



EVA Console Manager (EVCOM) Users Guide

Version: 1.1.14
Release date: 2016-05-17

© 2008 - 2016 MediaTek Inc.

This document contains information that is proprietary to MediaTek Inc.

Unauthorized reproduction or disclosure of this information in whole or in part is strictly prohibited.

Specifications are subject to change without notice.

Liability Disclaimer

Mediatek.inc may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked as reserved or undefined.

Mediatek.inc reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information. The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request. Contact your local Mediatek.inc sales office or your distributor to obtain the latest specification and before placing your product order.

Table of Contents

Table of Contents	2
1 Revision History	4
2 Introduction.....	5
2.1 Overview	5
2.2 Glossary.....	5
3 Start using EVCOM	7
4 EVCOM Command.....	11
4.1 DsplInvoke.....	11
4.2 DspRevoke	11
4.3 DspQuery.....	12
4.4 DspConfigTone.....	12
4.5 DspConfigCpt	13
4.6 DspConfigSit.....	14
4.7 ChanConfig.....	15
4.8 ChanPlayTone	16
4.9 ChanStopTone	17
4.10 ChanPlayDtmfTone	17
4.11 ChanPlayCid.....	18
4.12 ChanPlayCid2.....	20
4.13 ChanPcmDump	20
4.14 ChanPlayCidDtmf	21
4.15 StrmConfig.....	22
4.16 StrmStart.....	24
4.17 StrmStop.....	25
4.18 StrmSendDtmfr	25
4.19 StrmPlayTone	26
4.20 StrmPlayDtmf.....	27
4.21 StrmQueryMedialInfo	27
4.22 StrmResetMedialInfo	28
4.23 InfcQuery	28
4.24 InfcConfigLine.....	30
4.25 InfcConfigHook	30
4.26 InfcConfighookTs.....	31
4.27 InfcConfigHwProf.....	32
4.28 InfcConfigRing	33
4.29 InfcRing.....	34
4.30 InfcRingParams	34
4.31 InfcLineTest	35
4.32 InfcCULineTest.....	37
4.33 InfcDCFeedParams	38
4.34 PcmLoopback.....	38
4.35 RtpLoopback.....	39

4.36	DspLoopback.....	39
4.37	DspReplaceData	40
4.38	Quit	40
5	Event.....	41
5.1	Event Message.....	41
5.2	Channel Event	41
5.2.1	CID	41
5.2.2	Tone	42
5.3	Stream Event.....	42
5.3.1	T38 Event	42
5.4	Interface Event	43
5.4.1	Line.....	43
5.4.2	Hook	44
6	Test Cases.....	45
6.1	Create Two-way call	45
6.2	Create Three-way call	45
6.3	Echo cancellation Test	46
6.4	FAX Pass-Through Test.....	47
6.5	FAX Relay (T.38) Test.....	49

1 Revision History

Revision history:

Revision	Author	Date	Description
1.0.0	Quark	2011/09/25	Initial version
1.0.1	PTChen	2011/10/7	Add JB configuration, JB and T38 event
1.0.2	PTChen	2011/11/14	Add CPT configuration, SIT configuration, hookTs configuration, rtcp port configuration.
1.0.3	PTChen	2011/12/30	Add RTP Tos, RTCP Tos, Dtmf remove configuration.
1.0.4	PTChen	2012/01/06	Add CID format – CID_FORMAT_ETSI_RPAS
1.0.5	MTK04880	2012/06/08	Add PCM Dump Illustration
1.1.0	PTChen	2012/10/03	MTK VoIP version
1.1.1	wayne.lee	2014/02/27	Add InfConfigHwProf
1.1.2	jinhua.zhang	2014/05/14	Add ChanStopTone
1.1.3	lin.chen	2014/05/15	Modify StrmConfig, add plc switch
1.1.4	weifeng.xu	2014/06/13	Add ChanPlayCidDtmf
1.1.5	lin.chen	2014/06/16	Modify ChanPlayCid, add date
1.1.6	lin.chen	2014/06/17	Add PcmLoopback
1.1.7	peter.jiang	2014/11/27	Add RtpLoopback , DspLoopback
1.1.8	lin.chen	2015/04/30	Add InfcRingParams
1.1.9	yafei.ren	2015/05/12	Add InfcLineTest
1.1.10	yafei.ren	2015/06/06	Modify InfcRingParams, add ringTripThreshold
1.1.11	april.wu	2015/07/22	Add InfcDCFeedParams
1.1.12	yafei.ren	2015/11/12	Add InfcCULineTest
1.1.13	shelven.lu	2016/01/14	Add DspReplaceData
1.1.14	guanhu.zou	2016/02/19	Add StrmPlayDtmf

2 Introduction

2.1 Overview

EVCOM, Enhance VoIP Architecture Console Manager, is an application designed to manipulate DSP functionalities through ADAM (Adaptive DSP Access Manager) API and provide to user as a demo program for ADAM programming reference.

Though EVCOM was meant to be a demo application for programmer, it is also a handy and powerful tool to test DSP functionalities without any signaling application hook-up.

EVCOM is provided in source code basis, so user can trace the source code and learn how to program each function.

This document provides usage guidance from user's point of view. The EVCOM commands can be categorized into four operation objects: DSP, Channel, Stream, and Interface. EVCOM provides the commands to change the configuration or invoke certain process of specific object. See Chapter 4 for each command detail description.

2.2 Glossary

ADAM	Adaptive DSP Access Manager
Cadence	A combination of signal on and off for certain time is called a cadence.
Caller ID (CID)	A telephony signal standard to indicate subscriber (caller) identification, usually telephone number, and other information, such as user name, calling date and time. ** There is Type-1 caller ID which is known as the on-hook caller ID. The caller ID is transmitted during the telephone ringing; There is also Type-2 caller ID which is known as the "call-waiting" caller ID or the off-hook caller ID. The caller ID is transmitted during a call-waiting request (only certain countries provide Type-2 caller ID service).
CPT	Call progress tone. Telephony signals used to indicate the state of service. i.e. Dial-tone indicates a line is ready for dialing out. Busy-tone indicates a line is occupied and cannot reach its destination.
Channel	A DSP process path connecting the PCM I/O from a physical audio hardware to a network CODEC I/O.
CNG	Comfort Noise Generation. By incorporating with VAD and silence compression and generate artificial background noise to save bandwidth and improve talking experience.
CODEC	Coded/Encoded, usually implies a process of conversion between raw data and compressed (coded) data.
DAA	Data Access Arrangement. A hardware component emulate a POTS phone to provide FXO function.
DSP	Digital Signal Processor
DTMF	Dual-Tone Multi-Frequency, a telephone standard to indicate (signaling) digits.
DTMF Relay	A RFC standard (RFC2833 obsolete by RFC4733) to transmit DTMF

	information in RTP payload instead of in-band audio to provide reliable DTMF transmission.
Echo Cancellation (Echo Canceller)	A process to remove echo.
EVA	Enhanced VoIP Architecture.
FXO	Foreign Exchange Office, a telephony endpoint (Telephone) or device used to signal Central Office (CO) its request or response of a phone call.
FXS	Foreign Exchange Station, a telephony endpoint or device at Central Office (CO) side to provide signal and power for FXO.
Interface	An interface is an abstract of a physical audio hardware.
OP Code	EVCOM operation code, a short conversion of EVCOM command.
P-time / P-rate	Packetization time (rate) used to negotiate and indicate the length (ms) of the audio in each packet payload.
SAS	Subscriber Alert Signal. A signal to alert the user (telephone) a call is waiting, may be followed with type-2 caller ID.
Silence Compression	A method to save bandwidth consumption by transmit silence indication packet (SID) instead of full RTP payload when user is not talking.
SLIC	Subscriber Line Interface Circuit. A hardware component emulate CO service to provide FXS function.
Stream	Stream is a path or process to disassembly sequential coded data (i.e. audio), transmit over network, and reassembly the coded data on the far-end to restore the original information.
VAD	Voice Activity Detection. A method to assess the audio level to determine if a user is talking.
PLC	Packet Loss Concealment.

3 Start using EVCOM

First, insert DSP Kernel Modules.

