DECT/CT2 BBSP Software Package

Literature Number: BPRA052
Texas Instruments Europe
March 1997

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Contents

List of Figures

List of Tables

DECT/CT2 BBSP SOFTWARE PACKAGE

ABSTRACT

The Texas Instruments ADPCM software support's a full list of features for use in ADPCM cordless telephones such as DECT and CT2.

- Full G.726 ADPCM Voice Coder (Including G.721/G.723)
- Supports A-Law, μ-Law and Linear
- Mutes for Transmit and Receive Channels
- Local Side-tone Generation
- Dual Tone Generator (DTMF Capable)
- DECT Echo Control software (Cancellation and Suppression)
- ADPCM receive volume control
- Modem Answer Tone Detection for Automatic Disabling of Echo Control
- Built in Test Routines

1. Software Description

The Texas Instruments ADPCM software supports a full list of features for use in ADPCM cordless telephones such as DECT and CT2.

1.1 ADPCM

The transcoding algorithm is fully compliant with the ITU G.726 recommendation (Incorporating G.721 and G.723 Blue Book recommendations) for transcoding 64Kbit/s PCM data to/from 16, 24, 32 or 40Kbit/s ADPCM data.

1.2 Echo Control

The DECT specification requires that for the specified signal levels an overall echo reduction be made of >34dB from a combination of cancellation, suppression, and hybrid matching. Hybrid matching is required to give -13±5dB. A maximum of 12dB may be obtained by suppression. With a worst case hybrid of -8dB and a maximum suppressor of -12dB, 14dB must be obtained by cancellation. The recommended values for each component are hybrid -13dB, canceller -20dB suppressor -9dB.

Table 1: ADPCM Echo Control

Source	Min	Nom	Max
Hybrid	-8dB	-13dB	-18dB
Canceller		-20dB	
Suppressor	-9dB	-9dB	-12dB
Total	-34dB	-42dB	

1.3 Echo Canceller

For all input signals inside the valid range of 0 -30dBm echoes are cancelled by at least 20dB subject to an echo canceller quantization floor of -60dBm0. The quantization floor limit does not affect performance when operated with the worst case hybrid, though, when hybrids that are better than that required are used, the echo canceller performance may be degraded for low signal levels, the overall specification for echo reduction of 34dB will however still be met.

Reduction below -60dbm0 requires the use of the Echo Suppressor. Echo Cancellation is required by DECT for echoes of 0…4ms after transmission and the BBSP4CH easily meets this requirement by canceling echoes 0…8ms after transmission.

1.4 Echo Suppressor

The receive path is suppressed by the echo suppression constant 0-24dB when a speech level which is greater than the background noise is detected. The background noise is measured by an adaptive technique where the log term signal is averaged over a long period of time. This average is limited in range from -21.2 to -32.5dBm0 so signals above -21.2dBm0 are always treated as speech and signals below -32.5dBm0 are always treated as background noise. The receive path is suppressed by the echo suppression constant 0-24dB when a speech level which is greater than the measured background is transmitted, the suppression is soft switched in over 6ms (DECT requires <10ms) if the transmitted signal falls below this threshold before the end of the switch on period then suppression will be returned to zero within 30ms. Once 6ms of signal above the background average + suppressor margin has been received, suppression will continue for 70ms after the transmitted signal falls below the measured background noise, it will then soft switch out over 30ms. The echo suppression constant can be programmed to be off, 3, 6, 9, 12, 15, 18 or 24dB. Only echo suppressions of 9 and 12dB are valid for use in a DECT system. The amount of signal above the background noise that is required to trigger the Echo Suppression is also programmable between 2.4, 3.5 and 4.5dB. This is to stop minor variations in the background noise from triggering the Echo Suppressor. The quantization noise floor of the suppressor is below -67dBm0.

2. System Cycle Times and Memory Requirements

2.1 Memory

The table below shows the memory requirements of the various routines a dummy routine MAINSRC has been included which drives a particular implementation to give an idea of the size of control program needed to control the main BBSP coding routines.

Table 2: ADPCM Memory Routines requirements

Module	Program ROM (Words)	Data ROM (Bytes)	Data RAM (Bytes)
MAINSRC	**563**	**6**	**52 (incl stack)**
ADPCM 16	**434**	**44**	**80x2 (160)**
ADPCM 24	**438**	**52**	**80x2 (160)**
ADPCM 32	**442**	**82**	**80x2 (160)**
ADPCM 40	**450**	**156**	**80x2 (160)**
TONES	**139**	**56**	**12**
SELFTEST	**269**	**8**	**0**
ECHO (8ms)	**227**	**66**	**298**

In the echo subroutine 2 words can be saved for every 125µs not cancelled down to 4ms. Below 4ms only 1 word can be saved per 125µs.

2.2 Cycles

Table 3: ADPCM Core Cycle Requirements

Rate	A-Law			µ-Law			Linear		
	Min	Nom	Max	Min	Nom	Max	Min	Nom	Max
16KBPS	**528**	**938**	**965**	**530**	**934**	**958**	**516**	**883**	**903**
24KBPS	**528**	**947**	**980**	**530**	**943**	**973**	**516**	**891**	**917**
32KBPS	**528**	**956**	**996**	**530**	**952**	**989**	**516**	**900**	**933**
40KBPS	**528**	**968**	**1,007**	**530**	**961**	**997**	**516**	**909**	**941**

Table 4: Echo Canceller Cycle Requirements

Subroutine (0ms Dead Time)	Cancel Only	Update
Dead-Time Per 125µs	**6**	**0**
Echo (3ms)	**551**	**203**
Echo (4ms)	**724**	**256**
Echo (5ms)	**897**	**309**
Echo (6ms)	**1070**	**362**
Echo (7ms)	**1243**	**415**
Echo (8ms)	**1416**	**468**

Table 5: Miscellaneous Cycle Requirements

Subroutine	Min	Max
Tone Generation	**67**	**67**
Echo Suppression	**49**	**79**

Update Timings are for use when update is enabled in software and the line conditions for Update are also valid. Update times are reduced if enabled in SW but line conditions are invalid.

Additional Cycles need to be allowed for interrupt service routines and command processing, typically this will be less than 100 cycles.

In a typical DECT application the following combination might be used:

Table 6: Typical DECT Application Cycles

Block	Cycles
Control	**60**
ADPCM 32KBPs	**996**
Echo Suppression	**79**
Echo Cancellation (4ms)	**724+256**
Total	**2115**

3. ADPCM.ASM

This software is fully compliant with G.721, G.723 and G.726 please see the relevant ITU specification for full details of the equations implemented.

This Module contains the actual ADPCM voice coder subroutines. The following entry subroutines should be used by ADPCM voice coder control programs like DUAL.ASM.

1) init_code — This subroutine initialises all variables needed for ADPCM encoding, it can be called at any time, no parameters are passed to it and none are returned. All internal registers except M are used and are undefined on exit.

2) init_decode — This subroutine initialises all variables needed for ADPCM decoding, it can be called at any time, no parameters are passed to it and none are returned. All internal registers except M are used and are undefined on exit.

3) code — This subroutine takes a 14 bit 2's complement linear PCM sample from variable "*pcm_in*" and ADPCM codes returning the 2,3,4 or 5 bit ADPCM value in "*adpcm_out*". Both "*pcm_in*" and "*adpcm_out*" must be in the same SD7 page as the RAM for the encoder.

4) decode_A — This subroutine takes a 2, 3,4 or 5 bit ADPCM value in "*adpcm_in*" and ADPCM decodes it with A-Law synchronous coding adjustment returning a 15 bit 2's complement linear PCM sample for compression in "*pcm_out*" for the main program. Both "*adpcm_in*" and "*pcm_out*" must be in the same SD7 page as the RAM for the decoder.

