALC mode. Input values are

- alc_mode: act as ALC
- alc_limiter: act as limiter.

alc_gain(target=-12, max_gain=30, min_gain=-17.25, noise_gate=-78)

Set the target level, highest and lowest gain levels and the noise gate as dB level. Permitted ranges are:

- target: -22.5 to -1.5 dB
- max_gain: -12 to 30 dB
- min_gain: -17 to 25 dB
- noise_gate: -78 to -30 dB

Excess values are limited to the permitted ranges. A value of -78 or less for **noise_gate** disables the noise gate function.

alc_time(attack=24, decay=192, hold=0)

Set the dynamic characteristic of ALC. The times are given as ms values. Permitted ranges are:

- attack: 6 to 6140
- decay: 24 to 24580
- hold: 0 to 43000

Excess values are limited within the permitted ranges.

deinit()

Disable all modules.

Example Code

```
# Micro_python WM8960 Codec driver
#
# Setting the codec to secondary mode using the default settings
#
from machine import Pin, I2C
import wm8960

i2c = I2C(0)
wm=wm8960.WM8960(i2c)
```

```
# Micro_python WM8960 Codec driver
#
# Setting the codec to primary mode using specific audio format se
#
from machine import Pin, I2C
import wm8960

i2c = I2C(0)
wm=wm8960.WM8960(i2c, primary=True, sample_rate=16000, bits=32)
```

Record with a Sparkfun WM8960 breakout board with Teensy in secondary mode(default):

```
# Micro_python WM8960 Codec driver
#
# The breakout board uses a fixed 24MHz MCLK. Therefore the intern
# PLL must be used as sysclk, which is the master audio clock.
# The Sparkfun board has the WS pins for RX and TX connected on th
# board. Therefore adc_sync must be set to sync_adc, to configure
# it's ADCLRC pin as input.
#
from machine import Pin, I2C
```

```
import wm8960
i2c = I2C(0)
wm=wm8960.WM8960(i2c, sample_rate=16_000,
    adc_sync=wm8960.sync_adc,
    sysclk_source=wm8960.sysclk_PLL,
    mclk_freq=24_000_000,
    left_input=wm8960.input_mic1,
    right_input=wm8960.input_closed)
```

Play with a Sparkfun WM8960 breakout board with Teensy in secondary mode(default)::

```
# The breakout board uses a fixed 24MHz MCLK. Therefore the interna
# PLL must be used as sysclk, which is the master audio clock.
# The Sparkfun board has the WS pins for RX and TX connected on th
# board. Therefore adc_sync must be set to sync_adc, to configure
# it's ADCLRC pin as input.

from machine import I2C
i2c=I2C(0)
import wm8960
wm=wm8960.WM8960(i2c, sample_rate=44_100,
    adc_sync=wm8960.sync_adc,
    sysclk_source=wm8960.sysclk_PLL,
    mclk_freq=24_000_000)
wm.set_volume(wm8960.module_headphone, 100)
```