****Note:** Depending on the (chipset) platform being used, the DSP kernel modules may be different or have extra module dependency. Please check the “User Guide” or “Readme” document for each platform for further detail. (Here is one example)

```
e.g. MT7525
# insmod sys_mod.ko
# insmod pcm.ko
# insmod lec.ko
# insmod slic3.ko
# insmod fxs3.ko slic1="le89156"
# insmod ksocket.ko
# insmod ortp.ko
# insmod acodec_x.ko
# insmod foip.ko
# insmod ovdsp.ko
# insmod pcmdump.ko
```

After Kernel Modules are loaded, execute EVCOM program.

```
# ./evcom
Initializing ADAM ...
ADAM Version: 0.1
Number of DSP: 1
Number of Interface: 3

EVCom >>
```

Type “dq” or “dspquery” to check current DSP initialization status and capability,

****Note:** The DSP capability varies on different (chipset) platform.

```
EVCom >> dq
DSP features:
Active status: Inactive
DSP ID: (0)
Number of Channel: 3
Stream per Channel: 2

EVCom >>
```

Type any invalid command will bring out the EVCOM command list,

```
EVCom >> ??
```

DspInvoke	DspRevoke	DspQuery	DspConfigTone
ChanConfig	ChanPlayTone	ChanPlayCid	ChanPlayCid2
ChanPcmDump	StrmConfig	StrmStart	StrmstoP
StrmSendDtmfr	StrmPlayTone	InfCQuery	InfCConfigLine
InfCConfigHook	InfCConfigRing	InfCRing	Quit

```
EVCom >>
```

Before you can issue command to control the DSP, you must do “di” or “dspinvoke” first to initialize the DSP functionalities.

```
EVCom >> dct
```

DSP was not initialized, please run DspInvoke first!

```
EVCom >>
```

Type “di” or “dspinvoke” to initialize DSP core. Once initialization completed, the DSP is ready to work under your command in EVCOM.


```

EVCom >> di
Invoking DSP ...

DSP initialization completed! You can issue command to use DSP now.

EVCom >>
== Event Once ==: Line
Interface (2) detected Line-Active-Reversed.

== Event Once ==: Hook
Interface (2) on-hooked.

== Event Once ==: Hook
Interface (1) on-hooked.

== Event Once ==: Hook
Interface (0) on-hooked.

EVCom >> dq
DSP features:
Active status: Active
DSP ID: (0)
Number of Channel: 3
Stream per Channel: 2

EVCom >>

```

To shutdown the DSP, type “dr” or “dsprevoke” to terminate DSP process.

****Note:** Once DSP is revoked, you cannot re-invoke DSP immediately. You must exit EVCOM, remove all DSP kernel modules, and start from again from inserting DSP modules.

```

EVCom >> dr
Once DSP is revoked, it cannot be invoked again until re-insert DSP modules, sure? (type 'yes' to revoke)
yes
Revoking DSP ...

DSP shutdown completed! You can quit evcom safely now.

EVCom >> di
DSP has been revoked, please exit evcom, remove and re-insert DSP modules, and start again!

EVCom >>

```

To leave EVCOM, type “q” or “quit” to exit the program.

****Note:** If DSP has not been revoked before quitting EVCOM, the exit process will revoke DSP automatically.

```
EVCom >> q
Quit evcom and shutdown DSP, sure? (type 'yes' to quit) yes
Exit evcom... goodbye!

Closing ADAM ... : Execution success!

#
```

4 EVCOM Command

4.1 DsplInvoke

Description:

Initialize and start the DSP process.

OP Code: di

Parameters & Attributes: None

Syntax:

<dspinvoke | di>

Usage example:

```
EVCom >> di
Invoking DSP ...

DSP initialization completed! You can issue command to use DSP now.

EVCom >>
```

4.2 DspRevoke

Description:

Terminate DSP process.

OP Code: dr

Parameters & Attributes: None

Syntax:

<dsprevoke | dr>

Usage example:

```
EVCom >> dr
Once DSP is revoked, it cannot be invoked again until re-insert DSP modules, sure? (type 'yes' to revoke)
yes
Revoking DSP ...

DSP shutdown completed! You can quit evcom safely now.

EVCom >>
```

4.3 DspQuery

Description:

Query DSP configuration.

OP Code: dq

Parameters & Attributes: None

Syntax:

<dspquery | dq>

Usage example:

```
EVCom >> dq
DSP features:
Active status: Active
DSP ID: (0)
Number of Channel: 3
Stream per Channel: 2

EVCom >>
```

4.4 DspConfigTone

Description:

Change DSP tone configuration.

OP Code: dct

Parameters & Attributes:

Parameters	Name	Value	Description
Tone ID	tone	1 ~ 39	Note: tone[0] is reserved for silence and cannot be configured.
Tone type	type	reg/mod	reg: regular mod: modulate (not ready yet)
Frequency [0 ~ 3]	f0 ~ f3	[frequency,power]	Frequency: 0~4000(Hz) Power: -40~0(db)
Modulation Frequency	mf	[base_frequency, modulation_frequency, modulation_power, modulation_depth]	Base frequency: 0 ~ 4000(Hz) Modulation frequency: 0 ~ 4000(Hz) Modulation power: -40 ~ 0(db) Modulation depth: 0 ~ 256
Cadence [0 ~ 2]	cad0 ~ cad2	[on_time,off_time, repeat]	On time: 0 ~ 65535(ms) Off time: 0 ~ 65535(ms) Repeat: 0 ~ 65535

Syntax:

```
< dspconfigtone | dct> tone=[tone_id] $attr0=[value] $attr1=[value] ...
```

Usage Example:

```
EVCom >> dspconfigtone tone=1 type=reg f0=350,-20 f1=440,-16 cad0=1000,0,10
```

Tone (1) configuration:

Tone type: Regular tone

Frequency(0) = 350Hz @ -20db

Frequency(1) = 440Hz @ -16db

Frequency(2) = 0Hz @ 0db

Frequency(3) = 0Hz @ 0db

Candence(0) = 1000(ms)/On, 0(ms)/Off, x 10 (times)

Candence(1) = 0(ms)/On, 0(ms)/Off, x 0 (times)

Candence(2) = 0(ms)/On, 0(ms)/Off, x 0 (times)

```
EVCom >> dct tone=2 type=mod mf=1300,25,-6,40 cad0=500,200,5
```

Tone (2) configuration:

Tone type: Modulated tone

Base frequency = 1300Hz

Modulation frequency = 25Hz

Modulation power = -6db

Modulation depth = 40

Candence(0) = 500(ms)/On, 200(ms)/Off, x 5 (times)

Candence(1) = 0(ms)/On, 0(ms)/Off, x 0 (times)

Candence(2) = 0(ms)/On, 0(ms)/Off, x 0 (times)

```
EVCom >>
```

4.5 DspConfigCpt

Description:

Change DSP call progress tone detection configuration(for FXO).

OP Code: dcc

Parameters & Attributes:

Parameters	Name	Value	Description
Cpt ID	Cpt	<dial ringback busy reorder c1 c2 c3 c4>	Note: c1~c4, customized tones.
Frequency [0 ~ 1]	f0 ~ f1	[frequency,power]	Frequency: 0~4000(Hz) Deviation: Tolerated frequency deviation from base frequency

Parameters	Name	Value	Description
Cadence [0 ~ 2]	cad0 ~ cad2	[minMake, maxMake, minBreak, maxBreak]	minMake: 0 ~ 65535(ms) maxMake: 0 ~ 65535(ms) minBreak: 0 ~ 65535(ms) maxBreak: 0 ~ 65535(ms)
Power	pwr	[power]	Power: -40~0(db)

Syntax:

<dspconfigcpt | dcc> cpt=[tone_type] \$attr0=[value] \$attr1=[value] ...

Usage Example:

```
EVCom >> dspconfigcpt cpt=busy f0=480,50 f1=620,50 cad0=450,550,450,550 pwr=-20
Cpt type (BUSY) configuration:
Frequency[0] = 480Hz, Deviation[0] = 50Hz
Frequency[1] = 620Hz, Deviation[1] = 50Hz
Power = -20db
cad[0] : minMake = 450ms, maxMake = 550ms, minBreak = 450ms, maxBreak = 550ms
cad[1] : minMake = 0ms, maxMake = 0ms, minBreak = 0ms, maxBreak = 0ms
cad[2] : minMake = 0ms, maxMake = 0ms, minBreak = 0ms, maxBreak = 0ms
EVCom >>
```

4.6 DspConfigSit

Description:

Change DSP special information tone detection configuration(for FXO).