5) decode_U — This subroutine takes a 2, 3, 4 or 5 bit ADPCM value in "*adpcm_in*" and ADPCM decodes it with μ-Law synchronous coding adjustment returning a 15 bit 2's complement linear PCM sample for compression in "*pcm_out*" for the main program. Both "*adpcm_in*" and "*pcm_out*" must be in the same SD7 page as the RAM for the decoder.

6) decode_L — This subroutine takes a 2, 3, 4 or 5 bit ADPCM value in "*adpcm_in*" and ADPCM decodes it without synchronous coding adjustment returning a 15 bit 2's complement linear PCM sample for compression in "*pcm_out*" for the main program. Both "*adpcm_in*" and "*pcm_out*" must be in the same SD7 page as the RAM for the decoder.

All other subroutines in this module are called by these subroutines and should not be used. If the software is not going to be used in any of A-Law, μ-Law, or Linear mode then the respective irrelevant subroutines; decode_A, decode_U and/or decode_L may be safely removed as they are not called internally.

When linking in these routines it is necessary to be very careful about how the memory is allocated. The module generates data space for 1 (Code or Decode) channel only; this must be placed in a data page with D7=0, this will then produce 2 logical maps with D7=0 for code and D7=1 for decode. Variable "*pcm_in*" and

"*adpcm_out*" must reside in the page with D7=0, and "*pcm_out*" and "*adpcm_in*" in the page with D7=1. Failure to observe this will cause the voice coder to fail. All data ROM variables must exist on a page with D7=1 so that they are not affected by the SD7 bit during coding.

The memory map produced using DUAL_ROM.LNK is an example of how to do this correctly. This places the ADPCM data RAM as shown below:

Block	Memory Location
Stack	**0x00..0x1f**
Code	**0x20..0x6f**
IO/Misc Control	**0x70..0x9f**
Decode	**0xa0..0xef**
Misc	**0xf0..Top of RAM**

3.1 Modem Answer Tone Detection

This function is performed within the G726 ADPCM function by means of some non-invasive additional code, this code principally monitors the 2nd order all pole adaptive predictor and the adaptive quantizer, in order to determine the frequency, signal/noise ratio and amplitude of the signal.

The second order predictor taken from the G726 specification is:

$$Se(k) = \sum_{i=1}^{2} a_i(k-i) \times S_r(k-i)$$

Expanding this gives:

$$Se(k) = a_1(k-1) \times S_r(k-1) + a_2(k-2) \times S_r(k-2)$$

Freezing the time variation in frequency gives:

$$Se(k) = a_1 \times S_r(k-1) + a_2 \times S_r(k-2)$$

Using the standard unit circle conversion equations for a second order filter:

$$a_1 = 2 \times r \times \cos(\theta)$$

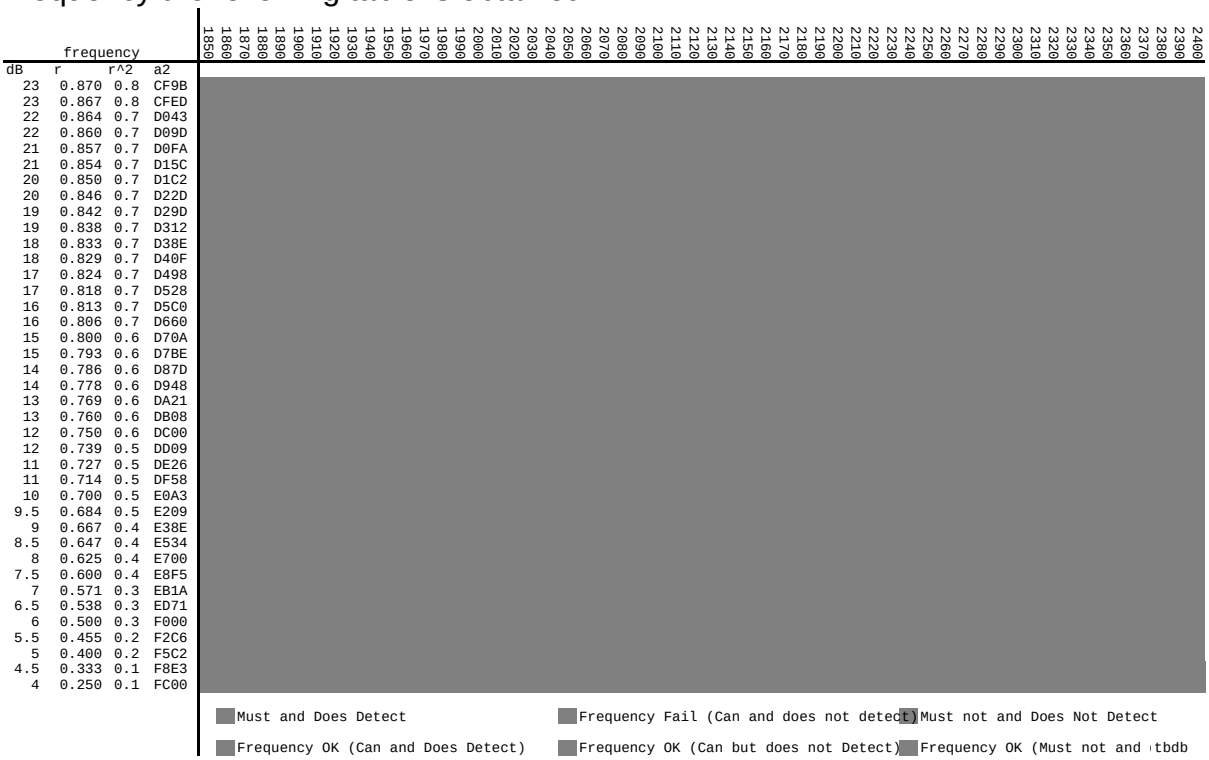
$$a_2 = -r^2$$

Results in:

$$Se(k) = 2 \times r \times \cos(\theta) \times S_r(k-1) - r^2 \times S_r(k-2)$$

It is now obvious the adaptive parameters gives the 'Q' and frequency of the predictor, and that A2 is proportional to the square of the 'Q' and A1 proportional to both the 'Q' and the frequency. Whilst A1 does vary with frequency for reasonably clean tones the frequency dependency is significantly narrower than the guard bands specified in G.164 for the detection of the modem answer tone and the frequency can therefore be determined sufficiently accurately. If it is assumed that frequency is independent of 'Q' and frequencies inside the tolerance region are chosen for the boundaries, then these frequencies will tend to increase with decreasing 'Q'

(increasing S/N). Overall, on plotting A1 against both Signal/Noise ratio and real frequency the following table is obtained.



dB	r	r^2	a2
23	0.870	0.8	CF9B
23	0.867	0.8	CFED
22	0.864	0.7	D043
22	0.860	0.7	D09D
21	0.857	0.7	D0FA
21	0.854	0.7	D15C
20	0.850	0.7	D1C2
20	0.846	0.7	D22D
19	0.842	0.7	D29D
19	0.838	0.7	D312
18	0.833	0.7	D38E
18	0.829	0.7	D40F
17	0.824	0.7	D498
17	0.818	0.7	D528
16	0.813	0.7	D5C0
16	0.806	0.7	D660
15	0.800	0.6	D70A
15	0.793	0.6	D7BE
14	0.786	0.6	D87D
14	0.778	0.6	D948
13	0.769	0.6	DA21
13	0.760	0.6	DB08
12	0.750	0.6	DC00
12	0.739	0.5	DD09
11	0.727	0.5	DE26
11	0.714	0.5	DF58
10	0.700	0.5	E0A3
9.5	0.684	0.5	E209
9	0.667	0.4	E38E
8.5	0.647	0.4	E534
8	0.625	0.4	E700
7.5	0.600	0.4	E8F5
7	0.571	0.3	EB1A
6.5	0.538	0.3	ED71
6	0.500	0.3	F000
5.5	0.455	0.2	F2C6
5	0.400	0.2	F5C2
4.5	0.333	0.1	F8E3
4	0.250	0.1	FC00

As can be seen the detected band includes all of the required band, excludes all of Must Not Detect band and includes a small proportion of the tolerance region. This level of detection can therefore be obtained using simple comparisons of the predictor coefficients without having to perform any other filtering.

