Programming Guide

FD6818

REV.1.0.0

content

Table of contents

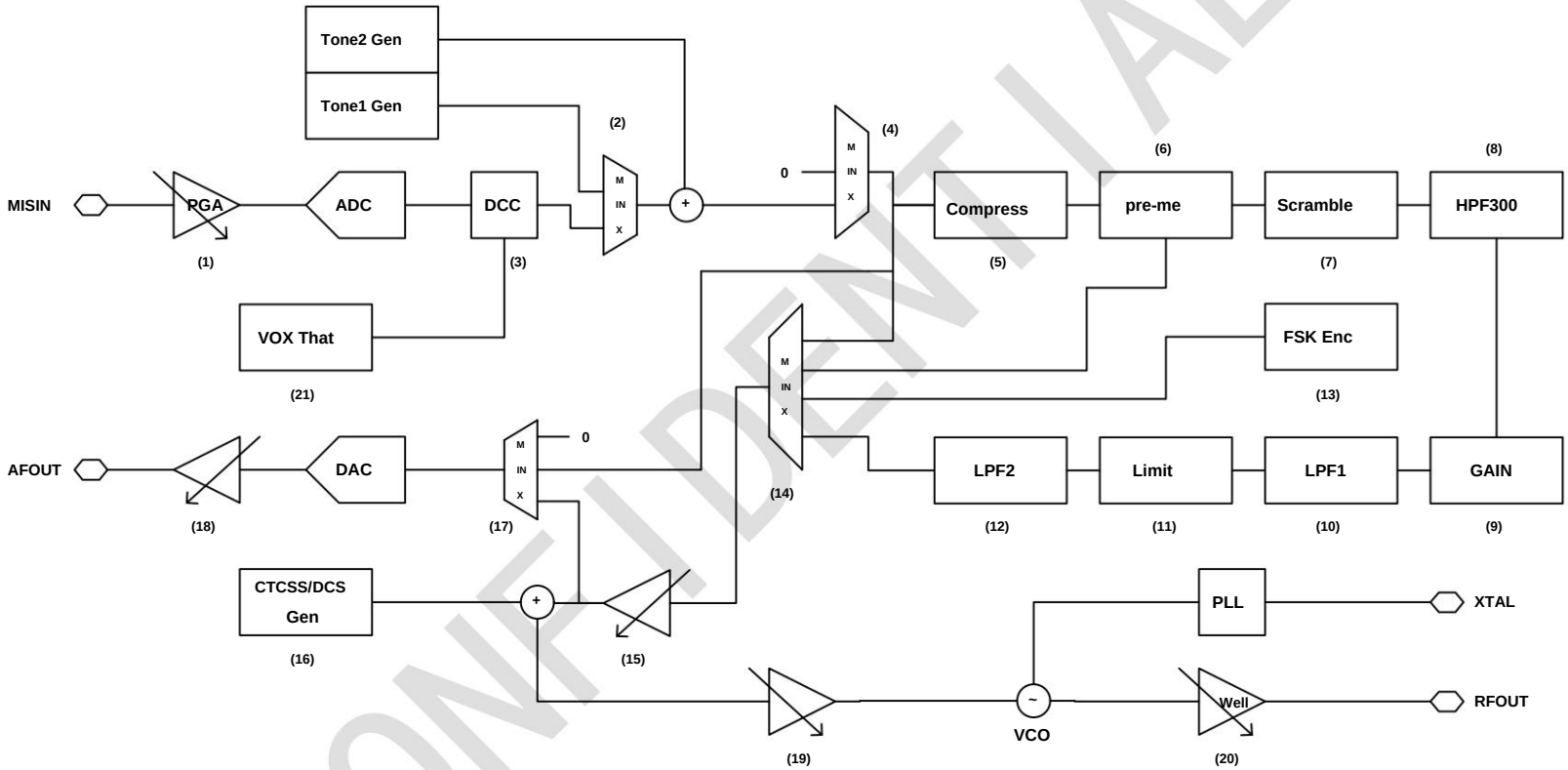
revision history

revision history

revision date	modify the content	revision location
2018.07.24	Wide and narrow bandwidth settings are wrong, delete the settings for REG_3AH	
2018.07.24	In VCO mode, set REG_03H incorrectly	VCO Mode Settings
2018.09.06	Modify the bandwidth setting at 25k/12.5k The	Band, frequency point, bandwidth mode settings
2018.09.07	first revision of the chip, modify the frequency band definition range	Band, frequency point, bandwidth mode settings

Launch block diagram

Launch block diagram



Launch block diagram

Block diagram description:

(1) MIC channel PGA analog gain

Register	Address(HEX)	Description
mic_pga_gain[4:0]	REG_18H[15:11]	Value Gain MIC_Sens(mV) 00000 = not valid 00001 = 4.6250 ~26.2 00010 = 6.1250 ~19.8 00011 = 10.6250 ~11.4 00100 = 9.1250 ~13.3 00101 = 13.6875 ~ 8.9 00110 = 15.1875 ~ 8.0 00111 = 19.6875 ~ 6.2 01000 = 12.1250 ~10.0(default) 01001 = 16.6875 ~ 7.3 01010 = 18.1875 ~ 6.7 01011 = 22.6250 ~ 5.4 01100 = 21.2500 ~ 5.7 01101 = 25.6250 ~ 4.7 01110 = 27.1250 ~ 4.5 01111 = 31.7500 ~ 3.8 10000 = 18.1875 ~ 6.7 10001 = 22.6875 ~ 5.3 10010 = 24.2500 ~ 5.0 10011 = 28.7500 ~ 4.2 10100 = 27.2500 ~ 4.4 10101 = 31.7500 ~ 3.8 10110 = 33.1250 ~ 3.7 10111 = 37.6250 ~ 3.2 11000 = 30.1875 ~ 4.0 11001 = 34.6250 ~ 3.5 11010 = 36.1250 ~ 3.4 11011 = 40.7500 ~ 3.0 11100 = 39.0000 ~ 3.1

		11101 = 43.7500 ~ 2.8 11110 = 45.1250 ~ 2.7 11111 = 49.5625 ~ 2.4

The value of the gain table of mic_pga_gain above represents the relative gain. Modifying this register will affect the VOX detection range and needs to be reset

Set the VOX threshold. When adjusting the MIC sensitivity, try not to modify this register, just modify the mic_sens_gain register, see the text for details

Refer to the "Setting Modulation Limits and MIC Sensitivity" section of the file.

(2) TONE1, TONE2 generation

Register	Address(HEX)	Description
tone1_gen	REG_70H[15]	Enable TONE1
tone1_gain[6:0]	REG_70H[14:8]	TONE1 tuning gain
tone2_gen	REG_70H[7]	Enable TONE2
tone2_gain[6:0]	REG_70H[6:0]	TONE2 tuning gain
tone1_freq[15:0]	REG_71H[15:0]	TONE1 frequency· control word =freq(kHz)* 2^26/6500
tone2_freq[15:0]	REG_72H[15:0]	TONE2 frequency· control word =freq(kHz)* 2^26/6500

If tone1_gen=1, the MIC input will be bypassed and a monophonic tone1 will be generated.

If tone1_gen=1, and tone2_gen=1, the MIC input is bypassed, and two-tone tone1+tone2 is generated at the same time.

(3) MIC DC filter

Register	Address(HEX)	Description
dcc_tx_bypass	REG_7EH[7]	1=MICIN DC Filter bypass
dcc_tx_bw[2:0]	REG_7EH[5:3]	MICIN DC Filter bandwidth(3dB) select 111=15Hz;110=30Hz;101=60Hz;100=120Hz; 011=240Hz;010=480Hz;

Default dcc_tx_bypass=0, dc_tx_bw=111. When used as DMR/dPMR mode, this DC can be turned off by setting dcc_tx_bypass=1

filter. When modifying the transmit frequency, you can use dcc_tx_bw=011 or 010 to attenuate 300Hz.

(4) Transmission path selection 1

Register	Address(HEX)	Description
audio_tx_mute	REG_50H[15]	1=Audio Tx Mute

audio_tx_mute can be used for idle between DTMF transmit symbols.

Launch block diagram

(5) Transmitting voice companding

See the "Voice Mode Settings" section of the documentation for details.

(6) Transmit speech pre-emphasis

Register	Address(HEX)	Description
audio_emph_tx_bypass	REG_9BH[0]	1=PRE-EMPHASIS bypass

(7) Transmit voice scrambling

See the "SCRAMBLE Mode Settings" section of the document for details.

(8) Transmit voice 300Hz high pass filter

Register	Address(HEX)	Description
audio_hpf_tx_bypass	REG_9BH[3]	1=Audio HPF 300Hz bypass for TX

~~(9) Transmit voice gain 1 (MIC sensitivity adjustment)~~

~~Register~~	~~Address(HEX)~~	~~Description~~
~~mic_sens_gain[5:0]~~	~~REG_97H[13:8]~~	~~MIC sensitivity adjust gain~~ ~~0=mute, 63=max, 0.5dB/step~~

~~See the "Setting Modulation Limits and MIC Sensitivity" section of the document for detailed instructions.~~

(10) Transmit speech low-pass filter 1

Register	Address(HEX)	Description
audio_lpf1_tx_bypass	REG_9BH[2]	1=Audio LPF1 bypass for TX

~~(11) Transmit voice limiter~~

~~Register~~	~~Address(HEX)~~	~~Description~~
~~audio_tx_limit_bypass~~	~~REG_50H[10]~~	~~1=Audio Tx Limit bypass~~
~~audio_tx_limit[9:0]~~	~~REG_50H[9:0]~~	~~Audio Tx Limit Value~~

It is recommended not to modify this register, the modulation limit only needs to adjust the dev_sh and dev_lvl registers (see the document "Setting the modulation limit and MIC for details"

Sensitivity" section).

(12) Transmit speech low-pass filter 2

Register	Address(HEX)	Description
audio_lpf2_tx_bypass	REG_9BH[1]	1=Audio LPF2 bypass for TX
audio_lpf2_tx_sel[1:0]	REG_43H[8:6]	Audio LPF bandwidth (Apass=1dB) for Tx 100 = 4.5 kHz 101 = 4.25 kHz 110 = 4 kHz 111 = 3.75 kHz 000 = 3 kHz 001 = 2.5 kHz 010 = 2 kHz 011 = 1.7kHz

(13) Transmit FSK code

See the "FSK Mode Settings" section for details.

(14) Transmission path selection 2

Register	Address(HEX)	Description
audio_tx_path_sel[1:0]	REG_9DH[3:2]	01=select pre-emphasis output 10=reserved 11=select LPF2 output

Transmission path selection, the output of several nodes can be selected for transmission.

~~(15) Transmit speech gain 2~~

~~Register~~	~~Address(HEX)~~	~~Description~~
~~audio_tx_gain_sh [2:0]~~	~~REG_53H[12:10]~~	~~111=max,000=min~~ ~~0 ~-42dB, 6dB/step, digital gain~~
~~audio_tx_gain[4:0]~~	~~REG_53H[4:0]~~	~~0=min,31=max~~

		-6~3.3dB, ~=1dB/step, digital gain

It is recommended not to modify this gain, because the voice and sub-audio will be added and transmitted later. If this gain is too high and the sub-audio modulation is large, it will cause waveform overflow. It is recommended to use the default value or initialize the recommended value.

(16) Transmit sub-audio generation

See the sub-audio settings section of the documentation for details.

(17) AF DAC channel selection

Register	Address(HEX)	Description
afout_invert	REG_47H[13]	1 = invert afout
afout_mode[4:0]	REG_47H[12:8]	0x10 = MUTE 0x11 = RX AFOUT 0x18 = BEEP/TX Side ToneÿCTCSS/CDCSS is not includeÿ 0x15 = RX ALARM TONE

This register is multiplexed for transceiver and can be used to set mute, receive AF output and sidetone BEEP output.

(18) AF DAC analog gain

Register	Address(HEX)	Description
afout_en	REG_03H[9]	1=Enable AFOUT DAC
dac_vgain[3:0]	REG_23H[3:0]	DAC maximum output level

Whether it is output receiving AF or BEEP sidetone, you need to set afout_enable=1 to enable AF DAC. gain dac_vgain setting,

The recommended maximum value is 1111 (default). Digital gain can be used for volume adjustment. For details, please refer to the document "Receive Mute (MUTE) and Volume Settings" part.

(19) Transmit modulation limitation

Register	Address(HEX)	Description
dev_in	REG_40H[12]	Enable FM deviation
dev_sh[3:0]	REG_40H[11:8]	FM deviation coarse tuningÿ 0000=max, 1111=min

Launch block diagram

dev_lvl[7:0]	REG_40H[7:0]	FM deviation fine tuningÿ 0000=min, 1111=max, GAIN=(256+dev_lvl[7:0])>>dev_sh[3:0]

If only used as VCO, you can set dev_en=0 to disable FM modulation. For details on modulation limits, see the document "Setting Modulation Limits and MIC Sensitivity" section.