OP Code: dcs

Parameters & Attributes:

Parameters	Name	Value	Description
Frequency [0 ~ 4]	f0 ~ f4	[frequency,power]	Frequency: 0~4000(Hz) Deviation: Tolerated frequency deviation from base frequency
Short duration	sd	[Mintime,Maxtime]	Mintime: 0 ~ 65535(ms) Maxtime: 0 ~ 65535(ms)
Long duration	ld	[Mintime,Maxtime]	Mintime: 0 ~ 65535(ms) Maxtime: 0 ~ 65535(ms)
Power	pwr	[power]	Power: -40~0(db)

Syntax:

<dspconfsit | dcs> \$attr0=[value] \$attr1=[value] ...

Usage Example:

```

EVCom >> dspconfigsit f0=914,50 f1=985,50 f2=1370,50 f3=1428,50 f4=1776,50 sd=250,300
ld=350,400 pwr=-39
SIT configuration:
Frequency[0] = 914Hz, Deviation[0] = 50Hz
Frequency[1] = 985Hz, Deviation[1] = 50Hz
Frequency[2] = 1370Hz, Deviation[2] = 50Hz
Frequency[3] = 1428Hz, Deviation[3] = 50Hz
Frequency[4] = 1776Hz, Deviation[4] = 50Hz
Power = -39db
minShortDur = 250ms, maxShortDur = 300ms
minLongDur = 350ms, maxLongDur = 400ms

EVCom >>

```

4.7 ChanConfig

Description:

Change channel configuration.

OP Code: cc

Parameters & Attributes:

Parameters	Name	Value	Description
Channel ID	ch	[0 ~ MAX_CHANNEL]	**Note: Depending on the exact channel numbers on the platform.
Detection Mask	det	dtmf[+ -], mdm[+ -], cpt[+ -], cid[+ -]	dtmf: Detect DTMF tone mdm: Detect modem (fax) tone. cpt: Detect call progress tone. (i.e. dial-tone, busy-tone, etc.) cid: Detect caller ID. (For FXO) **Use [+ -] to turn on/off the detector, i.e. dtmf+,cid-, which is to enable DTMF detector and disable caller ID detector. **Default: All On.
Echo Canceller	ec	[on off]	Enable or disable echo canceller. *Default: On
Tx Gain Amplify	tx	-20 ~ 20	Change Tx gain power between +/-20(db). *Default: 0
Rx Gain Amplify	rx	-20 ~ 20	Change Rx gain power between +/-20(db). *Default: 0

Syntax:

```
<chanconfig | cc> ch=[channel_id] $attr0=[value] $attr1=[value] ...
```

Usage Example:

```
EVCom >> chanconfig ch=1 ec=on det=mdm- tx=-3 rx=3
```

Channel (1) configuration:

Enabled Detectors:

Active : dtmf (DTMF_TONE)

Inactive : mdm (FAX/MODEM_TONE)

Active : cpt (CALL_PROGRESS_TONE)

Active : cid (Caller_ID)

EC : Enable

Tx Gain: -3db

Rx Gain: 3db

```
EVCom >>
```

4.8 ChanPlayTone

Description:

Generate a tone to the channel.

OP Code: cpt

Parameters & Attributes:

Parameters	Name	Value	Description
Channel ID	ch	[0 ~ MAX_CHANNEL]	**Note: Depending on the exact channel numbers on the platform.
Tone ID	tone	[1~39],...	Tone ID to be played. User can put several tone IDs and play in sequence, i.e tone=1,5,8,2,3
Repeat Time	rpt	0 ~ 65535	Times of repeat the tone sequence
Duration	dur	10~300000(ms)	Tone play duration

Syntax:

```
<chanplaytone | cpt> ch=[channel_id] tone=[tone_id<,tone_id,...>] rpt=[repeat] dur=[duration]
```

Usage Example:

```
EVCom >> chanplaytone ch=0 tone=1 rpt=2 dur=2000
```

Tone repeat times: 2

Playing Tone (1).

dur:2000

Generate tone(s) on channel (0), please check the telephone!

```
EVCom >>
```


4.9 ChanStopTone

Description:

Stop a tone to the channel.

OP Code: cst

Parameters & Attributes:

Parameters	Name	Value	Description
Channel ID	ch	[0 ~ MAX_CHANNEL]	**Note: Depending on the exact channel numbers on the platform.

Syntax:

<chanstoptone | cst> ch=[channel_id]

Usage Example:

```
EVCom >> cst ch=0
Stop tone(s) on channel (0), please check the telephone!

EVCom >>
```

4.10 ChanPlayDtmfTone

Description:

Generate a DTMF tone to the channel.

OP Code: cdt

Parameters & Attributes:

Parameters	Name	Value	Description
Channel ID	ch	[0 ~ MAX_CHANNEL]	**Note: Depending on the exact channel numbers on the platform.
Digit number	digit	[0 ~ 9], *, #, [A(a) ~ D(d)]	DTMF tone number to be played.
Duration	dur	0 ~ 65535 (ms)	DTMF tone duration

Syntax:

<chanplaydtmfTone | cdt> ch=[channel_id] digit=[dtmf_digit] dur=[duration]

Usage Example:

```

EVCom >> cdt ch=0 digit=1 dur=5000
DTMF tone duration: 5000
Playing DTMF tone (1).
Generate DTMF tone(s) on channel (0), please check the telephone!

EVCom >>

```

4.11 ChanPlayCid

Description:

Generate a caller ID to the channel.

****Note:** This feature is for advance user who wants to manually transmit caller ID to the telephone. However, to successfully generate caller ID to the telephone, user might need to configure the SLIC to correct state (On-hook Transmission) so the SLIC can transmit caller ID correctly.

OP Code: cpc

Parameters & Attributes:

Parameters	Name	Value	Description
Channel ID	ch	[0 ~ MAX_CHANNEL]	**Note: Depending on the exact channel numbers on the platform.
Caller ID	cid	[number],...	EVCOM only generate number
name	name	[name]	
power	cidp		
Caller ID format	cidf	<us eu jp rpass dtas>	Caller ID data format. us-BELLCORE_FSK, eu-ETSI_FSK, jp-NTT
Message Waiting Indicator	mwi	none set clear	
datetime	date	[MMddHHmm]	
T2	T2	200~500(ms)	DTMF cid timing
T3	T3	500~800(ms)	RPAS timing
T4	T4	45~500(ms)	DTAS timing

Syntax:

```

<chanplaycid | cpc> ch=[channel_id] cid=[number] name=[name] cidp=[power]
cidf=<us | eu | jp | rpass | dtas>, Note: us-BELLCORE_FSK, eu-ETSI_FSK, jp-NTT
mwi=<none | set | clear>
date=[datetime], Note:datetime format as MMddHHmm
RPAS timing: T2=200~500(ms),T3=500~800(ms)
DTAS timing: T2=200~500(ms),T4=45~500(ms)

```

Usage Example:

```

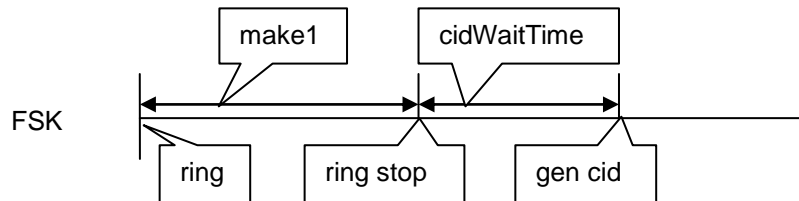
EVCCom >> chanplaycid ch=0 cid=1234 name=abcd cidf=us cidp=-10 mwi=none
Generate BELLCORE_FSK caller ID [1234] name[abcd] on channel (0), the fskower is -10 mwi=none
T2:30(10ms) T3:65(10ms) T4:30(10ms),please check the telephone!

EVCCom >>
    
```

Note: Detail description of timing.(gen is short of generate)

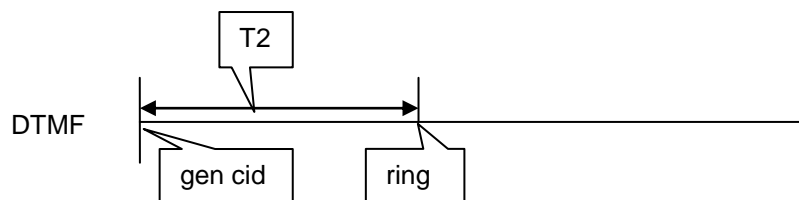
1. FSK mode

Ring first, then generate cid.