The modem answer tone specification additionally specifies that the signal level must be above a certain level before detection can occur. While the indication of level within the predictor varies on a sample-to-sample basis and is not very good, level information is, however, also available from the adaptive quantizer. In G726 the quantizer has an internal adaption gain, Y, that is roughly proportional to the error output from the predictor, and this is obviously dependant on the quality of prediction for a given type of signal. Luckily, the only interesting type of signal is a 2100Hz tone with a S/N of better than 12dB. For this type of signal the internal quantizer adaption gain is reasonably well defined and a simple comparison can be used to determine if the signal is above the required level. This level was obtained empirically by simulating the minimum tone that was required to be detected. Various forms of Y are available and there is little to choose between them so, in the end, YU was chosen.

The fourth and final criterion for the detection of a modem answer tone is that the tone must be present for long enough to be not falsely triggered by speech signals. To detect this, a simple counter is introduced that counts how many consecutive samples have met the above conditions. To ensure that minor disturbances do not adversely affect the detection, the clock is incremented for good detection and decremented by

a small amount for near misses and a significant amount for signals that are well outside the detection zone.

Overall the equations for detection become:

$A1_{fr\_low} \geq A1_{now} \geq A1_{fr\_high}$	*Frequency*	
$A2_{now} \leq A2_{thr1}$	*Signal / Noise*	*Counter +*
$YU_{now} \geq YU_{thr}$	*Amplitude*	
$A2_{now} \leq A2_{thr2}$	*NonTone*	*Counter – –*
	Otherwise	*Counter –*
$Counter_{now} \geq Counter_{thr}$	*Time*	*SingalDetection*

As can be seen this can be implemented in very few MIPs.

4. Software Contents

4.1 ADPCM16.ASM, ADPCM24.ASM, ADPCM32.ASM, ADPCM40.ASM

These modules assign the assembly time constant ADPCM to 16, 24, 32 or 40 to assemble the main ADPCM.ASM program in any of 16KBPS, 24KBPS, 32KBPS or 40KBPS respectively.

4.2 TONE.ASM

The tone generation software consists of two independent tone generators generating tone by the standard impulse induced ring oscillator technique. The equation for this is shown below, where SR are the sequential samples:

$$SR_{n+1} = A_1 \times SR_n + A_2 \times SR_{n-1}$$

Where the filter gain is r and there are θ radians between adjacent samples.

$$A_1 = 2 \times r \times \cos(\theta)$$
$$A_2 = -r^2$$

For continuous sine wave the gain must be 1. Substituting r=1 gives:

$$A_1 = 2 \times \cos(\theta)$$
$$A_2 = -1$$

If initialised on a zero crossing then SR_n= 0 and SR_{n-1}= sin(θ). Only A_1 and SR_{n-1} therefore need to be programmed.

The values for programming as DTMF dial tones are stored internally and can be loaded with the DTMF generator. The actual frequencies generated are shown below.

Table 7: Actual DTMF Frequencies Generated by BBSP

Required Frequency	Actual Frequency	Error	Required Frequency	Actual Frequency	Error
697	696.97	+0.0045%	1209	1208.98	-0.0013%
770	770.02	+0.0025%	1336	1335.98	-0.0016%
852	852.02	-0.0027%	1477	1476.98	-0.0013%
941	940.99	-0.0006%	1633	1632.99	-0.0004%

Alternatively, routines for programming these parameters directly are also provided. In order to reduce the length of parameters to 8 bits, two 8-bit parameters Freq and Ampl are expanded to the 16-bit values for A_1 and SR_{n-1} using the following equations:

$$A_1 = -0.9375 \times (Ampl)^2 + 31860 \quad (0 < Ampl < 255)$$

$$A_2 = -1$$

$$SR_1 = 0$$

$$SR_2 = \frac{(Freq)^2}{2} \quad (0 < Freq < 255)$$

The values Freq and Ampl then generate tones with actual Frequency and Amplitude given by:

$$\text{Frequency(Hz)} = \frac{4000}{\pi} \times \text{Cos}^{-1}\left(\frac{-0.9375 \times \text{Freq}^2 + 31860}{32768}\right)$$

$$\text{Amplitude(dBm)} = 20 \times \text{Log}_{10}\left(\frac{\text{Ampl}^2 \times \text{Sin}\left(\frac{\text{Frequency} \times \pi}{4000}\right)}{32768}\right)$$

Rearranging these equations for the input parameters Ampl and Freq compared to the wanted parameters Amplitude and Frequency the following equations are obtained:

$$Freq = \sqrt{\frac{-32768 \times \cos\left(\frac{Frequency \times \pi}{4000}\right) + 31860}{0.9375}}$$

$$Ampl = \sqrt{\frac{32768 \times 10^{\left(\frac{Amplitude}{20}\right)}}{\sin\left(\frac{Frequency \times \pi}{4000}\right)}}$$

If	Frequency	=	1000Hz	and	Amplitude	=	-20dB
Then	Freq	=	96.274	and	Ampl	=	68.074

However Freq and Ampl are byte input values and ($0 \leq Freq \leq 255$),($0 \leq Amp \leq 255$), therefore, for the programmed values of Freq and Ampl, the actual Frequency and Amplitude become:

If	Freq	=	96	and	Ampl	=	68
Then	Frequency	=	997.27Hz	and	Amplitude	=	-20.03dB

All the output Frequencies for Freq are shown below:

0	300.4	52	588.9	104	1077.4	156	1643.9	208	2342.2
1	300.6	53	597.4	105	1087.5	157	1655.8	209	2357.9
2	301.1	54	606.0	106	1097.7	158	1667.7	210	2373.9
3	301.8	55	614.6	107	1107.9	159	1679.6	211	2389.9
4	302.9	56	623.2	108	1118.1	160	1691.6	212	2406.1
5	304.3	57	631.9	109	1128.3	161	1703.6	213	2422.4
6	306.0	58	640.7	110	1138.6	162	1715.7	214	2438.9
7	308.0	59	649.5	111	1148.9	163	1727.8	215	2455.6
8	310.3	60	658.3	112	1159.2	164	1740.0	216	2472.4
9	312.8	61	667.2	113	1169.5	165	1752.2	217	2489.3
10	315.6	62	676.1	114	1179.9	166	1764.5	218	2506.5
11	318.7	63	685.0	115	1190.3	167	1776.8	219	2523.8
12	322.1	64	694.0	116	1200.7	168	1789.2	220	2541.3
13	325.7	65	703.0	117	1211.2	169	1801.7	221	2559.0
14	329.6	66	712.0	118	1221.7	170	1814.1	222	2576.9
15	333.7	67	721.1	119	1232.2	171	1826.7	223	2595.0
16	338.0	68	730.2	120	1242.7	172	1839.3	224	2613.3
17	342.6	69	739.4	121	1253.3	173	1852.0	225	2631.8
18	347.3	70	748.6	122	1263.9	174	1864.7	226	2650.6
19	352.3	71	757.8	123	1274.5	175	1877.5	227	2669.6
20	357.5	72	767.0	124	1285.2	176	1890.3	228	2688.8
21	362.8	73	776.3	125	1295.9	177	1903.2	229	2708.3
22	368.4	74	785.6	126	1306.6	178	1916.2	230	2728.1
23	374.1	75	794.9	127	1317.3	179	1929.2	231	2748.2
24	379.9	76	804.3	128	1328.1	180	1942.3	232	2768.6
25	385.9	77	813.7	129	1338.9	181	1955.4	233	2789.2
26	392.1	78	823.1	130	1349.8	182	1968.7	234	2810.3
27	398.4	79	832.5	131	1360.6	183	1982.0	235	2831.6
28	404.9	80	842.0	132	1371.5	184	1995.3	236	2853.4
29	411.5	81	851.5	133	1382.5	185	2008.8	237	2875.5
30	418.2	82	861.0	134	1393.5	186	2022.3	238	2898.1
31	425.0	83	870.6	135	1404.5	187	2035.9	239	2921.1
32	432.0	84	880.2	136	1415.5	188	2049.6	240	2944.6
33	439.0	85	889.8	137	1426.6	189	2063.3	241	2968.5
34	446.2	86	899.4	138	1437.7	190	2077.1	242	2993.1
35	453.5	87	909.1	139	1448.8	191	2091.0	243	3018.2
36	460.8	88	918.8	140	1460.0	192	2105.0	244	3043.9
37	468.3	89	928.5	141	1471.2	193	2119.1	245	3070.3
38	475.8	90	938.3	142	1482.4	194	2133.3	246	3097.4
39	483.5	91	948.0	143	1493.7	195	2147.5	247	3125.4
40	491.2	92	957.8	144	1505.0	196	2161.9	248	3154.2
41	498.9	93	967.7	145	1516.4	197	2176.3	249	3184.0
42	506.8	94	977.5	146	1527.8	198	2190.9	250	3214.9
43	514.7	95	987.4	147	1539.2	199	2205.5	251	3247.1
44	522.7	96	997.3	148	1550.7	200	2220.2	252	3280.6
45	530.8	97	1007.2	149	1562.2	201	2235.1	253	3315.7
46	538.9	98	1017.2	150	1573.7	202	2250.0	254	3352.7
47	547.1	99	1027.1	151	1585.3	203	2265.1	255	3391.9
48	555.4	100	1037.1	152	1597.0	204	2280.3		
49	563.7	101	1047.2	153	1608.6	205	2295.6		
50	572.0	102	1057.2	154	1620.4	206	2311.0		
51	580.4	103	1067.3	155	1632.1	207	2326.5		

TONE.ASM contains all the tone generation functions:

1) InitDtmf Initialises both tones to the DTMF tone number passed in the B register. Uses A0 and X registers.

Digit	Value in B
0-9	0-9
*	10
#	11
A	12
B	13
C	14
D	15

2) INIT_A1_A Initialises frequency 1's A1 variable according to the equations described above. The input is passed in the upper 8 bits of A0. Uses A0, A2 and M registers.

3) INIT_A1_B Initialises frequency 2's A1 variable according to the equations described above. The input is passed in the upper 8 bits of A0. Uses A0, A2 and M registers.

4) INIT_SR2_A Initialises frequency 1's SR1 and SR2 variables according to the equations described above. The input is passed in the upper 8 bits of A0. Uses A0, A2 and M registers.

5) INIT_SR2_B Initialises frequency 2's SR1 and SR2 variables according to the equations described above. The input is passed in the upper 8 bits of A0. Uses A0, A2 and M registers.

6) ToneSamp Generates the next DTMF output sample returns a 15 bit 2's complement linear PCM sample in a0. Uses A0,A2,B,X and M, registers.

4.3 ST_DUAL.ASM

This subroutine performs a simple self-test of the basic hardware of the BBSP. The following tests are performed.

1. Initialise data RAM to pseudo random sequence for later check-summing.
2. Check parallel access to the IO space.
3. Perform a sequence of tests using each of the BBSP's arithmetic instructions (6 times) and each register (18 times) on the random RAM data, and writing the results in the register back to the RAM area for later check-summing.
4. Check that each branch instruction occurs when its flag is valid and does not occur when its flag is invalid.
5. Add the first 64 random numbers in the data RAM each with a different shift value together and store the result back in the data RAM for later check-summing.
6. Checksum the RAM and ROM locations using an xor and rotate algorithm and XOR with the expected value to confirm that the above tests have passed. The last 6 locations, which are changed by Texas Instruments for manufacturing tests, are not checked by this routine as their values are not determined at design time.
7. Check all the RAM with a checkerboard and inverted checkerboard pattern.
8. Check that the data BUS can read and write 05555H and 0AAAAH.
9. Return to the calling program with the Z flag set if all tests have passed and clear if any test has failed. If a test is failed the program ROM location where the failure was detected is written to the data memory location 0x20 of page 0.

Please note that the self-test program destroys all RAM locations and Register values when run and all other software will need to be re-initialised afterwards for normal operation to resume. The 16bit hexadecimal checksum number is dependant on the data ROM contents and needs to be changed for all software using this sub-routine; the high byte of the checksum is defined by chksum_h and the low byte by chksum_l at the beginning of this file.

4.4 ECHOBBSP.ASM

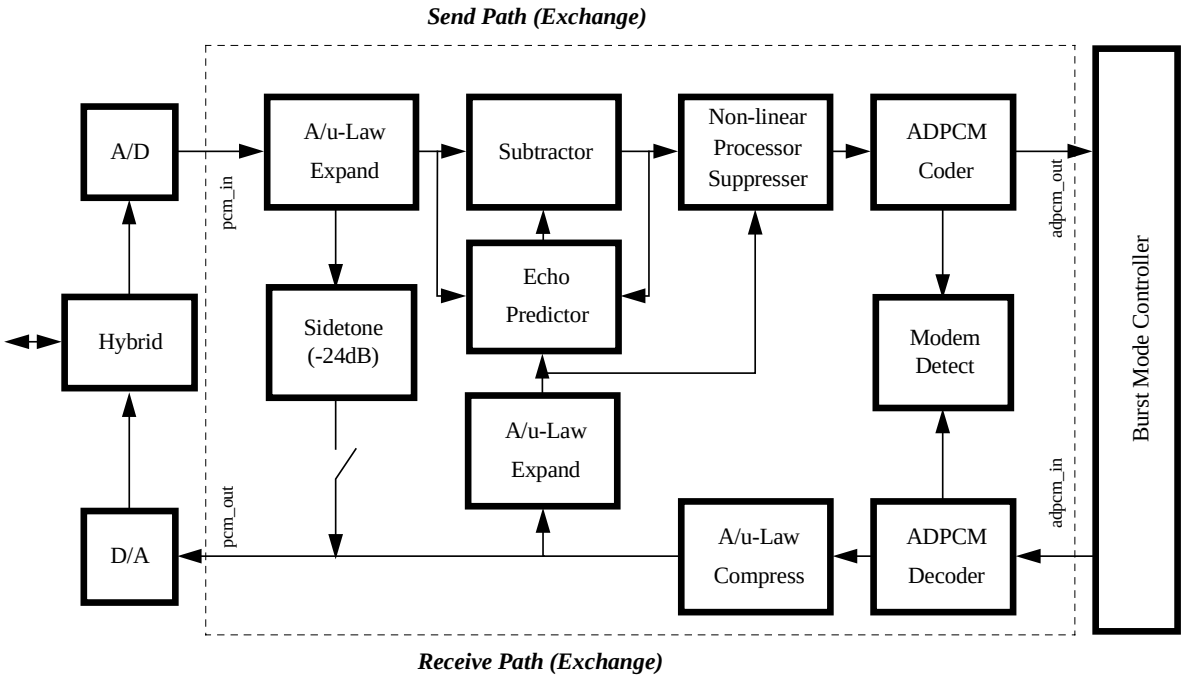


Figure 1: BBSP Echo Control Software Block

5. Algorithm

5.1 Description

The Texas Instruments Echo control software on the BBSP4CH is based on the ETSI 300 175-8 transmission requirements for DECT specification. This specification itself refers to the ITU standard G.165 and, where echo control parameters are not specified specifically for DECT, the values specified in the G.165 recommendation have been used.