(20) Transmit power control

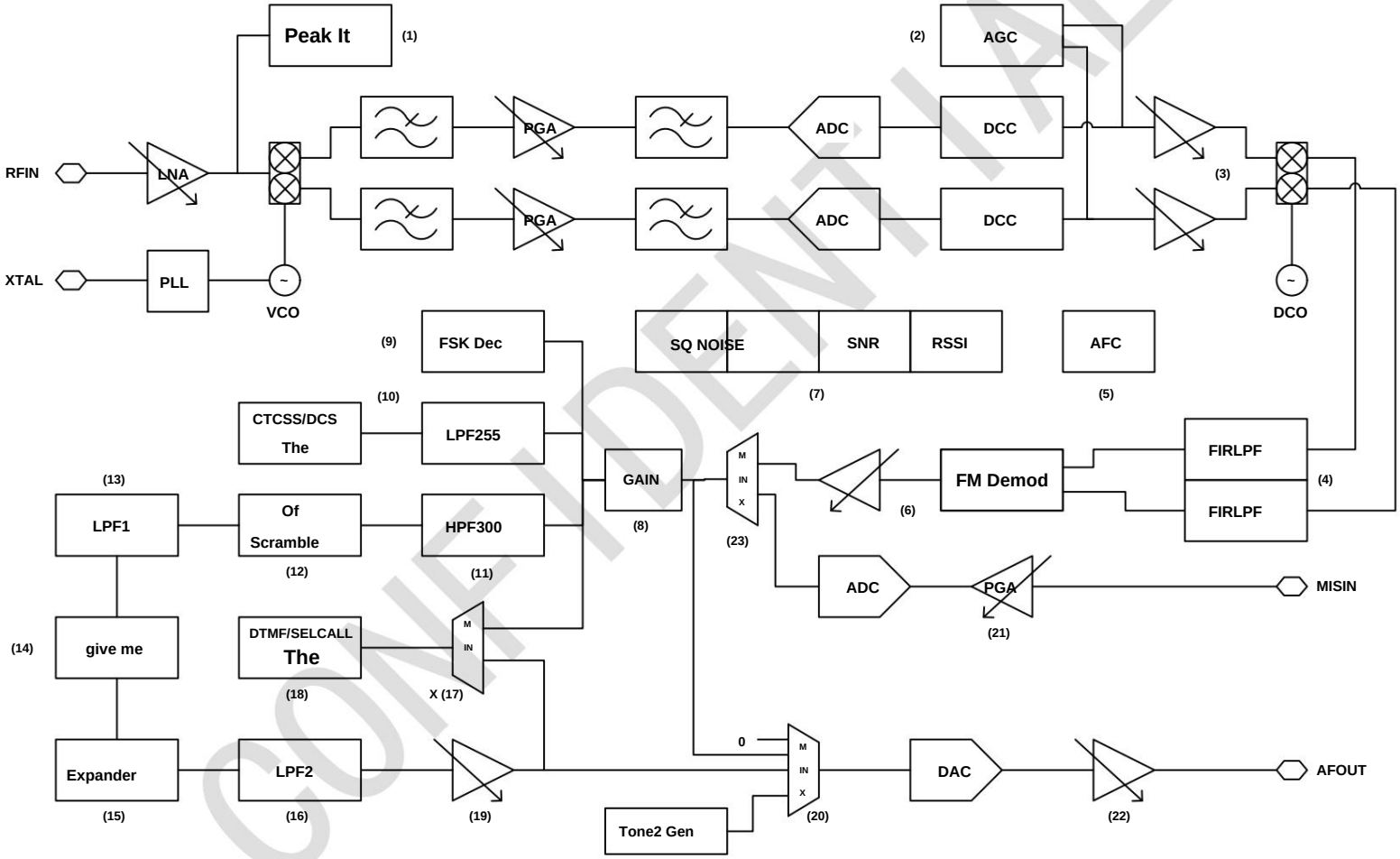
For detailed instructions, see the "Setting Transmit Power" section of the document.

(21) Voice control (VOX) detection

For details, see the "Voice Control (VOX), Transmit Timeout (TOT) Settings" section of the document.

Receiving block diagram

Receiving block diagram



Receiving block diagram

Block diagram description:

(1) Analog RF front-end and gain

Register	Address(HEX)	Description
lna_peak_rssi[7:0]	REG_62H[7:0]	RSSI after LNA, 1dB/step. (Read Only)

Broadband signal strength indication after LNA.

(2) Receive AGC

Register	Address(HEX)	Description
agc_rssi[7:0]	REG_62H[15:8]	RSSI after DCC, 1dB/step. (Read Only)

Narrowband digital signal strength indication after ADC and DCC.

(3) Receive IQ digital gain 1

Register	Address(HEX)	Description
dig_gain_rx[4:0]	REG_7DH[12:8]	Gain after AGC, digital down conversion. 1dB/ step

(4) Receive IQ filter, digital gain 2

Register	Address(HEX)	Description
firlpf_bw[2:0]	REG_43H[14:12]	RF filter bandwidth (Apass=0.1dB) 000 = 1.7 kHz 001 = 2 kHz 010 = 2.5 kHz 011 = 3 kHz 100 = 3.75 kHz 101 = 4 kHz 110 = 4.25 kHz 111 = 4.5 kHz if wb=1, firlpf_bw *=2;
firlpf_bw_for_weak[2:0]	REG_43H[11:9]	RF filter bandwidth when signal is weak.
wb	REG_43H[5]	1 = 25kHz/20kH; 0 = 12.5kHz/6.25kHz
firlpf_gain[1:0]	REG_43H[1:0]	Gain after FIR LPF

		00=0dB; 01=6dB; 10=12dB; 11=18dB

(5) Automatic Frequency Calibration (AFC)

See the "AFC Settings" section of the document for detailed instructions.

(6) FM demodulation and gain

Register	Address(HEX)	Description
fmdem_gain[1:0]	REG_44H[12:11]	Gain after FM Demodulation 00=0dB; 01=6dB; 10=12dB; 11=18dB

(7) SQÿNOISEÿSNRÿRSSI

See the Squelch (SQ) Settings and RSSI, NOISE and SNR section of the document for detailed instructions.

(8) Receive DISC signal gain

Register	Address(HEX)	Description
~~audio_rx_gain1[5:0]~~	~~REG_96H[13:8]~~	~~0=mute, 63=max, 0.5dB/step~~

The DISC signal contains both audio and sub-audio, so this register adjusts both audio and sub-audio gains.

(9) Receive FSK decoding

See the "FSK Mode Settings" section of the document for details.

(10) Sub-audio low-pass filter, sub-audio decoding

Register	Address(HEX)	Description
~~subau_rx_dcc_bw[2:0]~~	~~REG_B6H[12:10]~~	~~CTC/DCS Rx HPF BW~~ ~~000=bypass;001=60Hz;010=30Hz;011=15Hz;~~

		~~100=8Hz;101=4Hz;110=2Hz;111=1Hz~~
subau_tx_atten_gain[1:0]	REG_B6H[9:8]	CTC/DCS Tx Atten Gain, -6dB/step
subau_rx_gain1[1:0]	REG_B6H[7:6]	CTC/DCS Rx Gain1, -6dB/step
subau_rx_gain2[1:0]	REG_B6H[5:4]	CTC/DCS Rx Gain2, -6dB/step
subau_rx_gain3[3:0]	REG_B6H[3:0]	CTC/DCS Rx Gain3, -6dB/step

For detailed instructions on sub-audio decoding, see the "Sub-audio Settings" section of the document.

(11) Receive voice 300Hz high-pass filter

Register	Address(HEX)	Description
audio_hpf_rx_bypass	REG_9BH[9]	1=Audio HPF 300Hz bypass for RX

(12) Receive voice descrambling

See the "SCRAMBLE Mode Settings" section of the document for details.

(13) Receive speech low-pass filter 1

Register	Address(HEX)	Description
audio_lpf1_rx_bypass	REG_9BH[8]	1=Audio LPF1 bypass for RX

(14) Receive speech de-emphasis

Register	Address(HEX)	Description
audio_emph_rx_bypass	REG_9BH[6]	1=DE-EMPHASIS bypass

(15) Receive speech despreading

See the "Voice Mode Settings" section of the documentation for details.

(16) Receive speech low-pass filter 2

Register	Address(HEX)	Description

audio_lpf2_rx_bypass	REG_9BH[7]	1=Audio LPF2 bypass for RX

(17) Receive DTMF/SELCALL input selection

Register	Address(HEX)	Description
dtmf_in_sel	REG_94H[5]	1=select gain out 0=select audio rx out

DTMF/SELCALL receive signal selection, the difference lies in whether the signal entering DTMF/SELCALL decoding is de-emphasized. can be set according to

Choose according to your needs. Default dtmf_in_sel=1.

(18) Receive DTMF/SELCALL decoding

For details, see the "DTMF Mode Settings" and "SELCALL Mode Settings" sections of the document.

(19) Receive voice volume control

Register	Address(HEX)	Description
audio_rx_gain_sh [2:0]	REG_53H[15:13]	000=max,111=min 0~-42dB, 6dB/step, digital gain
audio_rx_gain[4:0]	REG_53H[9:5]	0=mute,31=max 1dB/step, digital gain

(20) AF DAC channel selection

Same as the launch block diagram (17).

(21) MIC channel PGA analog gain

Same as the launch block diagram (1).

(22) AF DAC analog gain

Same as the launch block diagram (18).

Receiving block diagram

(23) Receive voice baseband mode selection

The received FM demodulated DISC signal is sent to the chip through MICIN for voice, sub-audio, DTMF, 5TONE and other decoding. in detail

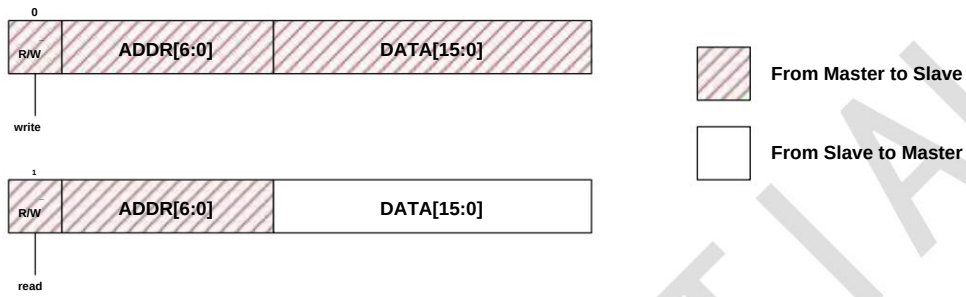
See the "DISC Baseband Processing Mode Settings" section of the documentation.

Interface timing

Interface timing

3-WIRE interface

The format is as follows:



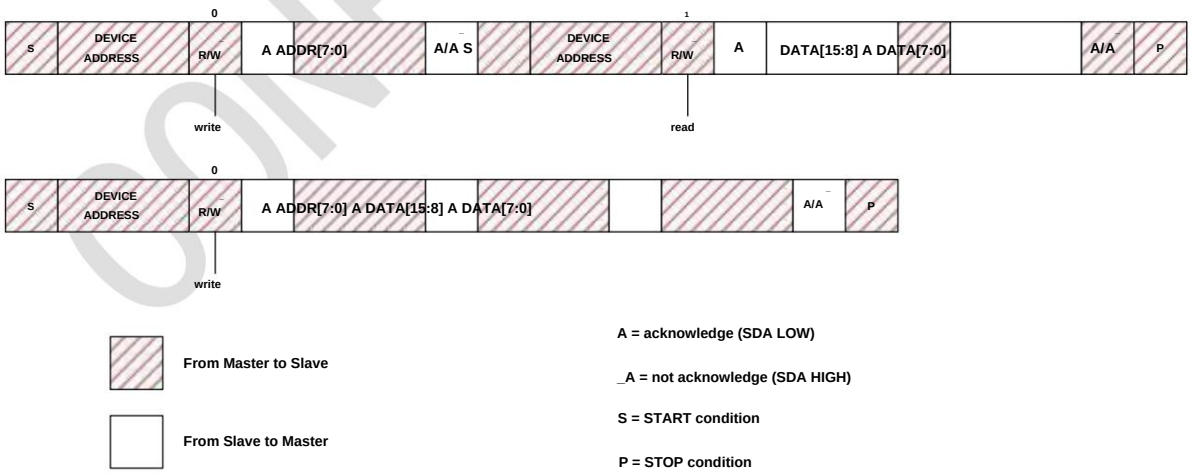
Tip: For 3-wire interface, when the read and write register address is greater than 0x7F, page flip operation is required. as follows:

Page0: WRITE(0x7F,0x0000); //default

Page1: WRITE(0x7F,0x0001);

2-WIRE interface

ÿÿÿÿ(Device Address =‘0101110’ when SCN is high(or no SCN pin), or Device Address = ‘1110001’when SCN is low)ÿ



Reminder: 2-wire interface read and write registers do not need page turning operation, and "#define I2C 0x80".

Configuration process

Configuration process

1. Initialize settings

```
WRITE(0x00,0x8000); //soft reset

WRITE(0x00,0x0000);

...
```

2. Set the crystal/crystal clock frequency

See "Crystal/Crystal Clock Frequency Settings" section for instructions.

3. Set the frequency band

See the description in the section "Frequency, Frequency, Bandwidth Mode Settings".

4. Set the in-band frequency point

See the description in the section "Frequency, Frequency, Bandwidth Mode Settings".