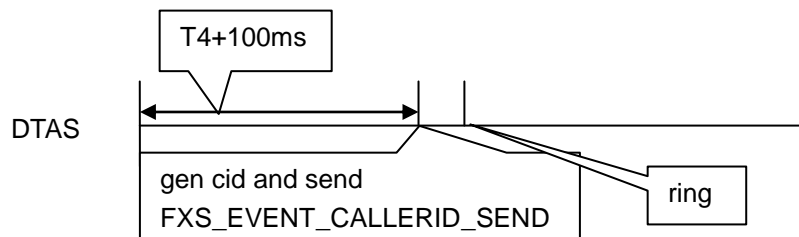
2. DTMF mode

First generate cid, after T2 timer, start ring.



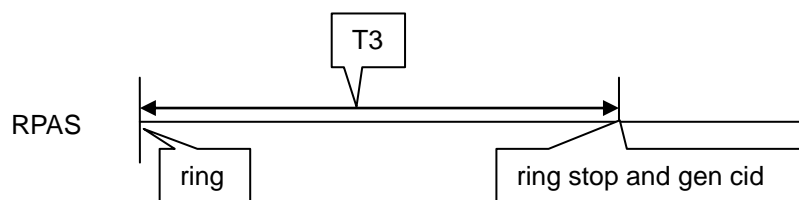
3. DTAS mode

Wait for T4+100ms, generate cid and send event FXS_EVENT_CALLERID_SEND, then start ring.



4. RPAS mode

First ring, after T3 timer, ring stop and generate cid.



4.12 ChanPlayCid2

Description:

Generate a type 2 caller ID with SAS signal to the channel.

OP Code: cpc2

Parameters & Attributes:

Parameters	Name	Value	Description
Channel ID	ch	[0 ~ MAX_CHANNEL]	**Note: Depending on the exact channel numbers on the platform.
Caller ID	cid	[number],...	EVCOM only generate number
Caller ID format	cidf	<us dtmf jp>	Caller ID data format.
power	cidp		Caller ID power.
Message Waiting Indicator	mwi	none set clear	
datetime	date	[MMddHHmm]	
T9	T9	0~15(10ms)	
T10	T10	0~15(10ms)	
T11	T11	4~5(10ms)	
T12	T12	6~20(10ms)	
T13	T13	4~12(10ms)	
T14	T14	15~16(10ms)	

note:T9+T10+T14<=32(10ms)

Syntax:

```
<chanplaycid2 | cpc2> ch=[channel_id] cid=[number] cidp=[fskpower]
    cidf=<us | eu | jp>, Note: us-BELLCORE_FSK, eu-ETSI_FSK, jp-NTT
    date=[datetime], Note:datetime format as MMddHHmm
    timing: T9=0~15(10ms) T10=0~15(10ms) T11=4~5(10ms) T12=6~20(10ms)
    T13=4~12(10ms) T14=15~16(10ms)(note:T9+T10+T14<=32(10ms))
```

Usage Example:

```
EVCom >> chanplaycid2 ch=0 cid=1234 cidp=-24 cidf=us mwi=none date=12302010 T9=5 T10=10
T11=4 T12=10 T13=8 T14=15
Generate BELLCORE_FSK caller ID [1234] name [] on channel (0), cidp=-24 mwi=none T9=5 T10=10
T11=4 T12=10 T13=8 T14=15, please check the telephone!

EVCom >>
```

4.13 ChanPcmDump

Description:

Enable PCM dump for debugging.

**** Note:** User can use this function to dump channel PCM data to designated network address (PC) and capture PCM dump log for debugging with sniffer software, such as Wireshark.

OP Code: cpd

Parameters & Attributes:

Parameters	Name	Value	Description
Channel ID	ch	[0 ~ MAX_CHANNEL]	**Note: Depending on the exact channel numbers on the platform.
IP address	ip	[IP_Address],...	IP address of destination PC to receive PCM dump log. Set ip =0.0.0.0 will disable the dump.

Syntax:

```
<chanpcmdump | cpd> ch=[channel_id] ip=[ip]
```

Usage Example:

```
EVCom >> chanpcmdump ch=0 ip=192.168.1.2
Dump channel (0) PCM to (192.168.1.2).

EVCom >>
```

4.14 ChanPlayCidDtmf

Description:

Generate a dtmf type caller ID to the channel.

OP Code: cpcd

Parameters & Attributes:

Parameters	Name	Value	Description
Channel ID	ch	[0 ~ MAX_CHANNEL]	**Note: Depending on the exact channel numbers on the platform.
Digit number	cid	[0 ~ 9], *, #, [A(a) ~ D(d)]	DTMF tone number to be played.
makeTime	make	40(ms)	DTMF call id make timer
breakTime	break	40(ms)	DTMF call id break timer
dtmfPower	dtmfp	-25~-45(dbm)	DTMF call id power
dtmfT3	dtmfT3	350(ms)	Timer between DTMF cid and the first ring

Syntax:

```
<chanplayciddtmf | cpcd> ch=[channel_id] cid=[number] make=40(ms) break=40(ms)
dtmfp=[DTMFpower] dtmfT3=350(ms)
```

Usage Example:

```
EVCom >> chanplayciddtmf ch=0 cid=1234 make=40 break=40 dtmfp=0 dtmfT3=350
Generate DTMF CID [1234] on channel (0), maketime:40 breaktime:40 dtmfp:0 dtmfT3:350, please check
the telephone!

EVCom >>
```

4.15 StrmConfig

Description:

Change a stream configuration.

****Note:** User can change a stream configuration anytime no matter if the stream is started or not and the configuration change will take effect immediately.

OP Code: sc

Parameters & Attributes:

Parameters	Name	Value	Description
Stream ID	st	[channel_id:stream_id] [0~MAX_CH:0~MAX_ST]	**Note: Channel ID and Stream ID depending on the exact channel and stream numbers on the platform.
Source Session Addresss	src	[IP_Address: rtp_port:rtcp_port]	Source IP address and rtp_port/rtcp_port used to transmit/receive RTP packet. ** Note: If not set rtcp_port, rtcp_port will use [rtp_port+1] automatically.
Destination Session Address	dst	[IP_Address: rtp_port:rtcp_port]	Destination IP address and rtp_port/rtcp_port used to transmit/receive RTP packet. ** Note: If not set rtcp_port, rtcp_port will use [rtp_port+1] automatically.
RTP Tos value	rtptos	Hexadecimal value	RTP packet tos value, using hexadecimal vaule.
RTCP Tos value	rtcptos	Hexadecimal value	RCTP packet tos value, using hexadecimal vaule.
CODEC	codec	<g711a g711u g722 g726 g729 t38>	Payload type to be used for stream transmission.
Uplink PTIME	ulPtime	<10 20 30 40 50 60>	Uplink Packetizaiton time, up to 60ms.
Downlink PTIME	dlPtime	<10 20 30 40 50 60>	Downlink Packetizaiton time, up to 60ms.
DTMF Relay	dtmfr	<on off>	Enable or disable DTMF relay.
DTMF Remove	dtmfrm	<on off>	Enable or disable DTMF remove.
Voice Activity Detection	vad	<on off>	Enable or disable Voice Activity Detection.
Comfort Noise	cng	<on off>	Enable or disable Comfort Noise

Parameters	Name	Value	Description
Generation			Generation.
Silence Compression	scomp	<on off>	Enable or disable silence compression.
Stream Direction	dir	<sr so ro in>	sr: Send-Receive so: Send-Only ro: Receive-Only in: Inactive
Packet Loss Concealment	plc	<on off>	Enable or disable Packet Loss Concealment.
Jitter Buffer	jb	[<a f>,jb_init_size, jb_max_size]	a=active, f=fixed, JB size is between 0~800 ms

Syntax:

<strmconfig | sc> st=[channel_id:stream_id] \$attr0=[value] \$attr1=[value] ...