5.2 Echo Prediction

The general algorithm of a Transversal FIR predictor is given in (1) below:

$$Y(n) = \sum_{k=0}^{31} W_k(n) * AIRIN(n - k)$$

(1) Transfer Structure of FIR Predictor

where:

w(n) is the weighting factor for each FIR parameter
AIRIN(n) is the input signal delayed by k samples
y(n) is the predicted echo on the near end signal

$$LRES(n) = LINEIN(n) - Y(n)$$

(2) Calculation of Error Signal from input and predicted signal

LRES(n) is the echo cancelled signal from the near end
LINEIN(n) is the near end signal including echo

Using this predicted signal the predicted echo is subtracted or cancelled from the return path as shown in (2) to produce the predicted error or echo cancelled signal. When there is no signal from the near end, this signal will be the error in the echo predictor. However, when there is a signal from the near end this signal will represent the near end signal with the echo from the far end cancelled out. In order to properly update the predictor it is necessary to know which of these is the case. The algorithm therefore contains a number of conditions that have to be met before the predictor is allowed to be updated.

1. The signal being sent from the far end must be above a certain threshold, i.e., there must be speech at the far end for there to be an echo capable of being cancelled.
2. The signal being received from the near end must be significantly less than that being sent from the far end, i.e., the predictor must not be updated during double talk.
3. Continuous jitter of the predictor variables can cause noise/distortion to the reception of the near end signal; if the noise of the echo is less than the noise caused by the update of the predictor it is better not to update the predictor.

The near end, far end and echo cancelled signal levels are continuously monitored using infinite impulse response filters with a short time constant so that zero-crossovers are not rejected as not being speech, but the filter rapidly detects short silences in the speech. The equations for both these filters are shown in (3) below. These IIR filters use a simple single pole decay that can be implemented using only shifts without the need for a multiplication.

$$Signal_{Level}(n) = Signal_{Level}(n-1) + \left[\frac{|Signal_{Input}(n)| - Signal_{Level}(n-1)}{32}\right]$$

(3) Equation for Measuring Mean Input Signal Level

By comparing the results of this equation for AIRIN a suitable threshold for speech transmission can be determined and the predictor updated only when the level exceeds this threshold. Double talk is detected by comparing the signal on both these filters and when the AIRIN filter is sufficiently greater than the LINEIN filter then the predictor can safely be updated. Alternatively the scaling factor for the update can be based on the relative strengths of the 2 signals, though this can give problems when the algorithm is initially reset if the echo signal is too strong.

5.3 Predictor Update

In order to update the predictor it is necessary to change the W_k parameters of the predictor, the faster these parameters are changed the faster the filter will converge on to the actual echo. However, if these parameters are changed rapidly a significant distortion is introduced to the signal received from the lines; indeed if too much feedback is applied, then the filter can even go unstable. It is therefore necessary to reach a compromise between update rate and the distortion introduced to the signal. From (1) and (2) it is obvious that the error in the predicted signal can be given as in (4) below.

$$LRES(n) = LINEIN(n) - \sum_{k=0}^{31} W_k(n) * AIRIN(n - k)$$

(4) Error Calculation for FIR Predictor

Although it is beyond the scope of this document, it has been shown that the update for this predictor is of the form shown in (5) below. Anyone interested should read the section on “Implementation of Adaptive Filters” in reference 1.

Given that the delay to the echo that is being cancelled is not a rapidly changing

$$W_k(n+1) = W_k(n) + u * LRES(n) * AIRIN(n - k)$$

(5) Ideal Predictor Update Equation

variable it is not necessary in this application to maximise the convergence rate of the echo canceller. The u constant can be made relatively small giving an echo canceller that produces a low level of distortion. A further simplification is to replace the error or the data of this equation - or both - by the sign of the error or data respectively. For each replacement the time required to achieve lock is approximately doubled; as the BBSP's ASCD instruction can perform the prediction fastest with a single sign substitution, it is this substitution that has been used. Equation (5) therefore is simplified to (6).

$$W_k(n+1) = W_k(n) + u * LRES(n) * SIGN(AIRIN(n - k))$$

(6) Error * Sign(Data) Predictor Update Equation

As the update rate of this filter is proportional to the input signal level the filter tends to converge faster for higher signal levels. This can be counteracted by normalising the update rate to the level of the input signal AIRIN. Care has to be taken not to place too much emphasis on individual samples as, near zero crossings, the error can be comparatively large compared to the signal level, and the signal from the LINEIN input is delayed by the echo. To avoid these problems the predictor update uses the mean input signal level for the AIRIN variable rather than the instantaneous signal level. The normalisation constant is given in (7) which, when substituted in (6), gives (8).

Equation (8) also shows the actual constant used to set the convergence stability compromise.

$$\text{Normalisation} = 2^{\text{INT}(\text{LOG}_2(\text{AIRIN}_{\text{Level}}))}$$

(7) Normalisation Equation

$$W_k(n+1) = W_k(n) + \left(\frac{2^{\text{INT}(\text{LOG}_2(\text{AIRIN}_{\text{Level}}(n)))} * \text{LRES}(n) * \text{SIGN}(\text{AIRIN}(n-k))}{512} \right)$$

(8) Normalised Error * Sign(Data) Predictor Update Equation

In order to stabilise this filter under tones and prevent overflows, a leak of 1/m is introduced into the update, the value of this leak is dependant on the level of cancellation achieved. The signal level before and after cancellation is measured using equation (3) above and the function shown in equation (9) below to determine their approximate amplitudes.

$$\text{INT}(\text{LOG}_2(\text{Signal}_{\text{level}}))$$

(9) Logarithmic Measure of Signal Level

The difference between these 2 logs gives the cancellation in 6dB steps, this is translated to a leak rate according to the values shown in the table below:

Measured Cancellation	Leak
>24dB	1/32768
18..24dB	1/16384
12..18dB	1/8192
6..12dB	1/4096
0..6dB	1/2048
-6..0dB	1/1024
…	…

This equation then becomes:

$$W_k(n+1) = \left(\frac{W_k(n) * (m-1)}{m} \right) + \left(\frac{2^{\text{INT}(\text{LOG}_2(\text{AIRIN}_{\text{Level}}(n)))} * \text{LRES}(n) * \text{SIGN}(\text{AIRIN}(n-k))}{512} \right)$$

(10) Leaky Normalised Error * Sign(Data) Predictor Update Equation

In terms of the actual inputs the overall equation for the update of the predictor coefficients is therefore:

$$W_k(n+1) = \left(\frac{W_k(n)*(m-1)}{m}\right) + \left(\frac{2^{INT(LOG_2(AIRIN_{Level}(n)))} * \left(LINEIN(n) - \sum_{l=0}^{31} W_l(n)*AIRIN(n-l)\right) * SIGN(AIRIN(n-k))}{512}\right)$$

$$m = 16384$$

(11) Full Predictor Parameter Update Equation

Overall this predictor will converge onto the echo in about 250µs, nearly cancelling the echo down to the floor level of the system. In order to further reduce the echo the use of an echo suppressor is required.