5. Set up the AGC table

6. OFFSET calibration setting sequence

```
WRITE(0x03,0xBDF1); //Open the receiving channel, not open AF

WRITE(0x04,REG_04H & 0xFCFF);

WRITE(0x04,REG_04H | 0x0300); //bit[9:8] is the OFFSET calibration enable bit

delay_ms(50);

WRITE(0x03,0x0000); //Enter IDLE after calibration
```

Tip: Every time you change the frequency band, you need to set the frequency point in the frequency band first, and then do the OFFSET calibration again.

7. Set wide/narrowband, sub-audio and INBAND signal modes

8. Enter the receiving/transmitting state

Send and receive status settings

Receive (RXON) settings

WRITE(0x03,0x0000);

WRITE(0x03,0xBFF1);

Transmit (TXON) Settings

WRITE(0x03,0x0000);

WRITE(0x03,0xC1FE);

launch with sidetone

WRITE(0x03,0x0000);

WRITE(0x03,0xC1FE | 1<<9); //Enable AF

No emission, only sidetone

WRITE(0x03,0x0002 | 1<<9); //Enable AF

IDLE settings (can be used to save power)

WRITE(0x03,0x0000);

Tip: Power saving mode can be achieved by switching between RXON and IDLE.

Send and receive status settings

Attachment: register description

Register	Address(HEX)	Description
vco_cal_en	REG_03H[15]	1=Enable PLL Calibration, (0->1)
pabias_en	REG_03H[14]	1=Enable PABIAS Output
rxlink_en[3:0]	REG_03H[13:10]	1111=Enable Rx Link, (LNA,MIXER,FILTER,ADC)
error_en	REG_03H[9]	1=Enable AFOUT DAC
pll_en[4:0]	REG_03H[8:4]	11111=Enable PLL, VCO
padrv_en	REG_03H[3]	1=Enable PADRV
micin_en	REG_03H[2]	1=Enable MICIN ADC
thon	REG_03H[1]	1=Enable Tx DSP
rxon	REG_03H[0]	1=Enable Rx DSP

Band, frequency point, bandwidth mode settings

Band, frequency point, bandwidth mode settings

Transmit and receive frequency setting

For example, set the frequency to 409.75MHz, then DEC2HEX(409.75*100000)=0x2713A98

WRITE(0x38,0x3A98);

WRITE(0x39,0x0271);

Tip: After setting the frequency every time, you need to re-RXON or TXON, the frequency will be switched; the minimum frequency resolution is 10Hz.

Broadband (25KHZ) mode setting

~~WRITE(0x3A,0x00E0);~~

WRITE(0x43,0x3028);

Tip: You can choose the appropriate transmit and receive bandwidth, the above are only recommended values.

Narrowband (12.5KHZ) mode setting

~~WRITE(0x3A,0x0040);~~

WRITE(0x43,0x4048);

Tip: You can choose the appropriate transmit and receive bandwidth, the above are only recommended values.

Attachment: register description

Register	Address(HEX)	Description
freq [15:0]	REG_38H[15:0]	Frequency =(REG_39H<<16 + REG_38H)*10 Hz
freq [31:16]	REG_39H[15:0]	
firlpf_bw[2:0]	REG_43H[14:12]	RF filter bandwidth (Apass=0.1dB) 000 = 1.7 kHz 001 = 2 kHz 010 = 2.5 kHz 011 = 3 kHz 100 = 3.75 kHz 101 = 4 kHz

Band, frequency point, bandwidth mode settings

		110 = 4.25 kHz 111 = 4.5 kHz if wb=1, firlpf_bw *=2;
firlpf_bw_for_weak[2:0]	REG_43H[11:9]	RF filter bandwidth when signal is weak.
audio_lpf2_tx_sel[2:0]	REG_43H[8:6]	Audio LPF bandwidth (Apass=1dB) for Tx 100 = 4.5 kHz 101 = 4.25 kHz 110 = 4 kHz 111 = 3.75 kHz 000 = 3 kHz 001 = 2.5 kHz 010 = 2 kHz 011 = 1.7kHz
wb	REG_43H[5]	Rx filter band width mode select 1=25khz/20khz wide band mode, 0=12.5khz narrow band mode

Attachment: frequency band setting table

BAND	REG_1AH	REG_40H 24M~26M	REG_40H 12M~13M	REG_40H 18M~20M
Band_740M_1120M	0x1f80	0x3ad0	0x37d0	0x373a
Band_370M_560M	0x2f50	0x39d0	0x38d0	0x383a
Band_247M_373M	0x9f30	0x385c	0x375c	0x38d7
Band_185M_280M	0x3f48	0x38d0	0x37d0	0x373a
Band_124M_186M	0xaf28	0x375c	0x365c	0x37d7
Band_93M_140M	0x4f44	0x37d0	0x36d0	0x363a
Band_62M_93M	0xbf24	0x365c	0x355c	0x36d7
Band_46M_70M	0x5f42	0x36d0	0x35d0	0x353a
Band_31M_46M	0xcf22	0x355c	0x345c	0x35d7
Band_23M_35M	0x6f41	0x35d0	0x34d0	0x343a
Band_16M_23M	0xdf21	0x345c	0x335c	0x34d7

Set modulation limits and MIC sensitivity

Set modulation limits and MIC sensitivity

When the chip is only used as a VCO, you can set dev_en=0 to turn off the FM modulation, so as to prevent the MICIN signal from affecting the local oscillator output.

The value of dev_sh is in a 2-fold decreasing relationship. For example, if dev_lvl is set to the same value, the modulation of dev_sh=7 is twice the modulation of dev_sh=8.

The suggested tuning sequence for modulation limit, MIC sensitivity and frequency response is:

1. Modulation limit (input a larger MICIN signal, such as 200mv, set dev_sh and dev_lvl to make the maximum modulation meet the design requirements)

2. Transmitting radio frequency (follow the setting instructions of transmitting radio frequency 300Hz and 3KHz to set accordingly. Because changing the frequency response will affect the MIC sensitivity, so to be placed in front)

3. MIC sensitivity (Debug mic_sens_gain to make the MIC sensitivity meet the design requirements)

Tip: Different frequency band modulation registers (REG_40H) need to be set with different values. In addition, since it will affect the modulation size of the sub-audio, it is necessary to first After determining the value of REG_40H, adjust the sub-audio transmit gain (REG_51H).

Attached: Modulation Limit and MIC Sensitivity Register

Register	Address(HEX)	Description
dev_in	REG_40H[12]	Enable FM deviation
dev_sh[3:0]	REG_40H[11:8]	FM deviation coarse tuningÿ 0000=max, 1111=min
dev_lvl[7:0]	REG_40H[7:0]	FM deviation fine tuning ÿ 0000=min, 1111=max, GAIN=(256+dev_lvl[7:0])>>dev_sh[3:0]
~~mic_sens_gain[5:0]~~	~~REG_97H[13:8]~~	~~MIC sensitivity adjust gain 0=mute,~~ ~~63=max, 0.5dB/step~~

~~ÿ For example, the current dev_lvl=0xA3, the modulation size is 2.1kHz. If you want to modify the modulation to 2.2khz, then the calculation method is as~~ follows:

~~dev_lvl= (0xA3+256)/2.1*2.2-256 -=0xB7;~~

If the adjustment range is large, you need to modify dev_sh[3:0], which is a monotonically decreasing relationship of 2 times.

Set transmit power

Set transmit power

The transmit power is controlled by the registers padrv_gain and pa_gain_vreg, and the PA Bias can be output to control the external PA power.

Attachment: Transmit Power Register

Register	Address(HEX)	Description
pabias_in	REG_03H[14]	1=Enable PAbias output PAbias
pabias_out[3:0]	REG_19H[3:0]	output 1.3~2.8V, 100mv/step 0000=1.3V ... 1111=2.8V
padrv_gain[2:0]	REG_28H[5:3]	111(max)->000(min)
pa_gain_vreg[2:0]	REG_28H[2:0]	111(max)->000(min)

Power (dB)	Padrv_gain[2:0] 100							
Pa_gain_vreg[2:0]	111	110	101	011		010	001	000
111	8.33	7.96	7.53	7.11	6.62	5.69	4.12	1.59
110	7.96	7.11	6.62	6.19	5.69	4.68	2.16	-2.16
101	7.11	6.19	6.19	5.23	4.68	3.10	0.84	-5.54
100	6.40	5.69	5.23	4.68	3.44	2.16	-0.67	-8.61
011	5.70	4.68	4.12	3.44	2.16	0.85	-3.62	-12.78
010	4.10	3.44	2.16	1.59	0.15	-2.84	-8.2	-19.53
001	2.80	1.59	0.84	-0.67	-2.16	-5.5	-11.4	-22.65
000	1.60	0.84	-0.67	-2.16	-4.28	-8.2	-13.7	-24.00

Receive mute (MUTE) and volume settings

Receive mute (MUTE) and volume settings

To set mute:

WRITE(0x47,(REG_47H & 0xE0FF) | 0x1000);

To set receive AF output:

WRITE(0x47,(REG_47H & 0xF0FF) | 0x1100);

To set the transmit sidetone output:

WRITE(0x47,(REG_47H & 0xE0FF) | 0x1800);

Attachment: mute, mute off, BEEP sidetone selection register

Register	Address(HEX)	Description
afout_mode[4:0]	REG_47H[12:8]	0x10 = MUTE 0x11 = RX AFOUT 0x18 = BEEP/ TX Side ToneÿCTCSS/CDCSS is not includeÿ 0x15 = RX ALARM TONE

Attachment: Receive volume register

Register	Address(HEX)	Description
audio_rx_gain_sh[2:0]	REG_53H[15:13]	000=max,111=min 0~-42dB, 6dB/step, digital gain
audio_rx_gain[4:0]	REG_53H[9:5]	0=mute,31=max, 1dB/step, digital gain
dac_vgain[3:0]	REG_23H[3:0]	DAC maximum output level

~~Tip: You can use the registers audio_rx_gain_sh and audio_rx_gain to combine the volume control gear, while dac_vgain is~~ fixed at a ~~Gear (generally fixed at the maximum gear).~~

Audio Response Adjustment

Audio Response Adjustment

low frequency audio response

Increase the transmit 300Hz amplitude:

```
WRITE(0x7F,0x0001); //page=1
WRITE(I2C | 0x30,0x8942);
WRITE(I2C | 0x31,0x3751);
WRITE(0x7F,0x0000); //page=0
```

Increase receive 300Hz amplitude:

```
WRITE(0x2C,0x8942);
WRITE(0x2D,0x3751);
```

high frequency audio response

Increase the transmit 3 KHz amplitude:

```
WRITE(0x7F,0x0001); //page=1
WRITE(I2C | 0x23,0xC1BB);
WRITE(I2C | 0x24,0x2226);
WRITE(0x7F,0x0000); //page=0
```

Increase the receive 3 KHz amplitude:

```
WRITE(0x7F,0x0001); //page=1
WRITE(I2C | 0x21,0xBBC0);
WRITE(I2C | 0x22,0x2616);
WRITE(0x7F,0x0000); //page=0
```

Voice Mode Settings

Voice Mode Settings

Attachment: Voice related registers

Register	Address(HEX)	Description
audio_emph_rx_bypass	REG_9BH[6]	DE-EMPHASIS bypass
audio_emph_tx_bypass	REG_9BH[0]	PRE-EMPHASIS bypass
audio_hpf_rx_bypass	REG_9BH[9]	Audio HPF 300Hz bypass for RX
audio_hpf_tx_bypass	REG_9BH[3]	Audio HPF 300Hz bypass for TX
audio_lpf1_rx_bypass	REG_9BH[8]	Audio LPF1 bypass for RX
audio_lpf1_tx_bypass	REG_9BH[2]	Audio LPF1 bypass for TX
audio_lpf2_rx_bypass	REG_9BH[7]	Audio LPF2 bypass for RX
audio_lpf2_tx_bypass	REG_9BH[1]	Audio LPF2 bypass for TX
cmpd_rx_factor[2:0]	REG_98H[14:12]	Expanding Factor. 100=1:3;011=1:2.5;010=1:2;001=1:1.5
cmpd_rx_th_high[5:0]	REG_98H[11:6]	Above this amplitude point, audio will be expanded according to the expanding factor. The unit of this threshold is 2dB.
cmpd_rx_th_low[5:0]	REG_98H[5:0]	Under this amplitude point, audio will be attenuated. The unit of this threshold is 2dB.
cmpd_rx_gain[4:0]	REG_96H[7:3]	Audio CMPD Gain, 1dB/step
cmpd_rx_bypass[2:0]	REG_96H[2:0]	Audio Compander bypass for RX 110=bypass with gain 001=bypass without gain 000=CMPD ON for RX Others=reserved
cmpd_tx_factor[2:0]	REG_99H[14:12]	Compressing Factor. 111=8:1;110=4:1;100=2:1;000=1:1
cmpd_tx_th_high[5:0]	REG_99H[11:6]	Above this amplitude point, audio will be compressed according to the compressing factor. The unit of this threshold is 2dB.
cmpd_tx_th_low[5:0]	REG_99H[5:0]	Under this amplitude point, audio will be attenuated. The unit of this threshold is 2dB.
cmpd_tx_gain[4:0]	REG_97H[7:3]	Audio CMPD Gain, 1dB/step
cmpd_tx_bypass[2:0]	REG_97H[2:0]	Audio CMPD bypass for TX

		110=bypass with gain 001=bypass without gain 000=CMPD ON for TX Others=reserved
cmpd_ct_intvl[5:0]	REG_9AH[13:8]	Companding Amplitude Detect Interval, 0.64ms/step
cmpd_atk_step[3:0]	REG_9AH[7:4]	Companding Gain Attack Speed. 0000=most fast ... 1111=most slow
cmpd_rls_step[3:0]	REG_9AH[3:0]	Companding Gain Release Speed. 0000=most fast ... 1111=most slow
cmpd_db_out[6:0]	REG_87H[6:0]	Audio amplitude output, 1dB/step. (Read Only)

Companding point setting method

Transmit companding point: add the micin signal amplitude to the companding point amplitude, read cmpd_db_out (REG_87H[6:0]), divide the resulting value by 2, set

Go to cmpd_tx_th_high (REG_99H[11:6]).

