

Voice Programmer' s Guide for Windows 2000

Copyright ©2001.8.24 Seoul Commtech Corporation

Table of Contents

1. Voice Software Reference Overview	8
1.1. Voice Product Terminology	8
1.2. Organization of This Voice Reference Guide.....	9
1.3. Voice Driver	10
1.4. Voice Libraries	11
1.4.1. Single Threaded Asynchronous Programming Model	
1.4.2. Multithreaded Synchronous Programming Model	
1.4.3. Extended Asynchronous Programming Model	
2. Using the Voice Reference Library	13
2.1. Voice Library	13
2.1.1. Device Management Functions	
2.1.2. Configuration Functions	
2.1.3. I/O Functions	
2.1.4. Convenience Functions	
2.1.5. Call Status Transition Event Functions	
2.1.6. SCbus Routing Functions	
2.1.7. Global Tone Detection Functions	
2.1.8. Global Tone Generation Functions	
2.1.9. R2MF Convenience Functions	
2.1.10. Speed and Volume	
2.1.11. Speed and Volume Convenience Functions	
2.1.12. PerfectCall Call Analysis Functions	
2.1.13. Structure Clearance Functions	
2.1.14. Extended Attribute Functions	
2.2. Voice Programming Requirements.....	22
2.2.1. Opening and Using Devices	
2.2.2. Opening and Using Voice Files	
2.2.3. Busy and Idle States	
2.2.4. I/O Terminations	
2.2.5. Error Handling	
2.2.6. Voice Library Include Files	
2.2.7. Compiling Applications	
3. Voice Function Reference	29
3.1. Voice Function Reference Overview	

SCbus Functions

3.2. Voice Library Function Descriptions	29
vpm_addspddig() - sets a DTMF digit to adjust speed	30
vpm_addtone() - adds the tone	34
vpm_addvoldig() - sets a DTMF digit to immediately adjust volume	40
vpm_adjsv() - adjusts speed or volume	44
vpm_blddt() - defines a simple dual frequency tone.....	48
vpm_blddtcad() - defines a simple dual frequency cadence tone	51
vpm_bldst() - defines a simple single frequency tone.....	54
vpm_bldstcad() - defines a simple single frequency cadence tone	57
vpm_bldtngen() - sets up tone generation template	60
vpm_chgdur() - alters standard definition of duration component.....	63
vpm_chgfreq() - changes the standard definition.....	67
vpm_chgrepcnt() - changes the standard definition	71
vpm_close() - closes SCT devices.....	75
vpm_clrcap() - clears all the fields in a DX_CAP structure	77
vpm_clrdigbuf() - causes the digits present in the firmware digit buffer.....	79
vpm_clrsvcond() - clears any speed or volume adjustment conditions.....	81
vpm_clrtpt() - clears all DV_TPT fields	83
vpm_deltone() - removes all user-defined tones	86
vpm_dial() - dials an ASCII string	89
vpm_distone() - disables detection of TONE ON.....	100
vpm_enbtone() - enables detection of TONE ON	104
vpm_fileclose() - closes the file associated with the handle	108
vpm_fileopen() - opens the file specified by filep	111
vpm_fileread() - reads the file specified by filep	114
vpm_fileseek() - moves file pointer associated with handle	118
vpm_filewrite() - writes count bytes from buffer into file associated with handle.	122
vpm_getcursv() - returns the specified channel' s current speed.....	126
vpm_getdig() - initiates the collection of digits	139
vpm_getevt() - used to synchronously monitor channels.....	136
vpm_getparm() - obtains the current parameter settings.....	139
vpm_getsvmt() - returns contents of Speed or Volume Modification Table.....	142
vpm_gtcalled() – obtains the current Caller ID message.	145
vpm_gtextcalled() – obtains the current MDM message.	149
vpm_initcallp() - initializes and activates PerfectCall Call Analysis	152

vpm_open() - opens a Voice device	156
vpm_play() - plays recorded voice data.....	158
vpm_playf() - synchronously plays voice data	166
vpm_playiottdata() - plays back recorded voice data from multiple sources.....	170
vpm_playtone() - plays tone defined by TN_GEN template	174
vpm_playvox() - plays voice data stored in a single VOX file	180
vpm_playwav() - plays voice data stored in a single WAVE file	183
vpm_rec() - records voice data from a single channel	186
vpm_recf() - permits voice data to be recorded	194
vpm_reciottdata() - records voice data to multiple destinations	199
vpm_recvox() - records voice data to a single VOX file	203
vpm_recwav() - records voice data to a single WAVE file	206
vpm_setevtmsk() - enables detection of Call Status Transition (CST) event.....	208
vpm_setgtdamp() - sets up the amplitudes.....	215
vpm_sethook() - provides control of the hookswitch status.....	218
vpm_setparm() - allows you to set the physical parameters.....	223
vpm_setsvcond() - sets adjustments and adjustment conditions	226
vpm_setsvmt() - updates the speed or volume.....	231
vpm_setuio() - allows an application to install a user I/O routine	236
vpm_stopch() - forces termination of currently active I/O functions.....	240
vpm_wtcallid() - wait for specified number of rings for caller ID	244
vpm_wtring() - waits for a specified number of rings.....	247
VPMX_ANSRSIZ() - returns the duration of the answer	250
VPMX_BDNAMEP() - returns a pointer	253
VPMX_BDTYPE() - returns the device type	255
VPMX_BUFDIGS() - returns the number of uncollected digits.....	257
VPMX_CHNAMES() - returns a pointer to an array.....	260
VPMX_CHNUM() - returns the channel number	262
VPMX_CONNTYPE() - returns the connection.....	264
VPMX_CPTERM() - returns last Call Analysis termination	268
VPMX_CRTNID() - returns the tone identifier	271
VPMX_DEVTYPE() - returns device type.....	275
VPMX_DTNFAIL() - returns character for dial tone	277
VPMX_FWVER() - returns version number of D/4x firmware.....	281
VPMX_HOOKST() - returns the current hook state.....	283
VPMX_LINEST() - returns a bitmapped representation of activity	285