5.4 Echo Suppressor

The Echo Suppressor can be programmed to give various echo suppressions from 0dB to 24dB. Whatever it is programmed to give, the echo suppression is soft switched in and out with an attack time of 5ms and a decay time of 25ms. If full suppression is reached, there is a hangover time of 70ms after the end of speech before the echo suppressor is switched out.

In the echo suppressor, speech is detected using a simple 4th order FIR on the absolute values of the speech being sent to the line. The equation for this filter is shown in (12) below.

$$\begin{aligned} &\text{If } \sum_{k=0}^{7} |AIRIN(n-k)| > Supp\_Thresh \\ &\text{And } \sum_{k=24}^{31} |AIRIN(n-k)| > Supp\_Thresh \\ &\text{Then } Suppressor_{Change}(n) = Attack\_Constant \\ &\text{Else } Suppressor_{Change}(n) = Decay\_Constant \end{aligned}$$

(12) Echo Suppressor FIR Speech Detector

The $Suppress_{Change}$ is used to drive an asymmetric 2nd order IIR filter to generate the actual Suppress factor. This IIR is subject to a hangover time of 70ms, which is initiated once echo suppression has reached the maximum value. There is no hangover if echo suppression does not get fully turned on. The equation for the IIR is shown in (13) below. The time constant for this filter is altered between the attack and the decay by varying the constant k1.

$$Suppress(n) = Suppressor_{Change}(n) + k1 * Suppress(n-1) + k2 * Surpress(n-2)$$

(13) Echo Suppressor IIR Suppression Filter

After filtering Suppress will have a range from 0 < Suppress < Supp_Thresh where Supp_Thresh is the fractional part of the signal to be suppressed. This value is then subtracted from 1 and the result multiplied by L_{RES} to give the suppressed output. The equation for this suppression function is shown in (14) below.

$$AIROUT(n) = L_{RES}(n) * [1 - Suppress(n)]$$

(14) Application of Echo Suppression to Signal

5.5 DECT Echo Control Requirements

ECHOBBSP.ASM contains the following functions:

1. initec — This subroutine initialises the echo canceller and echo suppressor variables. The echo canceller is reset in the no echo state and the echo suppressor in the fully decayed state. The X register determines what echo suppressor level is initiated. See below for values.
2. chg_es — This subroutine reprograms the echo suppressor level to the value given in the X register. (0 Off, 1 3dB, 2 6dB, 3 9dB, 4 12dB, 5 15dB, 6 18dB, 7 24dB).
3. echo_c — This subroutine analyses the long-term correlation delay between the output to the line and the input from the line it detects any attempts to remove any echoes that occur within 4ms of the original output.
4. echo_s — This subroutine analyses the output speech level and when the average of the rectified output level reaches a certain level it suppresses the signal received from the line.

The following global variables are made available:

1. max_echo — This byte variable sets the maximum time of the cancelled echo in ms and is valid for values between 1 and 8.
2. ES_Marg — This word variable sets the echo suppressor Margin to 2.4dB, 3.5dB or 4.5dB, when programmed to -1,0 and 1 respectively.
3. Bck_Smp — This word variable sets the rate at which the background noise filter is updated. The filter is updated every nth frame where n is the programmed value; a value of 45 gives a half time of 1sec.
4. EC_dt — This byte variable sets the number of samples by which the output is delayed before use by the canceller; 8 samples corresponds to 1ms and the total time of this and the max_echo must be between 4 and 8ms.
5. EC_eg — This byte variable sets the gain adjustment on the input to the echo canceller for hybrid gains. It programs a dynamic shift of various echo canceller parameters. Valid values are 0, 1 and 2, corresponding to PABX gain adjustments of 0dB, 6dB and 12dB respectively.

5.6 Details of Texas Instruments Echo Controller

5.6.1 Signal Delay

The Echo Canceller will cancel echoes up to 8ms after the original output.

5.6.2 Group Delay

The group delay of the echo canceller is 125μs in each direction.

5.6.3 Minimum Echo Loss

The Echo canceller requires that the minimum echo loss from R_{out} to S_{in} is 6dB. Echoes at levels greater than these are taken as a double-talk situation. Provision is made in the echo canceller to program it to compensate for gains in the PABX between the echo canceller and the hybrid of up to 12dB.

5.6.4 Echo Canceller Update

The update of the echo canceller estimator can be suppressed by the controller. If it is not suppressed by the controller, then the estimator will be updated when speech is detected on the transmit side and not on the receive side. Transmit speech is detected when the short-term averaged signal level is greater than -30..5dBm0. Receive speech is detected when the averaged receive level is at a level greater than the averaged transmit level-6dB after PABX gain compensation.

5.6.5 Convergence Time

The echo canceller is designed to have converged onto the echo after being supplied with 500ms of suitable inputs, as specified above.

5.6.6 Artificial Echoes

Some echo-less digital systems produce artificial echoes of -24±2dBm, echoes produced by these systems will be cancelled to below -54dBm0 like ordinary network echoes.

5.6.7 Residual Echo.

After passing through a converged Echo Canceller, valid near end echoes of -13±5dB will be cancelled by more than 20dB or quantization limited to -60dBm0. Some typical performance data for the echo canceller after 125ms and 500ms of convergence with white noise is shown in the table below. (Due to spectral differences in speech signals, all echo cancellers take longer to converge with real speech than with white noise, typically the convergence time is doubled.)

R_{in}(dBm0)	$S_{in}=R_{in}$-6dB (Echo)	$S_{in}=R_{in}$-12dB (Echo)	$S_{in}=R_{in}$-18dB (Echo)	$S_{in}=R_{in}$-24dB (Echo)
0.0	**-25.8**	**-40.7**	**-42.5**	**-42.5**
-6.0	**-33.8**	**-47.2**	**-48.8**	**-49.8**
-12.0	**-42.6**	**-53.7**	**-55.1**	**-56.0**
-18.0	**-53.1**	**-59.9**	**-61.0**	**-61.2**
-24.0	**-62.1**	**-64.1**	**-63.7**	**-63.6**
-30.0	**-42.3**	**-48.4**	**-54.1**	**-59.6**

The same data produces the following performance after 500ms of convergence.

R_{in}(dBm0)	$S_{in}=R_{in}$-6dB (Echo)	$S_{in}=R_{in}$-12dB (Echo)	$S_{in}=R_{in}$-18dB (Echo)	$S_{in}=R_{in}$-24dB (Echo)
0.0	**-45.1**	**-47.2**	**-47.3**	**-47.3**
-6.0	**-51.6**	**-53.7**	**-53.9**	**-53.8**
-12.0	**-58.3**	**-59.5**	**-59.7**	**-59.5**
-18.0	**-63.1**	**-63.0**	**-62.8**	**-62.3**
-24.0	**-64.4**	**-64.1**	**-63.8**	**-63.6**
-30.0	**-54.3**	**-60.0**	**-62.2**	**-63.7**

These Figures are shown in the graphs in Figure 2 and Figure 3 below respectively for 125ms and 500ms.