Receive despreading point: Set the signal source modulation frequency offset to the companding point frequency offset, read cmpd_db_out (REG_87H[6:0]), divide the obtained value by 2, set

Set it to cmpd_rx_th_high (REG_98H[11:6]).

SCRAMBLE mode setting

SCRAMBLE mode setting

Attachment: Scrambling and frequency setting register

Register	Address(HEX)	Description
scramb_en	REG_04H[5]	Enable SCRAMBLER
scramb_freq[15:0]	REG_54H[15:0]	SCRAMBLE frequency control word =3.3(KHz)*2^26/6500 - The scrambler inversion mixing frequency should be kept between 2.6kHz and 3.5kHz

DTMF mode settings

DTMF mode settings

Attachment: DTMF/SELCALL Receive Register

Register	Address(HEX)	Description
dtmf_code_ready	REG_0BH[12]	1=DTMF/ SELCALL symbol received ready. (Read Only)
dtmf_code[3:0]	REG_0BH[11:8]	DTMF/ SELCALL symbol received. (Read Only)
dtmf_det_th[5:0]	REG_94H[12:7]	DTMF/ SELCALL detect threshold, 1dB/step
dtmf_symbol_mode	REG_94H[6]	DTMF/ SELCALL symbol mode 1=”symbol”+”idle”+”symbol”+… (DTMF) 0=”symbol”+”symbol”+”symbol”+…(5TONE)
dtmf_in_sel	REG_94H[5]	1=select gain out 0=select audio rx out
dtmf_mode	REG_94H[4]	Dual/Single Tone Detect mode 1=dual tone 0=single tone
dtmf_symbol_max[3:0]	REG_94H[3:0]	SELCALL maximum symbol number 15=symbol “0”~”F” (DTMF) 14=symbol “0”~”E” (THIS) ...

Standard DTMF Symbol Table

DTMF Symbol		High Frequency (Hz)			
		1209	1336	1477	1633
Low Frequency (Hz)	697	‘1’	‘2’	‘3’	‘A’
	770	‘4’	‘5’	‘6’	‘B’
	852	‘7’	‘8’	‘9’	‘C’
	941	'AND'	‘0’	‘F’	‘D’

Launch process:

1. Set transmit mute: WRITE(0x50,REG_50H | 0x8000);

DTMF mode settings

2. Set the frequency and gain of TONE1 and TONE2

3. Delay T1

4. Set the transmission on: WRITE(0x50,REG_50H);

5. After delay T2, if DTMF transmission continues to jump to 1, transmit the next DTMF symbol

0	TONE1+TONE2	0	TONE1+TONE2	...
T1	T2	T1	T2	...

Receiving process:

1. Wait for the interrupt, or read the dtmf_code_ready register until = 1

2. Read dtmf_code[3:0] to get the received DTMF symbol, if the reception is not over, jump to 1 and continue to wait for the next DTMF symbol

SELCALL mode setting

SELCALL mode setting

Launch process:

1. Set transmit mute: WRITE(0x50,REG_50H | 0x8000);

2. Delay T1

3. Set the launch to open: WRITE(0x50,REG_50H);

4. Set the frequency and gain of TONE1

5. After delay T2, if SELCALL transmission is not over, jump to 4 and transmit the next SELCALL symbol

0	TONE1	TONE1	TONE1	...
T1	T2	T2	T2	...

Receiving process:

1. Wait for the interrupt, or read the dtmf_code_ready register until = 1

2. Read dtmf_code[3:0] to get the received SELCALL symbol, if the reception is not over, jump to 1 and continue to wait for the next SELCALL symbol

1050HZ single tone decoding

1050Hz single tone decoding

Using DTMF for 1050 Hz alarm decoding, the configuration is as follows:

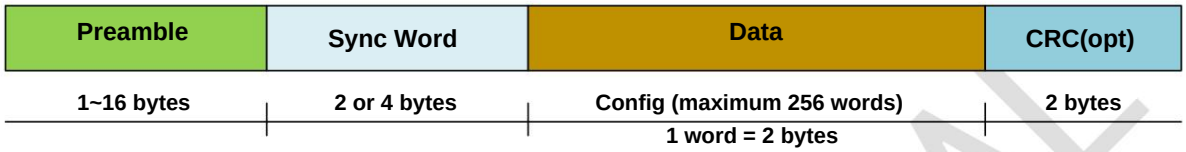
```
WRITE(0x04,0x0302);

WRITE(0x09,0x0002);

WRITE(0x09,0x105A);

WRITE(0x09,0x204B);

WRITE(0x09,0x3066);

WRITE(0x09,0x403B);

WRITE(0x07,0x1021);

WRITE(0x51,0x9400);

WRITE(0x52,0x1cb2);

WRITE(0x54,0x3269);

WRITE(0x7F,0x0001); //page1 for SPI

WRITE(0x11 | I2C,0x06FD);

WRITE(0x14 | I2C,0x8024 | 10<<7); //[12:8] decode threshold, change @2019.04.22

WRITE(0x7F,0x0000); //page0 for SPI


//polling REG_0B and REG_60,

//if REG_0B[12:8]==0x11 && REG_60[0]==1

//1050Hz is detectd.
```

FSK mode settings

FSK mode settings

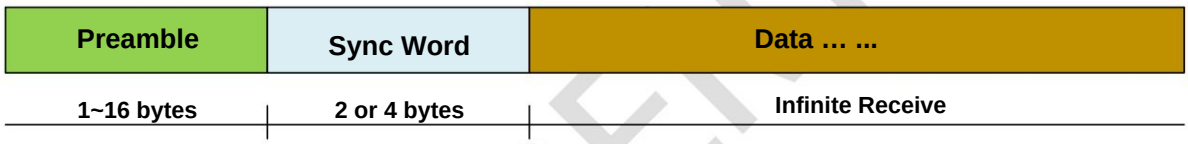
FSK frame format 1 (fixed length)

This mode supports fixed-length data frame transmission, and CRC operation is optional.



FSK frame format 2 (unlimited reception)

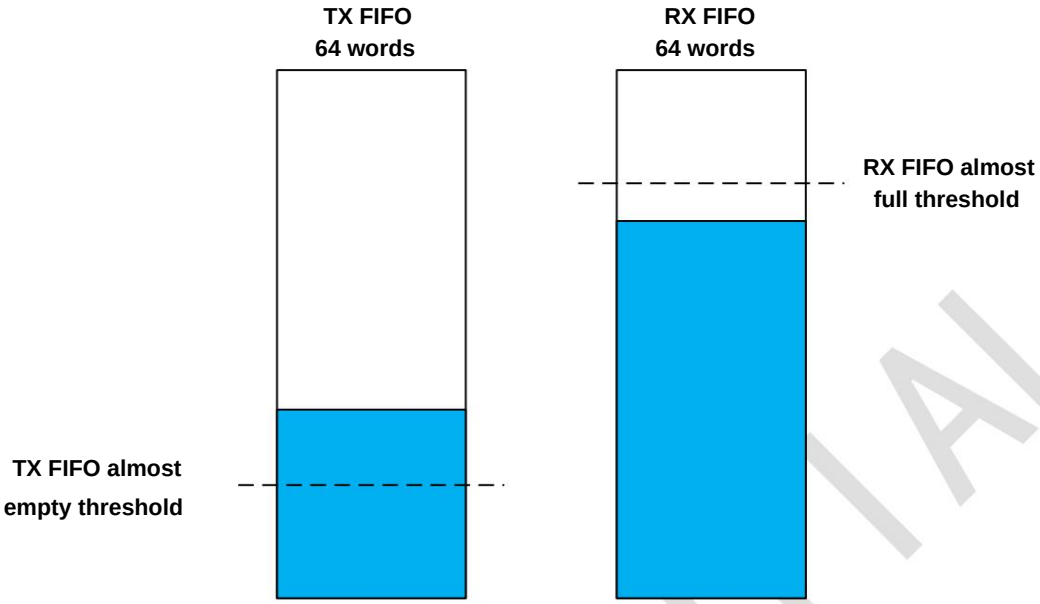
This mode supports the transmission of larger data. In this case, the length of the data packet is unknown, and the device will permanently import the data after the sync word into the FIFO. but not CRC operation is supported because the corresponding device at the end of the data packet is unknown. This mode is more flexible and is suitable for user-defined frame structure and frame length.



TX and RX FIFO

The chip integrates two FIFOs of 64 words (=128 bytes), one for TX and one for RX (as shown in the figure below). by writing REG_5FH register to write data to TX FIFO, read RX FIFO data by reading REG_5FH register. TX FIFO Yes fsk_tx_fifo_ae_th[5:0] (TX FIFO Almost Empty) can be set to prompt data writing to TX FIFO by generating an interrupt; RX FIFO has fsk_rx_fifo_af_th[5:0] (RX FIFO Almost Full) that can be set, and prompts to count the RX FIFO by generating an interrupt. read. At the same time, TX FIFO and RX FIFO each have a clear register which can be used to clear or reset TX FIFO and RX FIFO.

FSK mode settings



Attachment: FSK setting register

Register	Address(HEX)	Description
fsk_en	REG_04H[4]	Enable FSK mode
fsk_rate[15:0]	REG_54H[15:0]	FSK/Scramble use the same rate register, =freq(kHz)*2^26/6500
fsk_crc_polyn[15:0]	REG_56H[15:0]	CRC polynomial coefficient, CCIT-16 is default
fsk_tx_fifo_clear	REG_59H[15]	Clear TX FIFO, 1=clear
fsk_rx_fifo_clear	REG_59H[14]	Clear RX FIFO, 1=clear
fsk_scramble_en	REG_59H[13]	1=Enable FSK Scramble
fsk_rx_en	REG_59H[12]	1=Enable FSK TX
fsk_tx_en	REG_59H[11]	1=Enable FSK RX
fsk_rx_data_inv	REG_59H[10]	1=invert FSK data after RX
fsk_tx_data_inv	REG_59H[9]	1=invert FSK data before TX
fsk_prmb_size[3:0]	REG_59H[7:4]	FSK preamble length select, length=(fsk_prmb_size[3:0]+1) bytes
fsk_sync_size	REG_59H[3]	FSK sync length select, 1=4 bytes (fsk_sync_byte0,1,2,3)

FSK mode settings

		0=2 bytes (fsk_sync_byte0,1)
fsk_sync_byte0[7:0]	REG_5AH[15:8]	FSK sync byte0
fsk_sync_byte1[7:0]	REG_5AH[7:0]	FSK sync byte1
fsk_sync_byte2[7:0]	REG_5BH[15:8]	FSK sync byte2
fsk_sync_byte3[7:0]	REG_5BH[7:0]	FSK sync byte3
fsk_tx_type_byte[7:0]	REG_5CH[7:0]	FSK type byte to be transmitted. “0x01“=FSK Format2 “0x21”=FSK Format1 without CRC “0x61”=FSK Format1 with CRC
fsk_length[7:0]	REG_5DH[15:8]	FSK data length when use FSK format1 length=(fsk_length[7:0]+1) words (1 word = 2 bytes)
fsk_tx_fifo_ae_th [5: 0]	REG_5EH[11:6]	FSK TX FIFO almost empty threshold(unit is 1word=2 bytes)
fsk_rx_fifo_af_th[5:0]	REG_5EH[5:0]	FSK RX FIFO almost full threshold(unit is 1word=2 bytes)
fsk_data[15:0]	REG_5FH[15:0]	TX/RX FIFO Data

Launch process:

1. Clear TX FIFO, then write data to TX FIFO

2. Transmit TXON

3. Wait for the interrupt to prompt the completion of the FSK transmission

4. End transmitting IDLE

Receiving process:

1. Receive RXON

2. Wait for the interrupt prompt to receive completion

3. Read the RX FIFO data and read the fsk_crc_ok flag for error control

4. End receiving IDLE, (delay for a certain interval time) RXON again and wait for the arrival of the next FSK frame

DISC baseband processing mode setting

DISC baseband processing mode setting

1. This mode is only used for baseband processing of voice and sub-audio transceivers. The baseband signals of receiving DISC and transmitting MIC are all connected by the MICIN port (the input of MICIN).