Usage Example:

```
EVCom >> strmconfig st=0:0 src=192.168.1.1:5566:5567 dst=192.168.1.2:5566:5567 rtp tos=46
rtcp tos=46 codec=g711a ulcodec=g711a redundant_pt=102 ptime=20 dtmf=on dtmf=off vad=on
cng=on scomp=on dir=sr plc=on jb=f,500,700
Channel 0 -> Stream 0 Configuration:
Stream state: Inactive
Source address: 192.168.1.1 Rtp port : 5566 Rtcp port : 5567
Destination address: 192.168.1.2 Rtp port : 5566 Rtcp port : 5567
RTP tos: 2e RTCP tos: 2e
Codec: G.711a
red: Inactive
red_mode: 0
2833_red: Inactive
2833_red_mode: 0
redundant_pt: 102, g726_pt: 80
ulPtime: 20 dlPtime: 20
ulBitrate: Invalid bitrate
vad: Active cng: Active
Silence compression: Active
DTMF Relay: Active DTMF Remove: Inactive
Stream direction: Send & Receive
RTCP report interval: 5
Jitter Buffer Configuration: Fixed, Initial Size: 500 Max Size: 700
T38 ecm enable: Enable
T38 version: 0
T38 maxRate: 5
T38 opMode: 0
T38 packet size: 0

EVCom >>
```

4.16 StrmStart

Description:

Start a streaming process.

OP Code: ss

Parameters & Attributes:

Parameters	Name	Value	Description
Stream ID	st	[channel_id:stream_id] [0~MAX_CH:0~MAX_ST]	**Note: Channel ID and Stream ID depending on the exact channel and stream numbers on the platform.

Syntax:

```
<strmstart | ss> st=[channel_id:stream_id]
```

Usage Example:

```
EVCom >> ss st=0:0
Enable streaming process on channel (0) -> stream (0).

EVCom >>
```

4.17 StrmStop

Description:

Stop a streaming process.

OP Code: sp

Parameters & Attributes:

Parameters	Name	Value	Description
Stream ID	st	[channel_id:stream_id] [0~MAX_CH:0~MAX_ST]	**Note: Channel ID and Stream ID depending on the exact channel and stream numbers on the platform.

Syntax:

```
<strmstop | sp> st=[hannel_id:stream_id]
```

Usage Example:

```
EVCom >> sp st=0:0
Disable streaming process on channel (0) -> stream (0).

EVCom >>
```

4.18 StrmSendDtmfr

Description:

Send DTMF relay (RFC2833/4733) packet to the stream.

****Note:** The stream has to be started first before executing this command.

OP Code: ssd

Parameters & Attributes:

Parameters	Name	Value	Description
------------	------	-------	-------------

Parameters	Name	Value	Description
Stream ID	st	[channel_id:stream_id] [0~MAX_CH:0~MAX_ST]	**Note: Channel ID and Stream ID depending on the exact channel and stream numbers on the platform.
DTMF Digit	dtmf	[0 ~ 9, *, #, A, B, C, D]	DTMF digit to be send in DTMF relay packet.
Duration	dur	0 ~ 65536	DTMF duration (ms). Minimum length must be greater or equal to 10ms.

Syntax:

```
<strmsenddtmf | ssd> st=[channel_id:stream_id] dtmf=<[0~9] | * | # | [a~d]> dur=[duration(ms), dur >= 10ms]
```

Usage Example:

```
EVCom >> ss st=0:0
Enable streaming process on channel (0) -> stream (0).

EVCom >> ssd st=0:0 dtmf=5 dur=5000
Generate DTMF_5 relay packet to channel (0) -> stream (0), dur: 5000(ms) please check with sniffer!

EVCom >>
```

4.19 StrmPlayTone

Description:

Generate tone to the stream.

OP Code: spt

Parameters & Attributes:

Parameters	Name	Value	Description
Stream ID	st	[channel_id:stream_id] [0~MAX_CH:0~MAX_ST]	**Note: Channel ID and Stream ID depending on the exact channel and stream numbers on the platform.
Tone ID	tone	[0 ~ 9],...	Tone ID to be played. User can put several tone IDs and play in sequence, i.e tone=1,5,8,2,3
Repeat Time	rpt	0 ~ 65535	Times of repeat the tone sequence

Syntax:

```
<strmplaytone | spt> st=[channel_id:stream_id] tone=[tone_id<,tone_id,...>] rpt=[repeat]
```

Usage Example:

```
EVCom >> spt st=0:0 tone=1 rpt=5
Playing Tone (1).
Generate tone(s) on channel (0) -> stream (0), please check the peer!

EVCom >>
```

4.20 StrmPlayDtmf

Description:

Generate dtmf to the stream.

OP Code: spd

Parameters & Attributes:

Parameters	Name	Value	Description
Channel ID	ch	[0 ~ MAX_CHANNEL]	**Note: Depending on the exact channel numbers on the platform.
Digit number	digit	[0 ~ 9], *, #, [A(a) ~ D(d)]	DTMF tone number to be played.
Duration	dur	0 ~ 65535 (ms)	DTMF tone duration

Syntax:

<strmplaydtmf | spd> ch=[channel_id] digit=[dtmf_digit] dur=[duration]

Usage Example:

```
EVCom >> strmplaydtmf ch=0 digit=1 dur=5000
Playing DTMF Tone (1).
Generate DTMF tone on far-end direction, channel(0), please check the telephone!

EVCom >>
```

4.21 StrmQueryMediaInfo

Description:

Query current media information.

OP Code: sqm

Parameters & Attributes:

Parameters	Name	Value	Description
Stream ID	st	[channel_id:stream_id] [0~MAX_CH:0~MAX_ST]	**Note: Channel ID and Stream ID depending on the exact channel and stream numbers on the platform.

Syntax:

Syntax: <StrmQueryMediaInfo | sqm> st=[channel_id:stream_id]

Usage Example:

```
EVCom >> StrmQueryMediaInfo st=0:0
Query Media Info on channel (0) -> stream (0).
RtpError      (0).
Packet Loss    (0).
Packet recv    (574243).
Packet Loss rate(0).
maxJitter      (220).
maxRTCPInterval (15080).
bufUnderflow   (2395701).
bufOverflow    (0).

EVCom >>
```

4.22 StrmResetMediaInfo

Description:

Clean current media information to zero.

OP Code: sqm

Parameters & Attributes:

Parameters	Name	Value	Description
Stream ID	st	[channel_id:stream_id] [0~MAX_CH:0~MAX_ST]	**Note: Channel ID and Stream ID depending on the exact channel and stream numbers on the platform.

Syntax:

Syntax: <StrmResetMediaInfo | srm> st=[channel_id:stream_id]

Usage Example:

```
EVCom >> StrmResetMediaInfo st=0:0
Reset Media Info on channel (0) -> stream (0).

EVCom >>
```

4.23 InfcQuery

Description:

Query interface configuration.

OP Code: iq

Parameters & Attributes:

Parameters	Name	Value	Description
Interface ID	if	[0 ~ MAX_INTERFACE]	**Note: Depending on the exact interface numbers on the platform.

Syntax:

<infcquery | iq> if=[interface_id]

Usage Example:

```

EVCom >> iq if=0
Interface (0) configuration:
Interface type: FXS
Line State: Line-Active-Forward
Hook State: ON-HOOK
Ring configuration:
Cadence0: 500(ms)/on, 1000(ms)/off
Cadence1: 0(ms)/on, 0(ms)/off
Cadence2: 0(ms)/on, 0(ms)/off
Caller ID: number=[]
Caller ID Generation at (0)th break

EVCom >> iq if=1
Interface (1) configuration:
Interface type: FXS
Line State: Line-Active-Forward
Hook State: ON-HOOK
Ring configuration:
Cadence0: 500(ms)/on, 1000(ms)/off
Cadence1: 0(ms)/on, 0(ms)/off
Cadence2: 0(ms)/on, 0(ms)/off
Caller ID: number=[]
Caller ID Generation at (0)th break

EVCom >> iq if=2
Interface (2) configuration:
Interface type: FXO
Line State: Line-Active-Reversed
Hook State: ON-HOOK

EVCom >>

```

4.24 InfConfigLine

Description:

Change interface line state, only works for FXS interface.

OP Code: icl

Parameters & Attributes:

Parameters	Name	Value	Description
Interface ID	if	[0 ~ MAX_INTERFACE]	**Note: Depending on the exact interface numbers on the platform.
Line State	line	<down fwd rev ring sleep>	down: Power down the line fwd: Set line power active forward. rev: Set line power active reverse. ring: Set line ringing. Note: Based on interface ring configuration. sleep: Set line in low power mode. Note: Depending on the platform if the feature supported.

Syntax:

<infconfigline | icl> if=[interface_id] line=<down | fwd | rev | ring | sleep>

Usage Example:

```
EVCom >> icl if=0 line=rev
Line State: Line-Active-Reversed

EVCom >>
== Event Once ==: Line
Interface (0) detected Line-Active-Reversed

EVCom >>
```

4.25 InfConfigHook

Description:

Change interface hook state, only works for FXO interface.