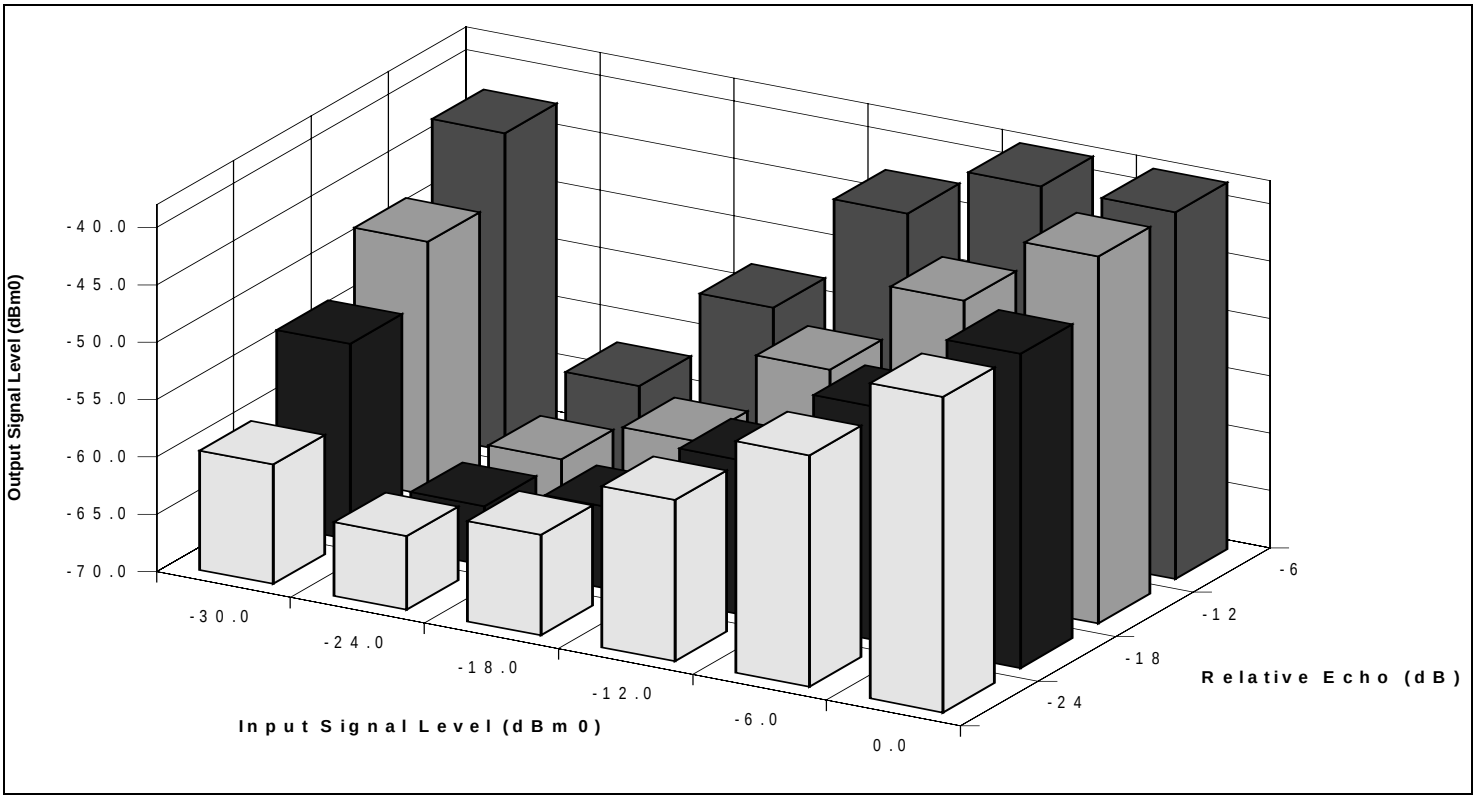


Figure 2: Residual Echo after 125ms

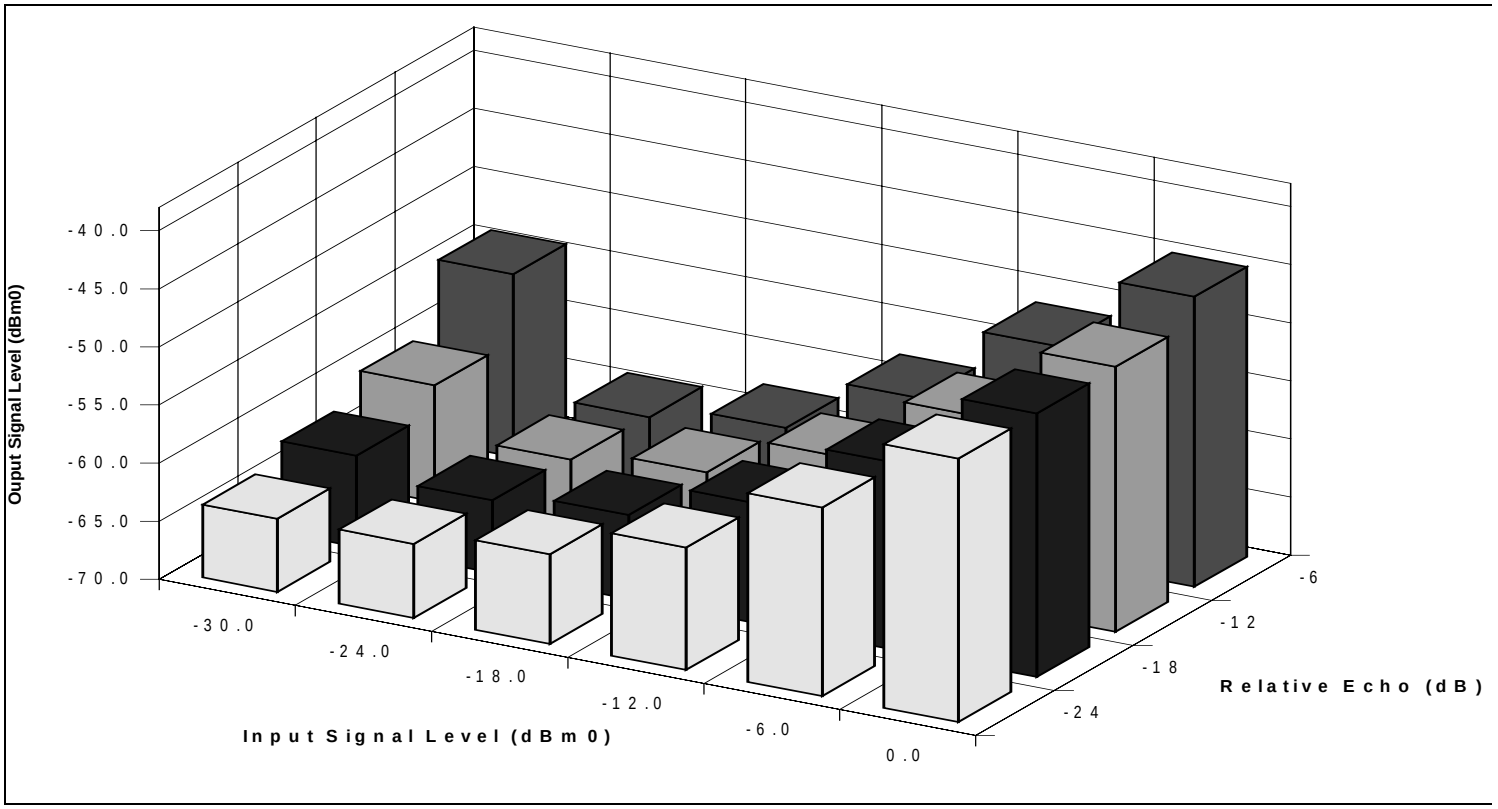


Figure 3: Residual Echo After 500ms

5.6.8 Echo Suppression

As well as echo cancellation the echo control software also contains a Non-linear processor Echo Suppressor, this gives an additional echo suppression of 0, 3, 6, 9, 12, 15, 18 or 24dB. It is recommended that the echo suppressor is set to 9dB. The performance of the Echo Suppressor with time for a sample of speech is shown in Figure 4 below. Transmit speech is detected when the short-term transmission level is more than 3.5dB (programmable) above the long-term averaged transmission level (Limited between -32.5 and -21.2dB).