The amplitude of the received DISC signal needs to be controlled below 200mV) and the input settings are as follows:

WRITE(0x44,0x53EC | 0x1); //[0] DISC mode enable

WRITE(0x7E,0x341E | 3<<6); //DCC bypass

WRITE(0x7F,0x0001); //page1

WRITE(I2C | 0x1C,0x076F);

WRITE(I2C | 0x1D,0xB9FD);

WRITE(I2C | 0x27,0x18DA);

WRITE(0x7F,0x0000); //page0

RXON:

WRITE(0x03,0x0000); WRITE(0x03,0x0005);

TXT:

WRITE(0x03,0x0000); WRITE(0x03,0x0006 | 1<<9); //Open AF

or

WRITE(0x03,0x0000); WRITE(0x03,0xC1FE); // directly modulate to RFO and transmit

2. Do voice and sub-audio receiving and decoding (the input signal amplitude of MICIN needs to be controlled below 200mV), and also used as VCO. set as Down:

WRITE(0x13,0x03FF);

WRITE(0x35,0xF108);

WRITE(0x3C,REG_3CH & 0xFFC0);

WRITE(0x3D,0x0000);

DISC baseband processing mode setting

WRITE(0x44,0x53EC | 0x1); //[0] DISC mode enable

WRITE(0x7E,0x341E | 3<<6); //DCC bypass

WRITE(0x7F,0x0001); //page1

WRITE(I2C | 0x1C,0x076F);

WRITE(I2C | 0x1D,0xB9FD);

WRITE(I2C | 0x27,0x18DA);

WRITE(0x7F,0x0000); //page0

RXON:

WRITE(0x04,0x0800);

WRITE(0x03,0x0000); WRITE(0x03,0x81FD); //RFO outputs VCO, AF outputs audio, and sub-audio decoding is done internally

TXT:

WRITE(0x03,0x0000); WRITE(0x03,0x0006 | 1<<9); //Only transmit audio filter, output from AF

or

WRITE(0x04,0x0400);

WRITE(0x03,0x0000); WRITE(0x03,0xC1FE)//normally modulate and transmit from RFO

BYPASS mode setting (for DMR/dPMR)

This mode can be used as a transceiver for DMR/dPMR and other products. MICIN sends the shaped filtered 4FSK signal for modulation and transmission, and AFOUT sends out

The adjusted 4FSK signal. The settings are as follows (the chip should be set to narrowband 12.5k mode):

WRITE(0x04,0x0300);

~~WRITE(0x3a,0x0040); // 12.5k~~

~~WRITE(0x43,0x7e09);~~ ->WRITE(0x43,0x7e08); // Set DMR bandwidth bit14:12

WRITE(0x48,0x6c5c); // cic mode

WRITE(0x4b,0xe442); // cic mode

WRITE(0x73,0x569a); //Close AFC

WRITE(0x7E,0x341E | 1<<7); //dcc_tx_bypass=1

If the transmission is realized by adjusting the external TCXO, it is also necessary to set

WRITE(0x40,0x0000); //Turn off internal modulation

In addition to the above settings, the transceiver control

TXON changed to

WRITE(0x47,0x6740); //Transmit directly after MIC ADC

WRITE(0x03,0x0000);

WRITE(0x03,0xC1FE);

RXON changed to

WRITE(0x47,0x6140); //Send AFOUT directly after FM demodulation

WRITE(0x03,0x0000);

~~WRITE(0x03,0xC1FE);~~ ->WRITE(0x03,0xBFF1);

VCO Mode Settings

VCO Mode Settings

Both receive and transmit are used as VCO

1. Modify the corresponding registers in the initialization

WRITE(0x34,0x606F); WRITE(0x35,0xF608); //Turn off the corresponding clock to reduce VCO interference

WRITE(0x40,0x0000); //Turn off modulation

2. Both TXON and RXON are used

WRITE(0x03,0x0000);

WRITE(0x03,0x01FA); WRITE(0x03,0x81FA);

3. Modify the transceiver frequency band according to your needs

Receive only for VCO use, transmit link is used normally

1. Modify the corresponding registers in the initialization

WRITE(0x34,0x206F); //Turn off the receiving part of the clock to reduce VCO interference

2. TXON is

WRITE(0x03,0x0000);

WRITE(0x40,????); //The value of normal modulation

WRITE(0x03,0xC1FE);

3. RXON is

WRITE(0x03,0x0000);

WRITE(0x40,0x0000); //Turn off modulation

WRITE(0x03,0x01FA); WRITE(0x03,0x81FA);

4. Modify the transceiver frequency band according to your needs

Key tone side tone BEEP and other settings

Key tone side tone on:

WRITE(0x03,0x0202); //enable AF and Tx(PLL and PA are not include)

WRITE(0x47,(REG_47H & 0xE0FF) | 0x1800);

The BEEP and CALL TONE frequencies of touch tone side tone are set by registers REG_71H, REG_72H:

Dual tone settings tone1_gen=1, tone2_gen=1

Mono set tone1_gen=1

The transmission frequency is freq(Hz), and the calculation formula is =freq(kHz)* 2^26/6500

To transmit CALL TONE and play sidetone at the same time, you need to set AFOUT(REG_03[9])=1, and set

WRITE(0x47,(REG_47H & 0xE0FF) | 0x1800);

To launch a MUTE:

WRITE(0x50,REG_50H | 0x8000);

Emit NOMUTE:

WRITE(0x50,REG_50H);

Receive alarm tone (TONE2)

WRITE(0x47,(REG_47H & 0xE0FF) | 0x1500);

Attachment: AFOUT sidetone control register

Register	Address(HEX)	Description
afout_mode[4:0]	REG_47H[12:8]	0x10 = MUTE 0x11 = RX AFOUT 0x18 = BEEP/TX Side ToneÿCTCSS/CDCSS is not includeÿ

		0x15 = RX ALARM TONE

Attachment: TONE1, TONE2 setting register

Register	Address(HEX)	Description
tone1_gen	REG_70H[15]	Enable TONE1
tone1_gain[6:0]	REG_70H[14:8]	TONE1 tuning gain
tone2_gen	REG_70H[7]	Enable TONE2
tone2_gain[6:0]	REG_70H[6:0]	TONE2 tuning gain
tone1_freq[15:0]	REG_71H[15:0]	TONE1 frequency· control word =freq(kHz)* 2^26/6500
tone2_freq[15:0]	REG_72H[15:0]	TONE2 frequency· control word =freq(kHz)* 2^26/6500

Sub audio settings

Sub audio settings

Turn off subaudio:

```
WRITE(0x51,0x0000);
```

To set up CTCSS transmit and receive:

```
WRITE(0x51,0x90C5); //[6:0] Gain

WRITE(0x07,0x0811); //100Hz

WRITE(0x07,0x0470 | 1<<13); //55Hz tail
```

Set up CDCSS transmit and receive:

```
WRITE(0x51,0x80B0); //[6:0] Gain

WRITE(0x07,0x0AD7); //134.4Hz DCS rate

WRITE(0x07,0x0AD7 | 1<<13); //134.4Hz CTCSS tail

WRITE(0x08,0x0813); //set '023' low 12-bit

WRITE(0x08,0x0763 | 1<<15); //set '023' high 12-bit
```

Attachment: Sub-audio setting register

Register	Address(HEX)	Description
subau_en	REG_51H[15]	1=Enable CTCSS/CDCSS
dcs_invert	REG_51H[13]	1=Transmit negative CDCSS code 0=Transmit positive CDCSS code
ctc_dcs_sel	REG_51H[12]	CTCSS/CDCSS mode select, 1=CTCSS, 0=CDCSS
dcs_24b_mode	REG_51H[11]	24/23bit CDCSS select,

Sub audio settings

		1=24bit, 0=23bit
subau_gain[6:0]	REG_51H[6:0]	Gain ÿ = (1+subau_gain[4:0]/32)<<subau_gain[6:5] [6:5]: CTCSS/CDCSS coarse tuning gain [4:0]: CTCSS/CDCSS fine tuning gain
ctc_freq[15:0]	REG_07H[15:0]	[13]=1 for CTCSS1/CDCSS [13]=0 for CTCSS0/CDCSS [12:0]: CTCSS frequency control word = freq(Hz)*2^27/6500000 [15]=1 for CDCSS
dcs_code[15:0]	REG_08H[15:0]	high 12bit [15]=0 for CDCSS low 12bit [11:0] CDCSS 23/24bit code
~~subau_rx_dcc_bw[2:0]~~	~~REG_B6H[12:10]~~	~~CTC/DCS Rx HPF Bw~~ ~~000=bypass;001=60Hz;010=30Hz;011=15Hz;~~ ~~100=8Hz;101=4Hz;110=2Hz;111=1Hz~~
subau_tx_atten_gain[1:0]	REG_B6H[9:8]	CTC/DCS Tx Atten Gain ÿ, -6dB/step
subau_rx_gain1[1:0]	REG_B6H[7:6]	CTC/DCS Rx Gain1 ÿ, -6dB/step
subau_rx_gain2[1:0]	REG_B6H[5:4]	CTC/DCS Rx Gain2 ÿ, -6dB/step
subau_rx_gain3[3:0]	REG_B6H[3:0]	CTC/DCS Rx Gain3 ÿ, -6dB/step

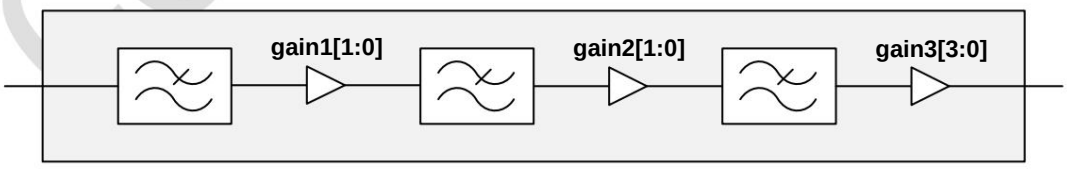
ÿÿ Sub-audio transmit gain can be adjusted in addition to subau_gain[6:0] and subau_tx_atten_gain[1:0]. Another thing to note

Yes, the sub-audio transmission modulation frequency offset is associated with the REG_40H register, so you must first determine the value of REG_40H (that is, the maximum modulation) and then adjust it

subau_gain[6:0] plus subau_tx_atten_gain[1:0].

ÿ The sub-audio receiving gain is subau_rx_gain1[1:0], subau_rx_gain2[1:0], subau_rx_gain3[3:0] in order. if needed

Adjustments should be prioritized from the first level.



254Hz Low Pass Filer

Sub audio settings

Attachment: CTCSS/CDCSS tail sound generation register

Register	Address(HEX)	Description
subau_tail_gen	REG_52H[15]	sub-audio tail generate 0=normal, 1=tail generate
ctc_tail_offs[1:0]	REG_52H[14:13]	CTCSS tail mode and phase change select, 00=No phase shiftÿgenerate CTCSS1ÿ 01=CTCSS0 120°phase shift, 10= CTCSS0 180°phase shift 11= CTCSS0 240°phase shift CTCSS Phase Decode is not supported.

When transmitting CTCSS, set subau_tail_gen=1, if ctc_tail_offs=0, CTCSS1 will be automatically transmitted as tail tone; if

If ctc_tail_offs is other value, it will automatically transmit CTCSS0 with corresponding phase shift as tail tone.

When transmitting CDCSS, after setting subau_tail_gen=1, CTCSS1 is automatically transmitted as tail tone.

Attachment: Sub-audio detection register

Register	Address(HEX)	Description
ctc_th_mode	REG_52H[12]	ctcss detect threshold mode, 1=~0.1%; 0=0.1 Hz
ctc_th_in[5:0]	REG_52H[11:6]	CTCSS found detect threshold
ctc_th_out[5:0]	REG_52H[5:0]	CTCSS lost detect threshold
dcs_detect[1:0]	REG_60H[11:10]	[0]:CDCSS positive code received [1]:CDCSS negative code received (Read Only)
ctc_detect[1:0]	REG_60H[1:0]	[1]:CTCSS1 received [0]:CTCSS0 received (Read Only)

There are two modes for CTCSS detection threshold, 1 is frequency percentage (unit is ~0.1%) mode, 0 is frequency (unit is 0.1Hz) mode (default).

//When mode=0, the unit corresponding to ctc_th_in/ ctc_th_out is about 0.1Hz

Sub audio settings

//When mode=1, the unit corresponding to ctc_th_in/ ctc_th_out is about 0.1%

//When mode=0, if ctc_th_in/ ctc_th_out=15, then the corresponding threshold is about 15*0.1=1.5hz

//mode=1, if ctc_th_in/ ctc_th_out=10, then when the solution is 67hz, the corresponding threshold is 10*67*0.1%=0.67hz

The sub-audio can be detected by reading the values of dcs_detect and ctc_detect regularly, or it can be obtained by interrupt.