OP Code: ich

Parameters & Attributes:

Parameters	Name	Value	Description
Interface ID	if	[0 ~	**Note: Depending on the exact interface

Parameters	Name	Value	Description
		MAX_INTERFACE]	numbers on the platform.
Hook State	hook	<on off flash>	on: Set On-hook off: Set Off-hook flash: Set Hook-flash

Syntax:

<infconfighook | ich> if=[interface_id] hook=<on | off | flash>

Usage Example:

```
EVCom >> ich hook=off if=2
```

```
Hook State: OFF-HOOK
```

```
EVCom >>
```

```
== Event Once ==: Hook
```

```
Interface (2) off-hooked.
```

```
== Event Once ==: Hook
```

```
Interface (0) off-hooked.
```

```
EVCom >>
```

4.26 InfcConfighookTs

Description:

Change interface hook time detection threshold, only works for FXS interface.

OP Code: ict

Parameters & Attributes:

Parameters	Name	Value	Description
Interface ID	if	[0 ~ MAX_INTERFACE]	**Note: Depending on the exact interface numbers on the platform.
Min_flash	fmin	0~65535(ms)	flashhook min time
Max_flash	fmax	0~65535(ms)	flashhook max time
Min_release	rmin	0~65535(ms)	release min time
Min_seize	smin	0~65535(ms)	seize min time
autoFlashDur	flash	0~65535(ms)	Note: Only works for FXO.
pulseGenSeize	pgs	0~65535(ms)	Note: Only works for FXO.
pulseGenRelease	pgr	0~65535(ms)	Note: Only works for FXO.
pulseDetSeizeMax	pdsmax	0~65535(ms)	
pulseDetSeizeMin	pdsmin	0~65535(ms)	
pulseDetReleaseMax	pdrmax	0~65535(ms)	

Parameters	Name	Value	Description
pulseDetReleaseMin	pdrmin	0~65535(ms)	
pulseDetEnable	pdd	on off	

Syntax:

```
<infconfighookts | ict> if=[interface_id] fmin=[Min_flash] fmax=[Max_flash] rmin=[Min_release]
smin=[Min_seize] flash=[autoFlashDur] pgs=[pulseGenSeize] pgr=[pulseGenRelease]
pdsmax=[pulseDetSeizeMax] pdsmin=[pulseDetSeizeMin] pdrmax=[pulseDetReleaseMax]
pdrbin=[pulseDetReleaseMin] pdd=[pulseDetEnable[on|off]]
```

Note: Hook release time(ms) to meet each hook state

Usage Example:

```
EVCom >> infconfighookts if=0 fmin=250 fmax=600 rmin=700 smin=300 flash=200 pgs=70 pgr=70
pdsmax=350 pdsmin=20 pdrmax=100 pdrmin=40 pdd=on
Hook Threshold: Min_flashTime:250 Max_flashTime:600 Min_releaseTime:700 autoFlashDur:200
seizeMin:300
pulse Gen Seize: 70 pulse Gen Release: 70
pulse Det Seize Max: 350 pulse Det Release Min: 20
pulse Det Release Max: 100 pulse Det Release Min: 40
pulse Det Enable:1

EVCom >>
```

4.27 InfcConfigHwProf

Description:

Change country code.

OP Code: ichp

Parameters & Attributes:

Parameters	Name	Value	Description
Interface ID	if	[0 ~ MAX_INTERFACE]	**Note: Depending on the exact interface numbers on the platform.
HwProf	HwProf	<0 1 2 3 ...>	Note: country code id

Syntax:

```
<InfcConfigHwProf | ichp> if=[interface_id] HwProf=<0| 1 | 2 | 3 |...>
```

Usage Example:


```

EVCom >> ichp if=0 HwProf=2
[DBG SLIC_Reset:L294]slic reset num:33

[DBG SLIC_Reset:L304]slic reset num:0

[DBG SLIC_Init:L587]FXS_NUM=1 FXO_NUM=0

TCSUPPORT_ZARLINK_LE89156A
slicParamReset_89156 Successful(CountryCode=Austria)
Interface (0) ,now the country code:2

EVCom >>

```

4.28 InfConfigRing

Description:

Change interface ring profile configuration, only works for FXS interface.

OP Code: icr

Parameters & Attributes:

Parameters	Name	Value	Description
Interface ID	if	[0 ~ MAX_INTERFACE]	**Note: Depending on the exact interface numbers on the platform.
Ring Cadence	cad0 ~ cad2	[on_time,off_time]	Signal on/off time (ms).
Ring Duration	dur	0 ~ 65535	(ms)
Caller ID	cid	[number]	Caller ID number.
Caller ID format	cidf	[us dtmf jp rpass]	**Note: us=BELLCORE_FSK, dtmf=ETSI_DTMF, jp=NTT, rpass=ETSI_RPAS
Caller ID At (N-th Ring-break)	cidb	0 ~ 255	Set N-th ring-break to generate caller ID.
Caller ID wait time	cidt	0 ~ 65535	Wait time to send CID after ring break
Caller ID power	cidp	0 ~ 65535	
Message Waiting Indicator	mwi	[none set clear]	

Syntax:

```
<infconfigring | icr> if=[interface_id] $attr0=[value] $attr1=[value] ...
```

Usage Example:

```

EVCom >> infcconfigring if=0 dur=500 cad0=500,1000 cad1=0,0 cad2=0,0 cidf=us cidb=0 cidt=100
cid=1234 cidp=-9
cid length = 4
mwi=none
cid maketime = 50
cid breaktime = 0
Ring configuration:
Cadence0: 500(ms)/On, 1000(ms)/Off
Cadence1: 0(ms)/On, 0(ms)/Off
Cadence2: 0(ms)/On, 0(ms)/Off
Ring duration= 500(ms)
Caller ID: number=[1234] cid power=[-18] format=[ BELLCORE_FSK] mwi=[none]
Caller ID Generation at (0)th ring break, cid wait time= 100(ms)
Dtmf type cid's maketime=50 breaktime=0 dtmfT3=350
TSK Power= -18
DTMF Power= 0
T2=:30(10ms) T3:65(10ms) T4=:30(10ms)

EVCom >>

```

4.29 InfcRing

Description:

Control interface ringing, only work for FXS type interface.

OP Code: ir

Parameters & Attributes:

Parameters	Name	Value	Description
Interface ID	if	[0 ~ MAX_INTERFACE]	**Note: Depending on the exact interface numbers on the platform.

Syntax:

<infcring | ir> if=[interface_id] dur=[duration(ms)], Note: duration is optional.

Usage Example:

```

EVCom >> ir if=0 dur=10000
Ring interface (0), check the telephone.

EVCom >>

```

4.30 InfcRingParams

Description:

Set ring parameters.

OP Code: irp

Parameters & Attributes:

Parameters	Name	Value	Description
Interface ID	if	[0 ~ MAX_INTERFACE]	**Note: Depending on the exact interface numbers on the platform.
ring type	prof	sine trap	
frequence	freq	1000~100000(Hz)	
amplitude	amp	0~85000(mV)	sum of dcBias and amplitude should below 100 Vpk!
dcBias	dcb	(mV)	sum of dcBias and amplitude should below 100 Vpk!
ringTripThreshold	rtt	0~62500(uA)	
amplitudeSlab	ampslab	47~65(V)	

Syntax:

```
<InfRingParams | irp> if=[interface_id] prof=<sine | trap> freq=[frequence(mHz)]
amp=[amplitude(mV)] dcb=[dcBias(mV)] rtt=[ringTripThreshold(uA)] ampslab=[amplitudeSlab(V)]
```

Usage Example:

```
EVCom >> InfRingParams if=0 prof=sine freq=25000 amp=45000 dcb=0 rtt=25000 ampslab=60
Ring params:
Type:0;
Frequency:25000;
Amplitude:45000;
dcBias:0;
ringTripThreshold:25000;
AmplitudeSlab:60;

EVCom >>
```

4.31 InfLineTest

Description:

Start line test.

OP Code: ilt

Parameters & Attributes:

Parameters	Name	Value	Description
Interface ID	if	[0 ~ MAX_INTERFACE]	**Note: Depending on the exact interface numbers on the platform.

Parameters	Name	Value	Description
test_items_id	testid	zarlink:1~4,7,10~16 siliconlab:1~7	**Note: Depending on the exact slic type.

Syntax:

<InfcLineTest | ilt> if=[interface_id] testid=[test_items_id,zarlink:1~4,7,10~16 siliconlab:1~7].

