Alango Kalimba HEP
Parameters specification and PSKEY structure

PSKEY version 3, HEP library version: 38+
May 11, 2015

Contents

HEP PSKEY structure logic

HEP library retrieves its parameters from PSKEY which number is sent to DSP by VM (Virtual Machine) during DSP initialization (Kalimba loading) as a parameter of CVC_LOADPARAMS_MSG message. In terminology of HEP, this PSKEY is called “Main HEP PSKEY”.

Main HEP PSKEY content is represented in format described in details in the next section of the present document. Main HEP PSKEY version contained in the PSKEY_VER field indicates PSKEY data format and therefore it must correspond to HEP library version used (for example, PSKEY_VER=1 is for HEP revisions from xxxx to xxxx). HEP library notifies the user about incompatible PSKEY version or wrong Main HEP PSKEY data or incompatible HEP type by reproducing periodical beeps (2 seconds long beep generated each 5 seconds).

Main HEP PSKEY contains packed values of HEP acoustic parameters, flags and addresses (numbers) of additional PSKEYs, namely:

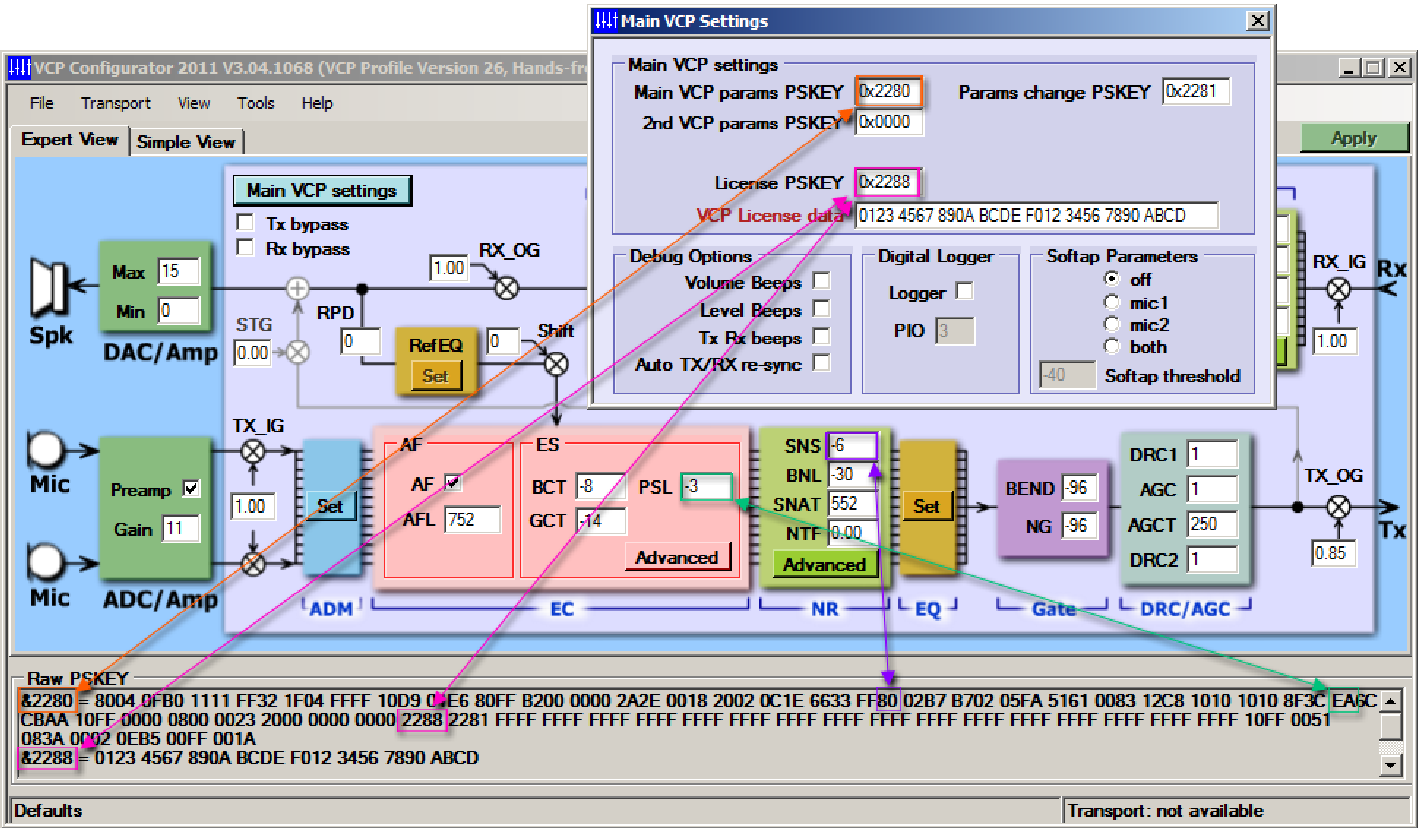
- Secondary HEP PSKEY number (contained inside PSKEY_SEC field of the Main HEP PSKEY)
- Parameter monitoring PSKEY number (contained inside PSKEY_MON field of the Main HEP PSKEY)
- HEP License PSKEY number (contained inside PSKEY_LIC field of the Main HEP PSKEY, optional)