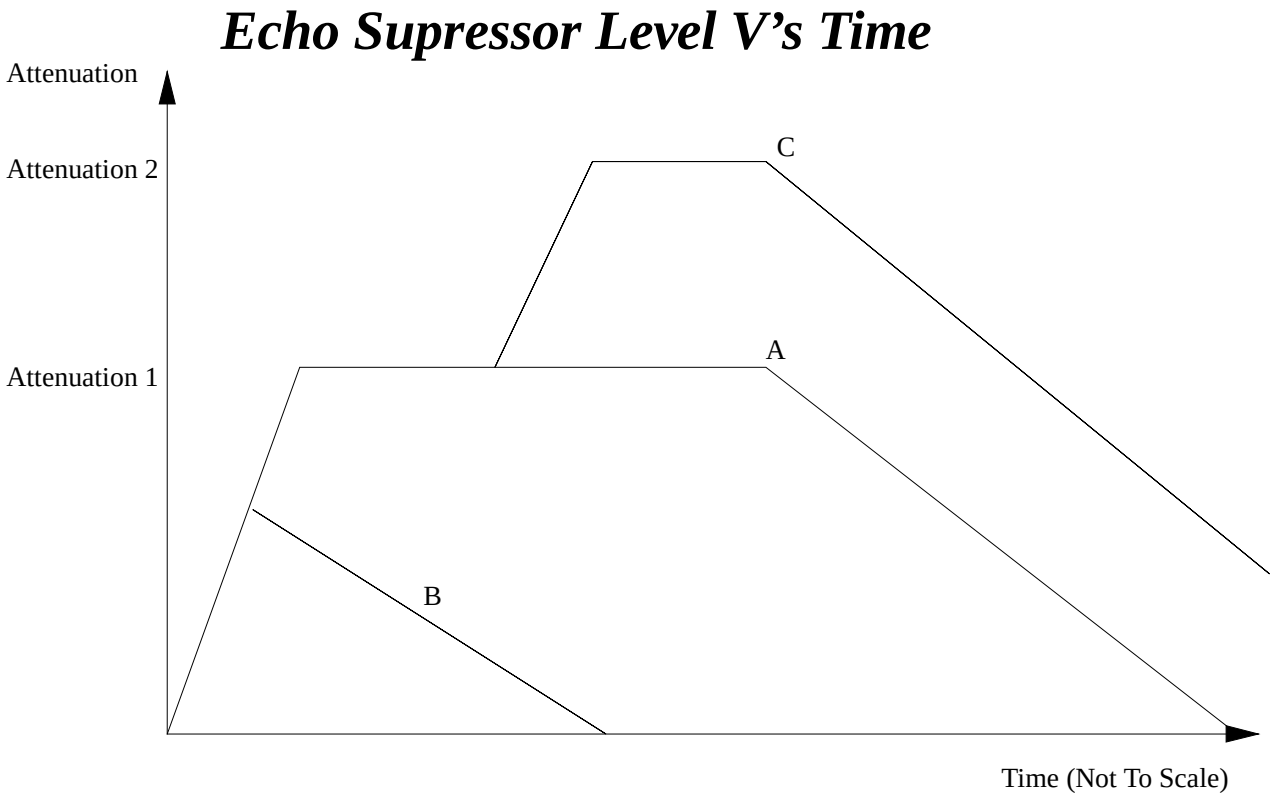


Figure 4: Echo Suppressor performance V's time for a speech sample

6. DECT Naming Conventions

The naming conventions used for signals in this document are based on those in Figure 4 of G.165.

0dBm0 corresponds to a μ-law digital level of 4096 or an A-law digital level of 2048.

7. DUAL.ASM

This module contains the reset and interrupt handling routines for the BBSP4CH in 8 ADPCM and 4 * (ADPCM + ECHO) modes. In this device the other modules ADPCM.ASM, ADPCM32.ASM, ST_DUAL.ASM and ECHOBBSP.ASM are also included in the link control file. Other ADPCM rates could be included instead for other masks. TONE.ASM is **not** supported by this interrupt control program.

It contains the following subroutines:

1) int1 — This subroutine responds to an interrupt on int1 and responds by switching the coder into 8-channel ADPCM only mode, disabling further int1's and acknowledging the interrupt.

2) int2 — This subroutine responds to interrupts on int2. These should not occur in reality but the service routine does acknowledge the interrupt and return operation to the main program so that if one occurs by accident it is less likely to crash the system

3) int3 — This is the internal serial interrupt routine. Core 0 acknowledges the interrupt to the global serial logic and all cores then perform the same function:
- Check for 4-channel or 8-channel operation. If 4-channel, switch control to the 4-channel interrupt service routine "*with_echo*", otherwise continue with 8-channel.
- read PCM/ADPCM inputs
- perform time critical ADPCM coding and decoding using "A_Cod_Dec".
- output results to serial interface.
- Perform non-time critical ADPCM coding using "code_p2".

4) with_echo — Reads the control interface for 4 * (ADPCM + Echo) commands and changes any flags required by the ADPCM, echo or flow control software, normally followed by running the main control software sub-routine "*Run_Main*". Certain commands that result in the ADPCM + Echo software not being called are implemented in this routine and these are:. (used in 4-channel mode only.)
- Reset ADPCM runs "*init_ec*" to reset the ADPCM and echo variables instead of "*Run_Main*".
- Self-test commands run the subroutine "*SELFTEST*" in ST_DUAL.ASM instead of "*Run_Main*".

5) Start — This subroutine resets the BBSP cores to their initial state on hardware reset or after self-test.

6) Run_Main — This subroutine: (used in 4-channel mode only.)
- Reads the PCM and ADPCM input information from the serial interface for the channel of this device.
- Handles the input and output cross point switches that allow the device to perform PCM and ADPCM loop-back.

- Calls the time critical ADPCM and Echo subroutines via "*Cod_Dec_Echo*"
- Writes the new PCM and ADPCM values to the serial interface for the channel of this device.
- Calls the non time critical Echo canceller update routine (if echo update is enabled) and the ADPCM routine "*code_p2*".

7) output_zero This subroutine write null values to the PCM and ADPCM output after a SW reset. (used in 4-channel mode only.)

8) Cod_Dec_Echo This subroutine performs the time critical ADPCM + Echo. (used in 4-channel mode only.)

- Call "*expand*" to convert an A-Law PCM value to linear.
- Handle mutes of the ADPCM input.
- Decode the ADPCM input
- Add sidetone (if enabled) to the PCM output.
- Compress the PCM output from linear to A-Law
- Expand the PCM output from A-Law to linear to model codec distortion in echo path.
- Call "*echo_c*" to cancel the echo and delay the data for the echo suppressor
- Choose either the cancelled or uncancelled echo depending on whether the canceller is enabled.
- Call "*echo_s*" to suppress the echo.
- Call "*code_p1*" to perform the time critical ADPCM compression

9) A_Cod_Dec Either reset or run a channel of ADPCM only on one page of data memory depending on the z flag. (used in 8-channel mode only).

10) Compress Compress's a linear PCM value to A-Law.

11) Expand Expands an A-Law value to linear.

12) TestLoop Continuously runs the self-test code for life tests. (not normally reachable)

References

1. ETSI300 175-8 DECT Specification
2. G.164 ITU Blue Book, Volume III, Fascicle III.1, Rec G.164
3. G.165 ITU Blue Book, Volume III, Fascicle III.1, Rec G.165
4. G.721/3/6 ITU Blue Book, Volume III, Fascicle III.4, Rec's G.721/3/6