Usage Example:

EVCCom >> InfcLineTest if=0 testid=1			
Line Test Result - LT_TID_LINE_V			
fltMask	=	1536	
measStatus	=	0	
vAcTip	=	109mVrms	
vAcRing	=	95mVrms	
vAcDiff	=	461mVrms	
vDcTip	=	366mV	
vDcRing	=	-16266mV	
vDcDiff	=	16164mV	
EVCCom >>			

Note:

Detail description of test items:

ZARLINK		
testid	Enum member name	description
1	EVA_LT_TID_LINE_V	Line Voltage Test
2	EVA_LT_TID_ROH	Receiver Off-Hook indication
3	EVA_LT_TID_RINGERS	Ringers test per FCC Part 68 REN def
4	EVA_LT_TID_RES_FLT	Resistive Fault
7	EVA_LT_TID_CAP	Capacitance measurement test
10	EVA_LT_TID_DC_FEED_ST	DC Feed Self Test
11	EVA_LT_TID_RD_LOOP_COND	Read Loop Condition
12	EVA_LT_TID_DC_VOLTAGE	DC VOLTAGE Test
13	EVA_LT_TID_RINGING_ST	Ringing Self Test
14	EVA_LT_TID_ON_OFF_HOOK_ST	On/Off hook Self Test
15	EVA_LT_TID_RD_BAT_COND	Read battery conditions
16	EVA_LT_TID_PRE_LINE_V	Pre Line Voltage Test

SILICON		
testid	Enum member name	description
1	EVA_LT_TID_HAZARDOUS_V	Hazardous Voltages Test
2	EVA_LT_TID_FOREIGN_V	Foreign Voltages Test
3	EVA_LT_TID_RESISTIVE_FAULTS	Resistive Faults Test

SILICON		
4	EVA_LT_TID_RES_OFFHOOK	Receiver Offhook Test
5	EVA_LT_TID_REN	REN Test
6	EVA_LT_TID_CAP_FAULTS	Capacitive Faults Test
7	EVA_LT_TID_REN_CAP	REN Capacitive Test

4.32 InfCULineTest

Description:

China Unicom inside and outside line test.

OP Code: iult

Parameters & Attributes:

Parameters	Name	Value	Description
Interface ID	if	[0 ~ MAX_INTERFACE]	**Note: Depending on the exact interface numbers on the platform.

Syntax:

<InfCULineTest | iult> if=[interface_id].

Usage Example:

```

EVCom >> InfCULineTest if=0
      China Unicom inside and outside line test result amd Params:
      AGACVoltage      =      43 mVrms
      BGACVoltage      =      344 mVrms
      ABACVoltage      =      380 mVrms
      AGDCVoltage      =      373 mV
      BGDCVoltage      =      -14816 mV
      ABDCVoltage      =      14838 mV
      AGInsulationResistance =      214748364 ohm
      BGInsulationResistance =      108288 ohm
      ABInsulationResistance =      214748364 ohm
      AGCapacitance      =      27790 pF
      BGCapacitance      =      58710 pF
      ABCapacitance      =      27840 pF
      DCFeed_Voltage      =      16 V
      Ringing_Voltage      =      2147430 Vrms
      Loop_Current      =      -2147483 mA
      OutlineStatus      =      OutLine_Normal
      InlineDCVStatus      =      Inline_Test_Item_Normal
      InlineRingVStatus      =      Inline_Test_Item_Normal
      InlineCurrentStatus      =      Inline_Test_Item_Normal

```

EVCom >>

4.33 InfcDCFeedParams

Description:

Set dc feed parameters.

OP Code: idp

Parameters & Attributes:

Parameters	Name	Value	Description
Interface ID	if	[0 ~ MAX_INTERFACE]	**Note: Depending on the exact interface numbers on the platform.
current	ila	18000~40000(uA)	
current	ilaSlab	18~28(mA)	**Note: For siliconlab slic.

Syntax:

<InfcDcFeedParams | idp> if=[interface_id] ila=[current(uA)] ilaSlab=[current(mA)]

Usage Example:

```
EVCom >> InfcDcFeedParams if=0 ila=25000 ilaSlab=20
DC Feed params:
ILA:25000;
ILA SLAB:20;

EVCom >>
```

4.34 PcmLoopback

Description:

Pcm loopback test switch.

OP Code: pl

Parameters & Attributes:

Parameters	Name	Value	Description
enableControl	switch	on off	**Note: function enable control switch

Syntax:

<PcmLoopback | pl> switch=<on | off>

Usage Example:

```
EVCom >> pcmloopback switch=on

EVCom >>
```

4.35 RtpLoopback

Description:

Rtp loopback test switch.

OP Code: rl

Parameters & Attributes:

Parameters	Name	Value	Description
Channel ID	ch	[0 ~ MAX_CHANNEL]	**Note: Depending on the exact channel numbers on the platform.
enableControl	switch	on off	**Note: function enable control switch

Syntax:

<RtpLoopback | rl> ch=[channel_id] switch=<on | off>

Usage Example:

```
EVCom >> pcmloopback ch=0 switch=on
Rtp LoopBack cid 0 en 1

EVCom >>
```

4.36 DspLoopback

Description:

Dsp loopback test switch.

OP Code: dl

Parameters & Attributes:

Parameters	Name	Value	Description
Channel ID	ch	[0 ~ MAX_CHANNEL]	**Note: Depending on the exact channel numbers on the platform.
enableControl	switch	[on off]	**Note: function enable control switch

Syntax:

<DspLoopback | dl> ch=[channel_id] switch=<on | off>

Usage Example:

```
EVCom >> DspLoopback ch=0 switch=on
Dsp LoopBack cid 0 en 1

EVCom >>
```

4.37 DspReplaceData

Description:

Dsp replace data test config.

OP Code: drd

Parameters & Attributes:

Parameters	Name	Value	Description
Channel ID	ch	[0 ~ MAX_CHANNEL]	**Note: Depending on the exact channel numbers on the platform.
enableControl	enable	on off	**Note: function enable control switch
direction	dir	tx rx	
stage	stage	even value 0 2 4...16	
filepath	filepath		path of specific file(raw audio data)

Syntax:

<DspReplaceData | drd> ch=[channel_id] enable=<on | off> dir=<tx | rx> stage=<replace stage,even value 0,2,4...16> filepath=[replace file path]

Usage Example:

```
EVCom >> DspReplaceData ch=0 enable=on dir=rx stage=0 filepath=/tmp/a.pcm
```

```
EVCom >>
```

4.38 Quit

Description:

Shutdown DSP process and leave this application.

OP Code: q

Parameters & Attributes: None

Syntax:

<quit | q>

Usage example:

```
EVCom >> q
```

```
Quit evcom and shutdown DSP, sure? (type 'yes' to quit)
```


5 Event

5.1 Event Message

EVCOM reports DSP and Interface event in the below format,

[T:<timestamp>] [Event Header]:[Event Type]
[Object] [Event Description]

Examples:

[T:0000000710] == Event Once ==: Hook
Interface (0) on-hooked.

[T:0000017350] ++ Event Begin ++: Tone
Channel (1) tone[DTMF_1] detected.

[T:0000018380] -- Event End --: Tone
Channel (1) tone[DTMF_1] detected.

Description:

Field	Data	Description
T:Timestamp	0 ~ 4294967296	DSP (or CPU) ticks used to note or calculate the time of a event.
Event Header	== Event Once == ++ Event Begin ++ -- Event End --	Indicate whether it is a beginning or an end of an event, or an instant event.
Event Type	CID Tone Line Hook	Indicate the event subject.
Object	Channel Stream [Channel -> Stream] Interface	Indicate the subject object where the event is reported for.
Event Description	Caller ID Tone Line State Hook State	Caller ID / Tone: See channel event. Line/Hook State: See interface event

5.2 Channel Event

5.2.1 CID

Caller ID event, EVCOM reports a CID event when a Type-I/II caller ID is detected on a channel.

```
[T:0002617420] == Event Once ==: CID
Channel (2) detected caller ID:
Number: 123
```

5.2.2 Tone

Tone event, EVCOM reports a tone event when a DTMF or Modem/FAX tone is detected on a channel. Call progress tone (CPT) detection only works on FXO interface.