Secondary HEP PSKEY (PSKEY_SEC) contains non-mandatory HEP parameter values. Its content is represented in a format described in details in section “Secondary (optional) HEP PSKEY structure” (below). If PSKEY_SEC is set to 0x0000, then Secondary HEP PSKEY is not read by HEP and parameter values contained in it are set to internal library defaults.

Parameter monitoring PSKEY (PSKEY_MON) consists of one word as follows: RNDVAL

RNDVAL word contains a random number. HEP Configurator writes a new random number into it each time parameters are “applied” (i.e. stored inside Main HEP PSKEY) to indicate HEP library about parameters change. HEP library running inside Kalimba DSP continuously monitors the content of this PSKEY by re-reading it every second. If the value of RNDVAL has changed, HEP reads its parameters from Main HEP PSKEY and performs full algorithm re-initialization. This way HEP library allows changing its parameters from HEP Configurator “on-the-fly” and applying them during active voice call without the need of call reconnection.

Normally, HEP Configurator is used to configure HEP parameters and control HEP PSKEY contents. However, in certain cases manual editing of HEP parameters may be required. Below is a visual example of relation between parameter values in HEP Configurator and their representation and places in PSKEYs:



Exact specification of HEP parameters representation in PSKEYs is provided below.

Primary (main) HEP PSKEY structure

Parameter ID and word number in PSKEY	Physical values range	Location in PSKEY and conversion formula into fixed-point representation	Fixed-point representation range and size
Word 1 - Flags			
AF_FLAG	flag (0 / 1)	Bit 0	0 … 1 (1 bit)
BYPASS_FLAG	flag (0 / 1)	Bit 1	0 … 1 (1 bit)
ADMLIM_FLAG	flag (0 / 1)	Bit 2	0 … 1 (1 bit)
GDNR_FLAG	flag (0 / 1)	Bit 3	0 … 1 (1 bit)
-	-	-	0 … 1 (1 bit)
BEEPS_SAT_FLAG	flag (0 / 1)	Bit 5	0 … 1 (1 bit)
ADMREVMIC_FLAG	flag (0 / 1)	Bit 6	0 … 1 (1 bit)
LOG_FLAG	flag (0 / 1)	Bit 7	0 … 1 (1 bit)
NR_LFNF_FLAG	flag (0 / 1)	Bit 8	0 … 1 (1 bit)
REINIT_BEEPS_DISABLE	flag (0 / 1)	Bit 9	0 … 1 (1 bit)
AFAH	flag (0 / 1)	Bit 10	0 … 1 (1 bit)
TSF_FLAG	flag (0 / 1)	Bit 11	0 … 1 (1 bit)
		Bit 12	0 … 1 (1 bit)
		Bit 13	0 … 1 (1 bit)
		Bit 14	0 … 1 (1 bit)
ADC_PREAMP_FLAG	flag (0 / 1)	Bit 15	0 … 1 (1 bit)
Word 2 – Flags			
Word 3			
DAC_GAIN_MIN	0 … 15, int	Bit [12:15], DAC_GAIN_MIN	0x0 … 0xF (4 bit)
DAC_GAIN_MAX	0 … 15, int	Bit [8:11], DAC_GAIN_MAX	0x0 … 0xF (4 bit)