Voice control (VOX), transmit timeout (TOT) settings

The judgment of voice control (VOX) adopts a double threshold algorithm, and its judgment logic is as follows:

```
if (vox_amp > vox_amp_th_in)
        VOX = 1;
else
if (vox_amp < vox_amp_th_out) && delay_out
        VOX = 0;
```

Attachment: VOX related registers

Register	Address(HEX)	Description
vox_en	REG_04H[2]	Enable VOX detection
vox_delay[3:0]	REG_7AH[15:12]	VOX=0 delay, *128ms
vox_rssi_th [5:0]	REG_46H[15:10]	RSSI threshold (2dB/step) for VOX. VOX works only when RSSI is lower than this threshold.
vox_amp_th_in[9:0]	REG_46H[9:0]	voice amp threshold for VOX=1 detect voice
vox_amp_th_out[9:0]	REG_79H[9:0]	amp threshold for VOX=0 detect voice amp
vox_amp[9:0]	REG_64H[9:0]	out. (Read Only)
vox_out	REG_0CH[2]	VOX result output. (Read Only)
to_and	REG_04H[6]	Enable TOT detection
tot_timer[7:0]	REG_31H[7:0]	tot timer, *327ms

Squelch (SQ) settings and RSSI, NOISE and SNR

The squelch judgment adopts the double threshold algorithm of rssi_sq and noise_sq, and its judgment logic is as follows:

```
if (rssi_sq > rssi_sq_th_in && noise_sq < noise_sq_th_in)
        SQ = 1;
else
if (rssi_sq < rssi_sq_th_out || noise_sq > noise_sq_th_out)
        SQ = 0;
```

Different squelch levels are distinguished by the threshold of rssi_sq. The threshold of noise_sq can be set to a fixed value, so that the squelch table is relatively simple.

Tip: Different schemes may have external LNAs with different gains. In order to normalize RSSI, the ext_lna_gain register is set to compensate.

If the external LNA gain is 10dB, then you need to set ext_lna_gain=10.

Attachment: SQ related registers

Register	Address(HEX)	Description
rssi_sq_th_in[7:0]	REG_78H[15:8]	RSSI threshold for SQ=1, 0.5dB/step
rssi_sq_th_out[7:0]	REG_78H[7:0]	RSSI threshold for SQ=0, 0.5dB/step
noise_sq_th_out[6:0]	REG_4FH[14:8]	noise threshold for SQ=0
noise_sq_th_in[6:0]	REG_4FH[6:0]	noise threshold for SQ=1
ext_lna_gain[4:0]	**REG_2CH[4:0]**	**External LNA gain RSSI, 1dB/step**
rssi_sq[8:0]	REG_67H[8:0]	0.5dB/step, RSSI (dB) = rssi_sq/2 – 160. (Read Only)
snr_out[7:0]	REG_61H[15:8]	SNR indicator, dB/step. (Read Only)
rssi_rel[7:0]	REG_65H[15:8]	RSSI relative, dB/step. (Read Only)
noise_sq[7:0]	REG_65H[7:0]	NOISE indicator, dB/step. (Read Only)
sq_out	REG_0CH[1]	SQ result output. (Read Only)
weak_rssi	REG_0CH[7]	1=signal is too weak. (Read Only)

SOFT MUTE settings

SOFT MUTE settings

Attachment: SOFT MUTE register

Register	Address(HEX)	Description
soft_mute_en	REG_90H[12]	1=Enable Soft Mute
soft_mute_atten[1:0]	REG_90H[9:8]	Soft Mute Atten Level 00=-16dB; 01=-12dB; 10=-8dB; 11=-4dB
soft_mute_rate[1:0]	REG_90H[7:6]	Soft Mute Rate (SNR/GAIN) 00=1/4; 01=1/2; 10=1; 11=2
snr_th_for_sm[5:0]	REG_90H[5:0]	SNR Threshold for Soft Mute. If SNR little than this value, Soft Mute begin.
snr_out[7:0]	REG_61H[15:8]	SNR indicator, dB/step. (Read Only)

When snr< snr_th_for_sm && weak_rssi && soft_mute_en, the soft_mute function is enabled inside the chip. This function has

It is beneficial to reduce the received audio noise floor under weak signals.

AFC settings

AFC settings

If the temperature compensated crystal oscillator TCXO is used, the frequency is relatively accurate, and the AFC function can be turned off (set afc_disable=1); if the crystal is used,

There is still a certain deviation after frequency calibration, you can turn on AFC (set afc_disable=0) and set a suitable afc_range.

Attachment: AFC register

Register	Address(HEX)	Description
afc_range[2:0]	RegW_73H[13:11]	AFC Range: 000: ~=2.2 kHz 001: ~=1.5 kHz 010: ~=1.1 kHz 011: ~=750 Hz 100: ~=550 Hz 101: ~=375 Hz 110: ~=275 Hz 111: ~=188 Hz if wb, afc range*=2
afc_disable	RegW_73H[4]	1=disable AFC
afc_rail	RegW_0CH[8]	0 = AFC not railed 1 = AFC railed (Read Only)

GPIO, interrupt settings

Attachment: INTERRUPT related registers

Register	Address(HEX)	Description
int_out	REG_0CH[0]	Interrupt output
irq_mask[15:0]	REG_3FH[15:0]	[15]: Enable FSK Tx finished interrupt [14]: Enable FSK FIFO almost empty interrupt [13]: Enable FSK Rx succeed interrupt [12]: Enable FSK FIFO almost full interrupt [11]: Enable FSK Header received interrupt [10]: Enable FSK SyncP succeed interrupt [9]: Enable FSK SyncN succeed interrupt [8]: Enable DTMF/SELCALL code received interrupt [7]: Enable PLL lock lost interrupt [6]: Enable CDCSS receive/lost interrupt [5]: Enable CTCSS receive/lost interrupt [4:3]: reserved [2]: Enable TOT time out interrupt [1]: Enable VOX receive/lost interrupt [0]: Enable SQ receive/lost interrupt
irq_vector[15:0]	REG_02H[15:12]	dtmf/selcall code index[3:0]
	REG_02H[11:10]	10=CDCSS positive code receive 11=CDCSS negative code receive 00=CDCSS positive code lost 01=CDCSS negative code lost
	REG_02H[9:8]	reserved
	REG_02H[7:6]	10=CTCSS0 receive interrupt 00=CTCSS0 lost interrupt 11=CTCSS1 receive interrupt 01=CTCSS1 lost interrupt
	REG_02H[5]	0=VOX lost 1=VOX receive
	REG_02H[4]	0=SQ lost

		1=SQ receive
	REG_02H[3:0]	15=FSK Tx Finished 14=FSK FIFO(almost empty) need to WRITE 13=FSK Rx Succeed 12=FSK FIFO(almost full) need to read 11=FSK Header received interrupt 10=FSK SyncP found interrupt 9=FSK SyncN found interrupt 8=DTMF/SELCALL receive interrupt 7=PLL lock lost interrupt 6=CDCSS receive/lost interrupt 5=CTCSS receive/lost interrupt 4,3=reserved 2=TOT time out interrupt 1=VOX receive/lost interrupt 0=SQ receive/lost interrupt

The interrupt can be output by any GPIO port of the chip, and the interrupt can be cleared by writing any value to the 02H register, such as

```
WRITE (0x02,0x0000); //clear interrupt
```

Tip: Interrupt is active at high level. When getting an interrupt, you must clear the interrupt before reading the interrupt vector table. Read the vector table, first according to irq_vector[3:0] to judge the interrupt source, and then find the specific interrupt event in the interrupt source. The processing is as follows:

```
int rdata = 0;
if (PIN_INT) { // MCU reads the chip's interrupt output PIN_INT
        WRITE(0x02,0x0000);
        Read(0x02,rdata);
        switch (rdata & 0xF) {
                case 0: SQ = (rdata >> 4) & 1; break;
                case 1: VOX = (rdata >> 5) & 1; break;
                case 2: break; //TOT emission timeout
```

```
                case 3: break; //reserved
                case 5:
                        switch ((rdata >> 6) & 1) {
                                case 0: CTC0 = (rdata >> 7) & 1; break;
                                case 1: CTC1 = (rdata >> 7) & 1; break;
                        }
                case 6:
                        switch ((rdata >>10) & 1) {
                                case 0: DCSP = (rdata >> 11) & 1; break; //DCS ÿÿ
                                case 1: DCSN = (rdata >> 11) & 1; break; //DCS ÿÿ
                        }
                case 7: break; //The PLL is out of lock and the PLL can be retriggered by first in the receive or transmit state
REG_03H[15]=0 and then REG_03H[15]=1
                case 8: CODE = (rdata >>12) & 0xf; break; //DTMF or SELCALL CODE
                case 9: break; //FSK SyncN Found
                case 10: break; //FSK SyncP Found
                case 11: break; //FSK Header Found
                case 12: break; //Read Data from FIFO
                case 13: break; //FSK Rx Succeed
                case 14: break; //WRITE Data to FIFO
                case 15: break; //FSK Tx Finished
        } //end switch
} //end if
```

GPIO, interrupt settings

Attachment: GPIO register settings

Register	Address(HEX)	Description
gpio_oen_b[7:0]	REG_25H[15:8]	gpioX output enable, low active, (X=0..7)
gpio_out_val[7:0]	REG_25H[7:0]	gpioX output value when gpioX_out_sel=0, (X=0..7)
gpio7_out_sel[3:0]	REG_27H[15:12]	gpio7 ouput select. 0=gpio_out_val[X], (X=0..7) 1=INT 2=SQ 3=VOX 4=subau_cmp (CTCSS/CDCSS compare result output) 5=CTCSS/CDCSS code output for software decode 6=SDO for 4-wire mode 7:11=reserved ~~12=I2S/MCBSP DOUT output~~ ~~13=I2S/MCBSP DFS output~~ ~~14=I2S/MCBSP DCLK output~~ ~~15=XTAL CLK div2 output~~
gpio6_out_sel[3:0]	REG_27H[11:8]	Description is the same as gpio7_out_sel[3:0]
gpio5_out_sel[3:0]	REG_27H[7:4]	Description is the same as gpio7_out_sel[3:0]
gpio4_out_sel[3:0]	REG_27H[3:0]	Description is the same as gpio7_out_sel[3:0]
gpio3_out_sel[3:0]	REG_26H[15:12]	Description is the same as gpio7_out_sel[3:0]
gpio2_out_sel[3:0]	REG_26H[11:8]	Description is the same as gpio7_out_sel[3:0]
gpio1_out_sel[3:0]	REG_26H[7:4]	Description is the same as gpio7_out_sel[3:0]
gpio0_out_sel[3:0]	REG_26H[3:0]	Description is the same as gpio7_out_sel[3:0]
gpio_in_val[7:0]	REG_28H[15:8]	gpioX input value, (X=0..7). (Read Only)

Tips: GPIO0~GPIO7 are all pull-down.

For example: use GPIO4 as SQ output, use GPIO5 as VOX output, you need to set

gpio_oen_b[4]=0, gpio4_out_sel[3:0]=2;

gpio_oen_b[5]=0, gpio5_out_sel[3:0]=3;

For example: use GPIO4 as variable output, you need to set

gpio_oen_b[4]=0, gpio4_out_sel[3:0]=0, the output value of GPIO4 is the value corresponding to gpio_out_val[4].

For example: use GPIO5 as input, you need to set

gpio_oen_b[5]=1, the read value of gpio_in_val[5] is the input value of GPIO5.

Crystal/Crystal Clock Frequency Setting

Crystal/Crystal Clock Frequency Setting

XTAL is the clock frequency (unit Hz), then N=XTAL/10*2, the clock frequency of the crystal/crystal oscillator is set as follows:

WRITE(0x3B,N);

WRITE(0x3C,(N>>16)<<8);

Except for the clock frequency, if the clock frequency is in the range of 12M~13M, you need to set:

WRITE(0x22,0x9E14);

WRITE(0x41,0x81C1);

WRITE(0x3D,0x4EC5);

If the clock frequency is in the range of 18M~20M, you need to set:

WRITE(0x22,0x5E14);

WRITE(0x41,0x81C2);

WRITE(0x3D,0x3483);

If the clock frequency is in the range of 24M~26M, you need to set: (default 26MHz, you can not set it)

WRITE(0x22,0x3E14);

WRITE(0x41,0x81C4);

WRITE(0x3D,0x2762);

Register Summary

Register Summary

Address	R/W	Default		Description	
REG_00H	R	0x6818	chip_id[15:0]	[15:0]	Chip identification code
REG_01H	R	revision_id[15:0]		[15:0]	Revision identification code
REG_02H	R		irq_vector[15:0]	[15:12]	dtmf/selcall code index[3:0]
				[11:10]	10=CDCSS positive code receive 11=CDCSS negative code receive 00=CDCSS positive code lost 01=CDCSS negative code lost
				[9:8]	reserved
				[7:6]	10=CTCSS0 receive interrupt 00=CTCSS0 lost interrupt 11=CTCSS1 receive interrupt 01=CTCSS1 lost interrupt
				[5]	0=VOX lost 1=VOX receive
				[4]	0=SQ lost 1=SQ receive
				[3:0]	15=FSK Tx Finished 14=FSK FIFO(almost empty) need to WRITE 13=FSK Rx Succeed 12=FSK FIFO(almost full) need to read 11=FSK Header received interrupt 10=FSK SyncP found interrupt 9=FSK SyncN found interrupt 8=PLL lock lost interrupt 6=CDCSS receive/lost interrupt