A tone event will be reported twice at the beginning and the end of detection with DSP (or CPU) ticks (ms). User can calculate how long the tone last by the ticks provided.

```
[T:0003135790] ++ Event Begin ++: Tone
Channel (1) tone[DTMF_4] detected.

[T:0003136360] -- Event End --: Tone
Channel (1) tone[DTMF_4] detected.

[T:0003268200] ++ Event Begin ++: Tone
Channel (0) tone[CED] detected.

[T:0003269580] -- Event End --: Tone
Channel (0) tone[CED] detected.
```

5.3 Stream Event

5.3.1 T38 Event

EVCOM reports a T38 event when a T38 event is detected in FAX process.

T38 State:

T38 State	Description
T38_DISCONN	Indicate the T38 stream is disconnected.

Example:

```
[T:0001800400] == Event Once ==: T38
Channel (1) stream (0): T38 Disconnected
```

5.4 Interface Event

5.4.1 Line

Line State:

Line State	Description
LINE_DOWN	Indicate a line power feed is out.
LINE_ACTIVE_FWD	Indicate a line in active state with polarity forward power feed.
LINE_ACTIVE_REV	Indicate a line in active state with polarity reverse power feed.
LINE_BUSY	Indicate a line is currently used by user (off-hook).
LINE_RING	Indicate a line is ringing.
LINE_RING_PAUSE	Indicate a line is in the break of ringing.
LINE_SLEEP	Indicate a line is in low power mode. (Depends on the hardware support)
LINE_ERROR	Indicate a line is problematic and cannot get its state.

Example:

```

EVCom >> icl line=down if=1
Line State: Line-Power-Down

EVCom >> icl line=fwd if=1
Line State: Line-Active-Forward

EVCom >>
[T:0003800400] == Event Once ==: Line
Interface (1) detected Line-Active-Forward.

EVCom >> icl line=rev if=1
Line State: Line-Active-Reversed

EVCom >>
[T:0003814310] == Event Once ==: Line
Interface (1) detected Line-Active-Reversed.

```

5.4.2 Hook

Hook State:

Hook State	Description
HOOK_SEIZE	Indicate the user has lifted the handset and reached off-hook threshold.
HOOK_RELEASE	Indicate the user has hung up the handset and reached the on-hook threshold.
HOOK_FLASH	Indicate the user has press hook flash key and meet the hook-flash threshold.
HOOK_PULSE_[1 ~ 20]	Indicate a hook pulse has met its make/break threshold.
HOOK_ERROR	Indicate an unknown hook state.

Example:

```

[T:0003886310] == Event Once ==: Hook
Interface (1) off-hooked.

[T:0005666040] == Event Once ==: Hook
Interface (1) hook-flashed.

[T:0005669090] == Event Once ==: Hook
Interface (1) on-hooked.

```

6 Test Cases

6.1 Create Two-way call

In this example, we setup a two-way stream between interface_0 (FXS) and interface_1 (FXS) to create a two-way call.

```

EVCom >> sc st=0:0 src=127.0.0.1:5000 dst=127.0.0.1:6000 dir=sr
Channel 0 -> Stream 0 Configuration:
Stream state: Active
Source address: 127.0.0.1:5000
Destination address: 127.0.0.1:6000
Codec: G.711a
Ptime: 20
Silence compression: Active
DTMF Relay: Active
Stream direction: Send & Receive

EVCom >> sc st=1:0 src=127.0.0.1:6000 dst=127.0.0.1:5000 dir=sr
Channel 1 -> Stream 0 Configuration:
Stream state: Inactive
Source address: 127.0.0.1:6000
Destination address: 127.0.0.1:5000
Codec: G.711u
Ptime: 20
Silence compression: Active
DTMF Relay: Active
Stream direction: Send & Receive

EVCom >> ss st=0:0
Enable streaming process on channel (0) -> stream (0).

EVCom >> ss st=1:0
Enable streaming process on channel (1) -> stream (0).

EVCom >>

```

6.2 Create Three-way call

Start two streams on the same channel and it will be conference automatically. User can use stream direction to control a stream on-hold.

```

EVCom >> sc st=0:0 src=192.168.1.56:5000 dst=192.168.1.200:5000 dir=sr
Channel 0 -> Stream 0 Configuration:
Stream state: Active
Source address: 192.168.1.56:5000
Destination address: 192.168.1.200:5000
Codec: G.711a
Ptime: 20
Silence compression: Active
DTMF Relay: Active
Stream direction: Send & Receive

EVCom >> sc st=0:1 src=192.168.1.56:6000 dst=192.168.1.201:5000 dir=sr
Channel 1 -> Stream 0 Configuration:
Stream state: Inactive
Source address: 192.168.1.56:6000
Destination address: 192.168.1.201:5000
Codec: G.711u
Ptime: 20
Silence compression: Active
DTMF Relay: Active
Stream direction: Send & Receive

EVCom >> ss st=0:0
Enable streaming process on channel (0) -> stream (0).

EVCom >> ss st=0:1
Enable streaming process on channel (1) -> stream (0).

EVCom >>

```

6.3 Echo cancellation Test

Create a two-way call same as 6.1. Enable and disable the echo cancellation on one channel and check the difference at the far-end.

(Do the same steps in 6.1 to create a two-way call)

.....

EVCom >> cc ch=0 ec=off

Channel (0) configuration:

Enabled Detectors:

Active : dtmf (DTMF_TONE)

Active : mdm (FAX/MODEM_TONE)

Active : cpt (CALL_PROGRESS_TONE)

Active : cid (Caller_ID)

EC : Disable

Tx Gain: 0db

Rx Gain: 0db

EVCom >> cc ch=**0** ec=**on**

Channel (**0**) configuration:

Enabled Detectors:

Active : dtmf (DTMF_TONE)

Active : mdm (FAX/MODEM_TONE)

Active : cpt (CALL_PROGRESS_TONE)

Active : cid (Caller_ID)

EC : Disable

Tx Gain: 0db

Rx Gain: 0db

6.4 FAX Pass-Through Test

Repeat the same step in 6.1 to create a two-way call first, then change the codec to g711a or g711u. Manually disable echo canceller. Connect two FAX machine to the interfaces and manually start the FAX transmission and receiving.

(Do the same steps in 6.1 to create a two-way call)

.....

```
EVCom >> sc st=0:0 codec=g711u
Channel 0 -> Stream 0 Configuration:
Stream state: Active
Source address: 127.0.0.1:5000
Destination address: 127.0.0.1:6000
Codec: G.711u
Ptime: 20
Silence compression: Active
DTMF Relay: Active
Stream direction: Send & Receive
```

```
EVCom >> sc st=1:0 codec=g711u
Channel 1 -> Stream 0 Configuration:
Stream state: Active
Source address: 127.0.0.1:6000
Destination address: 127.0.0.1:5000
Codec: G.711u
Ptime: 20
Silence compression: Active
DTMF Relay: Active
Stream direction: Send & Receive
```

```
EVCom >> cc ch=0 ec=off
Channel (0) configuration:
Enabled Detectors:
Active : dtmf (DTMF_TONE)
Active : mdm (FAX/MODEM_TONE)
Active : cpt (CALL_PROGRESS_TONE)
Active : cid (Caller_ID)
EC : Disable
Tx Gain: 0db
Rx Gain: 0db
```

```
EVCom >> cc ch=1 ec=off
Channel (1) configuration:
Enabled Detectors:
Active : dtmf (DTMF_TONE)
Active : mdm (FAX/MODEM_TONE)
Active : cpt (CALL_PROGRESS_TONE)
Active : cid (Caller_ID)
EC : Disable
Tx Gain: 0db
Rx Gain: 0db
```


6.5 FAX Relay (T.38) Test

Repeat the same step in 6.1 to create a two-way call first, then change the codec to t38. Connect two FAX machine to the interfaces and manually start the FAX transmission and receiving.

(Do the same steps in 6.1 to create a two-way call)

.....

```
EVCom >> sc st=0:0 codec=t38
```

Channel 0 -> Stream 0 Configuration:

Stream state: Active

Source address: 127.0.0.1:5000

Destination address: 127.0.0.1:6000

Codec: T.38

Ptime: 20

Silence compression: Active

DTMF Relay: Active

Stream direction: Send & Receive

```
EVCom >> sc st=1:0 codec=t38
```

Channel 1 -> Stream 0 Configuration:

Stream state: Active

Source address: 127.0.0.1:6000

Destination address: 127.0.0.1:5000

Codec: T.38

Ptime: 20

Silence compression: Active

DTMF Relay: Active

Stream direction: Send & Receive

```
EVCom >>
```