ADC_GAIN	0 … 15, int	Bit [4:7], ADC_GAIN	0x0 … 0xF (4 bit)
REF_SHIFT	0 … 15, int	Bit [0:3], REF_SHIFT	0x0 … 0xF (4 bit)
Word 4			
IG	+24 … -96, dB	Bit [8:15], round(255*((10^((IG-24)/20))^0.25))	0x00 … 0xFF (8 bit)
OG	+36 … -96, dB	Bit [0:7], round(255*((10^((OG-36)/20))^0.25))	0x00 … 0xFF (8 bit)
Word 5			
RPD	-200 … 200, ms	Bit [8:15], RPD/2	0x00 … 0xFF (8 bit)
-	-	Bit[6:7]: -	0x0 … 0x3 (2 bit)
DEBUG_SND	0,1,2,3	Bit [4:5] 0 – disable, 1 – TX beeps, 2 – sweep tone, 3 – white noise	0x0 … 0x3 (2 bit)
LOG_PIO	0 … 15	Bit[0:3], LOG_PIO	0x0 … 0xF (4 bit)
Word 6			
AFL	8 … 128, ms	Bit [8:15], round(AFL(ms)/4)	0x02…0x20 (8 bit)
ARF	0.0 … 1.0, f	Bit [0:7], ARF * 255	0x00 … 0xFF (8 bit)
Word 7			
ADM_FRC	0 … -96, dB	Bit [8:15], round(255*((10^(ADM_FRC /20))^0.25))	0x00 … 0xFF(8 bit)
ADMFG	-12 … +18, dB	Bit[3:7], round(31*(((10^(ADMFG /20))/8)^0.25)); (-0.9 … 1.0 float) --> 1.0	0x0 … 0x1F (5bit)
ADMMEG	0 … 2.0, f	Bit [0:2], round(7*(ADMMEG /2))	0x0 … 0x7 (3 bit)
Word 8			
ADMNLPG	0 … -23, dB	Bit [12:15], round(15 *10^(ADMNLPG / 20))	0x0 … 0xF (4 bit)
ADMWNG	0 … -23, dB	Bit [8:11], round(15 *10^(ADMWNG / 20))	0x0 … 0xF (4 bit)
ADMMAA	45 … 90, deg	Bit [4:7], round((ADMMAA1 - 45)/3)	0x0 … 0xF (4 bit)
ADMMPE	0 … 15, deg	Bit[0:3], ADMMPE	0x00 … 0xF (4 bit)
Word 9			
ADMDIST	10 … 128, mm	Bit [8:15]: Bit8 = 0 (close talk or far field), Bit[9:15], ADMDIST	0x00 … 0xFF(8 bit)
	8 … 1024, mm	Bit [8:15]: Bit8 = 1 (broad-side), Bit[9:15],	0x00 … 0xFF(8 bit)