Register Summary

					5=CTCSS receive/lost interrupt 4=DTMF/SELCALL receive interrupt 3=reserved 2=TOT time out interrupt 1=VOX receive/lost interrupt 0=SQ receive/lost interrupt
REG_03H R/W 0x0000 vco_cal_en				[15]	0->1=Enable PLL Calibration
			pabias_in	[14]	1=Enable PABIAS
			rxlink_en[3:0]	[13:10]	1111=Enable Rx Link, (LNA,MIXER,FILTER,ADC)
			aerror_en	[9]	1=Enable AFOUT DAC
			pll_en[4:0]	[8:4]	11111=Enable PLL, VCO
			padrv_en	[3]	1=Enable PADRV
			micin_en	[2]	1=Enable MICIN ADC
			thon	[1]	1=Enable Tx DSP
			rxon	[0]	1=Enable Rx DSP
REG_04H R/W 0x0000 reserved				[15:10] Reserved	
			adc_cal_en	[9]	0->1=Enable ADC calibration
			offset_cal_in	[8]	0->1=Enable offset calibration
			pkd_disable	[7]	1=Disable Peak Detection
			to_and	[6]	1=Enable TOT detection
			scramb_en	[5]	1=Enable SCRAMBLER
			fsk_en	[4]	1=Enable FSK mode
			~~amdem_en~~	~~[3]~~	~~1=Enable AM Demodulation~~
			vox_en	[2]	1=Enable VOX detection
			dtmf_en	[1]	1=Enable DTMF/SELCALL
			~~mcbsp_en~~	~~[0]~~	~~1=Enable MCBSP/I2S module~~
REG_07H W			ctc_freq[15:0]	[15:0]	[13]=1 for CTCSS1/CDCSS [13]=0 for CTCSS0/CDCSS [12:0]: CTCSS frequency control word = freq(Hz)*2^27/6500000

Register Summary

REG_08H W			dcs_code[15:0]	[15:0]	[15]=1 for CDCSS high 12bit [15]=0 for CDCSS low 12bit [11:0] CDCSS 23/24bit code
REG_09H W			dtmf_coef [15:0]	[15:0]	[15:12]=coefficient index [7:0]=coefficients for DTMF/SELCALL detection
REG_0BH	R		reserved	[15:13] Reserved	
			dtmf_code_ready	[12]	1=DTMF/ SELCALL symbol received ready.
			dtmf_code[3:0]	[11:8] DTMF/ SELCALL symbol received.	
			reserved	[7:0]	Reserved
REG_0CH	R		reserved	[15:10] Reserved	
			pll_lock	[9]	1=pll lock 0=pll lock lost
			afc_rail	[8]	0 = AFC not railed 1 = AFC railed
			weak_rssi	[7]	1=signal is too weak.
			reserved	[6:4]	Reserved
			until_out	[3]	TOT output
			vox_out	[2]	VOX result output.
			sq_out	[1]	SQ result output.
			int_out	[0]	Interrupt output
REG_18H R/W 0x4125 mic_pga_gain[4:0]				[15:11] 10000 max 01000(default) 00100 00010 00001 min	
			reserved	[10:0]	Reserved
REG_19H R/W 0x900F reserved				[15:4]	Reserved
			pabias_out[3:0]	[3:0]	PAbias output 1.3~2.8V, 100mv/ step 0000=1.3V ... 1111=2.8V

Register Summary

REG_23H R/W 0x8F8F reserved		[15:4]	Reserved
	dac_vgain[3:0]	[3:0]	DAC maximum output level
~~REG_24H R/W 0x603F reserved~~	~~———~~	~~[15:12] Reserved~~	
	~~mcbsp_clks_in_sel[2:0]~~	~~[11:9]~~	~~Select GPIOx as I2S/MCBSP DCLKS input (x=0..7)~~
	~~mcbsp_clk_in_sel[2:0]~~	~~[8:6]~~	~~Select GPIOx as I2S/MCBSP DCLK input (x=0..7)~~
	~~mcbsp_fs_in_sel[2:0]~~	~~[5:3]~~	~~Select GPIOx as I2S/MCBSP DFS input (x=0..7)~~
	~~mcbsp_dr_in_sel[2:0]~~	~~[2:0]~~	~~Select GPIOx as I2S/MCBSP DIN input (x=0..7)~~
REG_25H R/W 0x0000 gpio_oen_b[7:0]		[15:8] gpioX output enable, low active, (X=0..7)	
	gpio_out_val[7:0]	[7:0]	gpioX output value when gpioX_out_sel=0, (X=0..7)
REG_26H R/W 0x0005 gpio3_out_sel[3:0]		[15:12] Gpio3 output select. 0=gpio_out_val[X], (X=0..7) 1=INT 2=SQ 3=VOX 4=subau_cmp (CTCSS/CDCSS compare result output) 5=CTCSS/CDCSS code output for software decode 6=SDO for 4-wire mode 7:11=reserved **12=I2S/MCBSP DOUT output** **13=I2S/MCBSP DFS output** **14=I2S/MCBSP DCLK output** **15=XTAL CLK div2 output**	
	gpio2_out_sel[3:0]	[11:8]	Description is the same as gpio3_out_sel[3:0]
	gpio1_out_sel[3:0]	[7:4]	Description is the same as gpio3_out_sel[3:0]
	gpio0_out_sel[3:0]	[3:0]	Description is the same as

Register Summary

					gpio3_out_sel[3:0]
REG_27H	R/W	0x3200	gpio7_out_sel[3:0]	[15:12]	Description is the same as gpio3_out_sel[3:0]
			gpio6_out_sel[3:0]	[11:8]	Description is the same as gpio3_out_sel[3:0]
			gpio5_out_sel[3:0]	[7:4]	Description is the same as gpio3_out_sel[3:0]
			gpio4_out_sel[3:0]	[3:0]	Description is the same as gpio3_out_sel[3:0] gpioX input
REG_28H	R/W	0x003F	gpio_in_val[7:0]	[15:8]	value, (X=0..7). (Read Only)
			reserved	[7:6]	Reserved
			padrv_gain[2:0]	[5:3]	PA output gain
			pa_gain_vreg[2:0]	[2:0]	PA output gain
REG_38H	R/W	0x3A98	freq [15:0]	[15:0]	Frequency
REG_39H	R/W	0x0271	freq [31:16]	[15:0]	=(REG_39H<<16 + REG_38H)*10 Hz
REG_3BH	R/W	0x3D62	xtal_freq [15:0]	[15:0]	XTAL Frequency
REG_3CH	R/W	0x1000	xtal_freq [23:16]	[15:8]	=((REG_3CH>>8)<<16 + REG_3BH)*5 Hz
			reserved	[7:0]	Reserved.
REG_3FH	R/W	0x0000	irq_mask[15:0]	[15:0]	[15]: Enable FSK Tx finished interrupt [14]: Enable FSK FIFO almost empty interrupt [13]: Enable FSK Rx succeed interrupt [12]: Enable FSK FIFO almost full interrupt [11]: Enable FSK Header received interrupt [10]: Enable FSK SyncP succeed interrupt [9]: Enable FSK SyncN succeed interrupt [8]: Enable DTMF/SELCALL

Register Summary

					code received interrupt [7]: Enable PLL lock lost interrupt [6]: Enable CDCSS receive/lost interrupt [5]: Enable CTCSS receive/lost interrupt [4:3]: reserved [2]: Enable TOT time out interrupt [1]: Enable VOX receive/lost interrupt [0]: Enable SQ receive/lost interrupt
REG_40H	R/W	0x3808	reserved		[15:13] Reserved
			dev_en	[12]	Enable FM deviation
			dev_sh[3:0]	[11:8]	FM deviation coarse tuningÿ 0000=max, 1111=min
			dev_lvl[7:0]	[7:0]	FM deviation fine tuningÿ 0000=min, 1111=max, GAIN=(256+dev_lvl[7:0])>>dev_sh[3:0]
REG_43H	R/W	0x6009	reserved	[15]	Reserved
			firlpf_bw[2:0]	[14:12]	RF filter bandwidth (Apass=0.1dB) 000 = 1.7 kHz 001 = 2 kHz 010 = 2.5 kHz 011 = 3 kHz 100 = 3.75 kHz 101 = 4 kHz 110 = 4.25 kHz 111 = 4.5 kHz if wb=1, firlpf_bw *=2;
			firlpf_bw_for_weak[2:0]	[11:9]	RF filter bandwidth when signal is weak.

Register Summary

			audio_lpf2_tx_sel[2:0]	[8:6]	Audio LPF bandwidth (Apass=1dB) for Tx 100 = 4.5 kHz 101 = 4.25 kHz 110 = 4 kHz 111 = 3.75 kHz 000 = 3 kHz 001 = 2.5 kHz 010 = 2 kHz 011 = 1.7kHz
			reserved	[5:2]	Reserved
			firlpf_gain[1:0]	[1:0]	Gain after FIR LPF 00=0dB; 01=6dB; 10=12dB; 11=18dB
REG_44H R/W 0x43EC reserved				[15:13] Reserved	
			fmdem_gain[1:0]	[12:11] Gain after FM Demodulation 00=0dB; 01=6dB; 10=12dB; 11=18dB	
			reserved	[10:0]	Reserved
REG_46H R/W 0x8050 vox_rssi_th[15:10]				[15:10] RSSI threshold (2dB/step) for VOX. VOX works only when RSSI is lower than this threshold.	
			vox_amp_th_in[9:0]	[9:0]	voice amp threshold for VOX=1 detect
REG_47H R/W 0x6140 reserved				[15]	Reserved
			reserved	[14]	Reserved
			afout_invert	[13]	1 = invert AFOUT
			afout_mode[4:0]	[12:8] 0x10 = MUTE 0x11 = RX AFOUT 0x18 = BEEP/TX Side Tone ÿCTCSS/CDCSS is not includeÿ 0x15 = RX ALARM TONE	
			reserved	[7:4]	Reserved

Register Summary

			dig_gain_tx[3:0]	[3:0]	Digital gain after MIC ADC, 1dB/ step
REG_4FH R/W 0x7E34 reserved				[15]	Reserved
			noise_sq_th_out[6:0]	[14:8]	noise threshold for SQ=0
			reserved	[7]	Reserved
			noise_sq_th_in[6:0]	[6:0]	noise threshold for SQ=1
REG_50H R/W 0x033C audio_tx_mute				[15]	1=Audio Tx Mute
			reserved	[14:11] Reserved	
			audio_tx_limit_bypass	[10]	1=Audio Tx Limit bypass
			audio_tx_limit[9:0]	[9:0]	Audio Tx Limit Value
REG_51H R/W 0x1050 subau_en				[15]	1=Enable CTCSS/CDCSS
			reserved	[14]	Reserved
			dcs_invert	[13]	1=Transmit negative CDCSS code 0=Transmit positive CDCSS code
			ctc_dcs_sel	[12]	CTCSS/CDCSS mode select, 1=CTCSS, 0=CDCSS
			dcs_24b_mode	[11]	24/23bit CDCSS select, 1=24bit, 0=23bit
			reserved	[10:7]	Reserved
			subau_gain[6:0]	[6:0]	[6:5]: CTCSS/CDCSS coarse tuning gain [4:0]: CTCSS/CDCSS fine tuning gain
REG_52H R/W 0x028F subau_tail_gen				[15]	sub-audio tail generate 0=normal, 1=tail generate
			ctc_tail_offs[1:0]	[14:13] CTCSS tail mode and phase change select, 00=No phase shiftÿgenerate CTCSS1ÿ 01=CTCSS0 120°phase shift, 10= CTCSS0 180°phase shift 11= CTCSS0 240°phase shift CTCSS Phase Decode is not	

Register Summary

					supported.
			ctc_th_mode	[12]	ctcss detect threshold mode, 1=~0.1%; 0=0.1 Hz
			ctc_th_in[5:0]	[11:6] CTCSS found detect threshold	
			ctc_th_out[5:0]	[5:0]	CTCSS lost detect threshold
REG_53H R/W 0x97DD		audio_rx_gain_sh [2:0]		[15:13] 111=max,000=min 0~-42dB, 6dB/step, digital gain	
			audio_tx_gain_sh [2:0]	[12:10] 111=max,000=min 0~-42dB, 6dB/step, digital gain	
			audio_rx_gain[4:0]	[9:5]	0=mute,31=max,1dB/step,
			audio_tx_gain[4:0]	[4:0]	digital gain 0=mute,31=max,1dB/ step, digital gain
REG_54H R/W 0x8517		scramb_freq[15:0]		[15:0] SCRAMBLE/FSK frequency control word =3.3(KHz)*2^26/6500 - The scrambler inversion mixing frequency should be kept between 2.6kHz and 3.5kHz	
REG_56H R/W 0x1021		fsk_crc_polyn[15:0]		[15:0] CRC polynomial coefficient, CCIT-16 is default	
REG_59H R/W 0x8078	fsk_tx_fifo_clear	fsk_rx_fifo_clear		[15]	Clear TX FIFO, 1=clear
			fsk_scramble_en	[14]	Clear RX FIFO, 1=clear
			fsk_rx_en fsk_tx_en	[13]	1=Enable FSK Scramble
			fsk_rx_data_inv	[12]	1=Enable FSK TX
			fsk_tx_data_inv	[11]	1=Enable FSK RX
				[10]	1=invert FSK data before TX
				[9]	1=invert FSK data after RX
			reserved	[8]	Reserved
			fsk_prmb_size[3:0]	[7:4]	FSK preamble length select, length=(fsk_prmb_size[3:0]+1) bytes
			fsk_sync_size	[3]	FSK sync length select, 1=4 bytes

Register Summary

					(fsk_sync_byte0,1,2,3) 0=2 bytes (fsk_sync_byte0,1)
			reserved	[2:0]	Reserved
REG_5AH	R/W	0x85CF	fsk_sync_byte0[7:0]	[15:8]	FSK sync byte0
			fsk_sync_byte1[7:0]	[7:0]	FSK sync byte1
REG_5BH	R/W	0xAB45	fsk_sync_byte2[7:0]	[15:8]	FSK sync byte2
			fsk_sync_byte3[7:0]	[7:0]	FSK sync byte3
REG_5CH	R/W	0x56F9	reserved	[15:8]	Reserved
			fsk_tx_type_byte[7:0]	[7:0]	FSK type byte to be transmitted. "0x01"=FSK Format2 "0x21"=FSK Format1 without CRC "0x61"=FSK Format1 with CRC
REG_5DH	R/W	0x3FCC	fsk_length[7:0]	[15:8]	FSK data length when use FSK format1 length=(fsk_length[7:0]+1) bytes
			reserved	[7:0]	Reserved
REG_5EH	R/W	0x0004	reserved	[15:12]	Reserved
			fsk_tx_fifo_ae_th [5: 0]	[11:6]	FSK TX FIFO almost empty threshold(unit is 1word=2 bytes)
			fsk_rx_fifo_af_th[5:0]	[5:0]	FSK RX FIFO almost full threshold(unit is 1word=2 bytes)
REG_5FH	R/W		fsk_data[15:0]	[15:0]	TX/RX FIFO Data
REG_60H	R		reserved	[15:12]	Reserved
			dcs_detect[1:0]	[11:10]	[0]:CDCSS positive code received [1]:CDCSS negative code received
			reserved	[9:2]	Reserved
			ctc_detect[1:0]	[1:0]	[1]:CTCSS1 received

Register Summary

					[0]:CTCSS0 received
REG_61H	R		snr_out[7:0]	[15:8]	SNR indicator, dB/step.
			reserved	[7:0]	Reserved
REG_62H	R		agc_rssi[7:0]	[15:8]	RSSI after DCC, 1dB/step.
			lna_peak_rssi[7:0]	[7:0]	RSSI after LNA, 1dB/step.
REG_64H	R		reserved	[15:10]	Reserved
			vox_amp[9:0]	[9:0]	voice amp out.
REG_65H	R		rssi_rel[7:0]	[15:8]	RSSI relative, dB/step
			noise_sq[7:0]	[7:0]	NOISE indicator, dB/step.
REG_67H	R		reserved	[15:9]	Reserved
			rssi_sq[8:0]	[8:0]	0.5dB/step, RSSI (dB) = rssi_sq/2 – 160.
REG_70H	R/W	0x7070	tone1_gen	[15]	Enable TONE1
			tone1_gain[6:0]	[14:8]	TONE1 tuning gain
			tone2_gen	[7]	Enable TONE2
			tone2_gain[6:0]	[6:0]	TONE2 tuning gain
REG_71H	R/W	0x2854	tone1_freq[15:0]	[15:0]	TONE1 frequency· control word =freq(kHz)* 2^26/6500
REG_72H	R/W	0x3065	tone2_freq[15:0]	[15:0]	TONE2 frequency· control word =freq(kHz)* 2^26/6500
REG_73H	R/W	0x568A	reserved	[15:14]	Reserved
			afc_range[2:0]	[13:11]	AFC Range: 000: ~=2.2 kHz 001: ~=1.5 kHz 010: ~=1.1 kHz 011: ~=750 Hz 100: ~=550 Hz 101: ~=375 Hz 110: ~=275 Hz 111: ~=188 Hz if wb, afc range*=2
			reserved	[10:5]	Reserved

Register Summary

			afc_disable	[4]	1=disable AFC
			reserved	[3:0]	Reserved
REG_75H R/W 0x0FB4 resreved				[15]	Reserved
			mcbsp_test	[14]	0=normal; 1=Transmit data equals to REG_78H
			reserved	[13]	Reserved
			mcbsp_fwid[4:0]	[12:8]	Transmit/Receive frame synchronization width.
			mcbsp_flen	[7]	Transmit/Receive frame length. 1=32 bit; 0=16 bit 0=1
			mcbsp_delay	[6]	data delay; 1=2 data delay
			mcbsp_master	[5]	0=slave mode; 1=master mode
			mcbsp_clks_master	[4]	1= mcbsp clk source master mode; 0= mcbsp clk source slave mode
			mcbsp_clks_psel	[3]	mcbsp clk source polarity select.
			mcbsp_clks_enable	[2]	mcbsp clk source enable
			mcbsp_clk_psel	[1]	Transmit/Receive clock polarity bit. It determines the polarity of DCLK as seen on the GPIO pin. 0=Transmit/Receive data is sampled on the rising edge of DCLK. 1=Transmit/Receive data is sampled on the falling edge of DCLK.
			mcbsp_fs_psel	[0]	Transmit/Receive frame synchronization polarity bit. It determines the polarity of DFS as seen on the GPIO pin. 0=Transmit/Receive frame synchronization pulses are

Register Summary

					active high. 1=Transmit/Receive frame synchronization pulses are active low.
REG_76H R/W 0xE307	mcbsp_phase_swap[1:0]			[15:14]	MCBSP/I2S LR data swap for 32 bit mode. [1]: for Tx, [0]: for Rx
		reserved		[13:12]	Reserved
		mcbsp_dsel		[11]	MCBSP/I2S data output select. 1=IQ, 0=FM Demodulated
		reserved		[10:9]	Reserved
		mcbsp_fifo_bypass		[8]	MCBSP/I2S FIFO bypass. FIFO is not needed in master mode, set "1" to bypass FIFO which can reduce time on the path.
		mcbsp_clkgdv[7:0]		[7:0]	MCBSP/I2S generator divide number.
REG_78H R/W 0x44FF	rssi_sq_th_in[7:0]			[15:8]	RSSI threshold for SQ=1, 0.5dB/step
		rssi_sq_th_out[7:0]		[7:0]	RSSI threshold for SQ=0, 0.5dB/step
REG_79H R/W 0x1040	reserved			[15:10]	Reserved
		vox_amp_th_out[9:0]		[9:0]	voice amp threshold for VOX=0 detect
REG_7AH R/W 0x881A	vox_delay[3:0]			[15:12]	VOX=0 delay, *128ms
		reserved		[11:0]	Reserved
REG_7DH R/W 0x4200	reserved			[15:13]	Reserved
		dig_gain_rx[4:0]		[12:8]	Gain after AGC, digital down conversion. 1dB/step
		reserved		[7:0]	Reserved
REG_7EH R/W 0x341E	reserved			[15:8]	Reserved
		dcc_tx_bypass		[7]	1=MICIN DC Filter bypass
		dcc_rx_bypass		[6]	1=Rx DC Filter bypass
		dcc_tx_bw[2:0]		[5:3]	MICIN DC Filter bandwidth(3dB) select 111=15Hz;110=30Hz;101=60H

Register Summary

					z;100=120Hz; 011=240Hz;010=480Hz;
			reserved	[2:0]	Reserved
REG_87H	R		reserved	[15:7]	Reserved
			cmpd_db_out[6:0]	[6:0]	Audio amplitude output, 1dB/ step.
REG_90H	R/W	0x8C20	reserved	[15:13] Reserved	
			soft_mute_en	[12]	1=Enable Soft Mute
			reserved	[11:10] Reserved	
			soft_mute_rate[1:0]	[9:8]	Soft Mute Rate (SNR/GAIN) 00=1/4; 01=1/2; 10=1; 11=2
			soft_mute_atten[1:0]	[7:6]	Soft Mute Atten Level 00=-16dB; 01=-12dB; 10=-8dB; 11=-4dB
			snr_th_for_sm[5:0]	[5:0]	SNR Threshold for Soft Mute. If SNR little than this value, Soft Mute begin.
REG_94H	R/W	0x8F5E	reserved	[15:13] Reserved	
			dtmf_det_th[5:0]	[12:7] DTMF/ SELCALL detect threshold, 1dB/step	
			dtmf_symbol_mode	[6]	DTMF/ SELCALL symbol mode 1=”symbol”+”idle”+”sym bol”+… (DTMF) 0=”symbol”+”symbol”+” symbol”+…(5TONE)
			dtmf_in_sel	[5]	1=select gain out 0=select audio rx out
			dtmf_mode	[4]	Dual/Single Tone Detect mode 1=dual tone 0=single tone
			dtmf_symbol_max[3:0]	[3:0]	SELCALL maximum symbol number 15=symbol “0”~”F” (DTMF) 14=symbol “0”~”E” (THIS)

Register Summary

					...
REG_96H R/W 0x1AE8 reserved				[15:14]	
			audio_rx_gain1[5:0]	[13:8] 0=mute, 63=max, 0.5dB/step	
			cmpd_rx_gain[4:0]	[7:3]	Audio CMPD Gain, 1dB/step
			cmpd_rx_bypass[2:0]	[2:0]	Audio CMPD bypass for RX 110=bypass with gain 001=bypass without gain 000=CMPD ON for RX Others=reserved
REG_97H R/W 0x0AA8 reserved				[15:14] Reserved	
			~~mic_sens_gain[5:0]~~	~~[13:8] MIC sensitivity adjust gain 0=mute, 63=max, 0.5dB/step~~	
			cmpd_tx_gain[4:0]	[7:3]	Audio CMPD Gain, 0.5dB/step
			cmpd_tx_bypass[2:0]	[2:0]	Audio CMPD bypass for TX 110=bypass with gain 001=bypass without gain 000=CMPD ON for TX Others=reserved
REG_98H R/W 0x7A14 reserved				[15]	Reserved
			cmpd_rx_factor[2:0]	[14:12] Expanding Factor. 100=1:3;011=1:2.5;010=1:2;001=1:1.5	
			cmpd_rx_th_high[5:0]	[11:6] Above this amplitude point, audio will be expanded according to the expanding factor. The unit of this threshold is 2dB.	
			cmpd_rx_th_low[5:0]	[5:0]	Under this amplitude point, audio will be attenuated. The unit of this threshold is 2dB.
REG_99H R/W 0x7855 reserved				[15]	Reserved
			cmpd_tx_factor[2:0]	[14:12] Compressing Factor. 111=8:1;110=4:1;100=2:1;000=1:1	
			cmpd_tx_th_high[5:0]	[11:6] Above this amplitude point,	

Register Summary

					audio will be compressed according to the compressing factor. The unit of this threshold is 2dB.
			cmpd_tx_th_low[5:0]	[5:0]	Under this amplitude point, audio will be attenuated. The unit of this threshold is 2dB.
REG_9AH R/W 0x0728 reserved				[15:14] Reserved	
			cmpd_ct_intvl[5:0]	[13:8] Companding Amplitude Detect Interval, 0.64ms/ step Companding Gain Attack Speed.	
			cmpd_atk_step[3:0]	[7:4]	0000=most fast ... 1111=most slow
			cmpd_rls_step[3:0]	[3:0]	Companding Gain Release Speed. 0000=most fast ... 1111=most slow
REG_9BH R/W 0x0004 reserved				[15:10] Reserved	
			audio_hpf_rx_bypass	[9]	Audio HPF 300Hz bypass for RX
			audio_lpf1_rx_bypass	[8]	Audio LPF1 bypass for RX
			audio_lpf2_rx_bypass	[7]	Audio LPF2 bypass for RX
			audio_emph_rx_bypass [6]		DE-EMPHASIS bypass
			reserved	[5:4]	Reserved
			audio_hpf_tx_bypass	[3]	Audio HPF 300Hz bypass for TX
			audio_lpf1_tx_bypass	[2]	Audio LPF1 bypass for TX
			audio_lpf2_tx_bypass	[1]	Audio LPF2 bypass for TX
			audio_emph_tx_bypass [0]		PRE-EMPHASIS bypass
REG_9DH R/W 0x29AD reserved				[15:4]	Reserved
			audio_tx_path_sel[1:0]	[3:2]	01=select pre-emphasis output 10=reserved

Register Summary

					11=select LPF2 output
			reserved	[1:0]	Reserved
REG_B6H	R/W	0x9d08	reserved	[15:13]	Reserved
			~~subau_rx_dcc_bw[2:0]~~	~~[12:10]~~	~~CTC/DCS Rx HPF Bw 000=bypass;001=60Hz;010=30Hz;011=15Hz; 100=8Hz;101=4Hz;110=2Hz;111=1Hz~~
			subau_tx_atten_gain[1:0]	[9:8]	CTC/DCS Tx Atten Gain, -6dB/step
			subau_rx_gain1[1:0]	[7:6]	CTC/DCS Rx Gain1, -6dB/step
			subau_rx_gain2[1:0]	[5:4]	CTC/DCS Rx Gain2, -6dB/step
			subau_rx_gain3[3:0]	[3:0]	CTC/DCS Rx Gain3, -6dB/step