		(ADMDIST / 8)	
ADM_ARFN	0.0 … 1.0, f	Bit [0:7], round(((ADM_ARFN^0.5) * 255)	0x00 … 0xFF (8 bit)
Word 10			
ADMUDF	0…8000 Hz	Bit[10:15]: ADMUDF/250	0x00 … 0x3F (6 bit)
ADMSTR	-90 … +90, deg	Bit[4:0], round(ADMSTR /3)	0x00 … 0x3F (6 bit)
ADMMODE	0,1,2,3	Bit[2:3]: 0 – Far Talk, 1 – Close Talk, 2 – Broad-side - HS, 3 – Broad-side - HF	0x0 … 0x3 (2bit)
ADMCNF	0,1,2,3	Bit[0:1], 0 - dual mic , 1 - first, 2 - second, 3 - difference	0x0 … 0x3 (2 bit)
Word 11			
BNL	0 … -96, dB	Bit [8:15], round(255*((10^(BNL /20))^0.25))	0x00 … 0xFF(8 bit)
BNLMING	0 … -96, dB	Bit[0:7], round(255*((10^(BNLMING /20))^0.25))	0x00 … 0xFF(8 bit)
Word 12			
SNS	0 … -40, dB	Bit [8:15], round(255 *10^(SNS / 20))	0x00 … 0xFF (8 bit)
TONEMING	0 … -96, dB	Bit[0:7], round(255*((10^(TONEMING /20))^0.25))	0x00 … 0xFF(8 bit)
Word 13			
SNAT	100 … 3060, ms	Bit [8:15], round(SNAT / 12)	0x00 … 0xFF (8 bit)
TNAT	0 … 3060, ms	Bit [0:7], round(TNAT / 12)	0x00 … 0xFF (8 bit)
Word 14			
-	-	Bit[14:15] -	0x0 … 0x3 (2bit)
WBNGG	0 … -9, dB	Bit[11:13], 7 = 0dB, 6=-3dB, 5=-4dB, 4=-5dB, 3=-6dB, 2=-7dB, 1=-8dB, 0=-9dB	0x0 … 0x7 (3bit)
BNLEF	1 … 8, int	Bit [8:10], BNLEF – 1	0x00 … 0x7 (3 bit)
NTF	-1.0 … 1.0, f	Bit [0:7], NTF * 127	0x00 … 0xFF (8 bit)
Word 15			
-DRC1_SAT	+24 … -96, dB	Bit[8:15], round(255*((10^((DRC1_SAT-24) /20))^0.25))	0x00 … 0xFF(8 bit)
DRC2_SAT	+24 … -96, dB	Bit[0:7], round(255*((10^((DRC2_SAT-24) /20))^0.25))	0x00 … 0xFF(8 bit)

Word 16			
-DRC1_SL	0 … -96, dB	Bit[8:15], round(255*((10^(DRC1_SL/20))^0.25))	0x00 … 0xFF(8 bit)
DRC1_NIL	0 … -96, dB	Bit [0:7], round(255*((10^(DRC1_NIL /20))^0.25))	0x00 … 0xFF(8 bit)
Word 17			
-	-	Bit[8:15],	0x00 … 0xFF(8 bit)
DRC1_NOL	0 … -96, dB	Bit [0:7], round(255*((10^(DRC1_NOL /20))^0.25))	0x00 … 0xFF(8 bit)
Word 18			
AGC_MXGL	0 … -96, dB	Bit[8:15], round(255*((10^(AGC_MXGL/20))^0.25))	0x00 … 0xFF(8 bit)
AGC_ZGL	0 … -96, dB	Bit[0:7], round(255*((10^(AGC_ZGL /20))^0.25))	0x00 … 0xFF(8 bit)
Word 19			
AGC_RT	32… 8192, ms	Bit [8:15], round(AGC_RT/32)	0x00 … 0xFF (8 bit)
AGC_LTHR	0 … -96, dB	Bit[0:7], round(255*((10^(AGC_LTHR /20))^0.25))	0x00 … 0xFF(8 bit)
Word 20			
--			
FREQSH	0…7, Hz	Bit [11:13], FREQSH	0x00 … 0x7 (3 bit)
FREQSC	0… 1.0, f	Bit [8:10]: round(FREQSC/0.15)	0x00 … 0x7 (3 bit)
~~VADTHR~~ AGC_VTHR	0.0…0.933, f	Bit[4:7]: 1 + round(AGC_VTHR * 15); 0 is treated as AGC_VTHR =0.25f (for compatibility)	0x0 … 0xF(4 bit)
AGC_MXG	0 … +45, dB	Bit [0:3], 0, +3.0, +.6.0, …, +45, dB	0x0 … 0xF (4 bit)
Word 21			
TX1OUT_LOG_FLAG	flag (0 / 1)	Bit 0	0 … 1 (1 bit)
TX2OUT_LOG_FLAG	flag (0 / 1)	Bit 1	0 … 1 (1 bit)
TX1IN_LOG_FLAG	flag (0 / 1)	Bit 2	0 … 1 (1 bit)

TX2IN _LOG_FLAG	flag (0 / 1)	Bit 3	0 … 1 (1 bit)
TX3IN _LOG_FLAG	flag (0 / 1)	Bit 4	0 … 1 (1 bit)
TX4IN _LOG_FLAG	flag (0 / 1)	Bit 5	0 … 1 (1 bit)
Word 22			
PSKEY_SEC	0x0000 … 0xFFFF		0x0000 … 0xFFFF (16 bit)
Word 23			
PSKEY_LIC	0x0000 … 0xFFFF		0x0000 … 0xFFFF (16 bit)
Word 24			
PSKEY_MON	0x0000 … 0xFFFF		0x0000 … 0xFFFF (16 bit)
Word 25 ... 37			
-	-	-	-
Word 38			
-	-	-	-
MIC_MUX		Sync ID: 0 – sink0, 1 – sink1, 2 – sink2, 3 – sink4. TX1: Bit[0:1], TX2: Bit[2:3], TX3: Bit[4:5], TX4: Bit[6:7]. Default configuration: 0xE4. **Note:** Value 0x00 is interpreted as a default mic configuration (0xE4) for PSkey compatibility.	0x00 … 0xFF (8 bit)
Word 39 ... 46			
LIMITER_EQ	-∞,-84 … -12,-6, 0, dB	For Fs=16 kHz, band width must be multiplied by 2	0x0 … 0xF (4 bits) per band, MSB - lower band, LSB - higher band, 32 bands = 8 words
Word 47..54			
ADM_REAR_EQ	-∞,-28 … -4,-2, 0,	For Fs=16 kHz, band width must be	0x0 … 0xF (4 bits) per

	dB	multiplied by 2	band, MSB - lower band, LSB - higher band, 32 bands = 8 words
Word 55..62			
EQ	-oo,-28 … -4,-2, 0, dB	For Fs=16 kHz, band width must be multiplied by 2	0x0 … 0xF (4 bits) per band, MSB … lower band, LSB … higher band, 32 bands = 8 words
Word 63			
PSKEY_PRI_CRC	0 … 256	Bit[0:7], x_hi XOR x_low, x = word1 XOR word2 XOR…..word60	0x0 … 0xFF (8 bit)
PSKEY_VER	0 … 256	Bit[0:7], 0x0003	0x0 … 0xFF (8 bit)

Total Primary PSKEY length: 63 words (of 63 available)

Secondary (optional) HEP PSKEY structure

Parameter ID and word number in PSKEY	Physical values range	Location in PSKEY and conversion formula into fixed-point representation	Fixed-point representation range and size
Word 1 … 8			
ARF_EQ	-oo,-28 … -4,-2, 0, dB	For Fs=16 kHz, band width must be multiplied by 2	0x0 … 0xF (4 bits) per band, MSB … lower band, LSB … higher band, 32 bands = 8 words
Word 9 … 62			
Word 63			
PSKEY_SEC_CRC	0 … 256	Bit[0:7], x_hi XOR x_low, x = word1 XOR word2 XOR…..word32	0x0 … 0xFF (8 bit)
PSKEY_VER	0 … 256	Bit[0:7], 0x0002	0x0 … 0xFF (8 bit)

Total Secondary PSKEY length: 63 words (of 63 available)