VPMX_LONGLOW() - returns duration of the longer silence.....	287
VPMX_PHYADDR() - returns the physical address	290
VPMX_SHORTLOW() - returns duration of shorter silence	292
VPMX_SIZEHI() - returns duration of initial non-silence.....	295
VPMX_STATE() - returns the current state	298
VPMX_TERMMSK() - returns a bitmap.....	300
VPMX_TONEID() - returns the user-defined tone id.....	304
VPMX_TRCOUNT() - returns number of bytes transferred	308

4. Voice Data Structures and Device Parameters 311

4.1. Voice Library Data Structures	
4.1.1. DV_DIGIT - <i>user digit buffer</i>	
4.1.2. DX_CAP - <i>change default call analysis parameters</i>	
4.1.3. DX_CST - <i>call status transition structure</i>	
4.1.4. DX_EBLK - <i>call status event block structure</i>	
4.1.5. DX_IOTT - <i>I/O transfer table</i>	
4.1.6. DX_SVMT - <i>speed/volume modification table structure</i>	
4.1.7. DX_SVCB - <i>speed/volume adjustment condition block</i>	
4.1.8. DX_UIO - <i>user-definable I/O structure</i>	
4.1.9. TN_GEN - <i>tone generation template structure</i>	
4.1.10. DX_XPB - <i>I/O transfer parameter block</i>	
4.2. Voice Board Parameter Defines for <code>vpw_getparm()</code>	

5. Voice Programming Conventions..... 333

5.1. Always Check Return Code in Voice Programming	
5.2. Clearing Voice Structures	
5.3. Using the Voice <code>vpw_playf()</code> and <code>vpw_recf()</code> Convenience Functions	
5.4. Using the Voice Asynchronous Programming Model	
5.5. Using Multiple Processes in Voice Synchronous Applications	

Appendix A - Standard Runtime Library 336

Voice Device Entries and Returns	
Event Management Functions	
Standard Attribute Functions	
DV_TPT Structure	
Using DX_PMOFF and DX_PMON	

Appendix B - Error Defines 346

Errors - <i>Voice Library</i>	
-------------------------------	--

Appendix C - DTMF	348
DTMF Tone Specifications	
Using MF Detection	
Appendix D - Related Voice Publications.....	349

List of Tables

Table 1. Voice Library Function Errors	27
Table 3. Valid Characters for Each Dialing Mode	90
Table 4. DX_CAP <i>Parameter Descriptions</i>	312
Table 6. Values Returned in ev_data	318
Table 7. DX_SVMT Entries.....	322
Table 8. DX_SVCB Entries	324
Table 9. TN_GEN Values	326
Table 12. Voice Device Inputs for Event Management Functions	333
Table 13. Voice Device Returns from Event Management Functions.....	333
Table 14. Standard Attribute Functions	334
Table 15. tp_length Settings	337
Table 16. tp_data Valid Values	340
Table 17. DV_TPT Fields Settings Summary	341
Table 18. Voice Library Function Errors	342

1. Voice Software Reference Overview

1.1 Voice Product Terminology

Product Naming	Guide	
CT-V04A : 4 가 Slot 가 , 가	1	
CT-V08A : 8 가 Slot 가 , 가	2	
CT-V16A : 16 가 Slot 가 , 가	2	
CT-S08A : 8 가 가 Ringer 가 , 8 가 Ring		
CT-S16A : 16 가 가 Ringer 가 , 16 가 Ring		
CT-F04A : 4 Fax Channel 가 , VPM Slot 4 FAX Dot Board		
Firmware Load File : Voice Board firmware file		download
SCbus : SCSA(Signal Computing System Architecture) voice resource TDM Bus		Board

1.2 Organization of This Voice Reference Guide

Voice Programmer's Guide For Windows 2000 Windows 2000
Voice Software Voice Driver Voice Library
instruction .

Chapter 1. Voice Software Reference Overview Voice Software
Overview . Component 가 SCT
Board , Voice Driver Voice Library .

Chapter 2. Using the Voice Reference Library Voice Library LibVpm.lib
function category
Overview Library , Programming .

Chapter 3. Voice Function Reference Voice Library
Reference .

Chapter 4. Voice Data Structure and Device Parameters

- Voice Library data structure table .
- vpm_getparm() vpm_setparm() Parameter

Chapter 5. SCT Voice Library Programming technique

Appendix A Voice Device Entry , Standard Runtime Library
return .

Appendix B Voice Library error .

Appendix C SCT Publication List .

Glossary **Index** .

Windows 2000 Voice Software	Overview	.
Voice Software		.
• Voice Driver		
• Voice Library of C functions		
• Standard Runtime Library		
Voice Driver	Voice Library	Voice Software
• Voice Software Install	demonstration program	System
	Release Software Installation Reference for Windows 2000	.
• Standard Runtime Library	Appendix A	Standard Runtime
	Library Programmer' s Guide for Windows 2000	.
• Voice Driver Features	Demo	Voice Feature
	Guide For Windows 2000	.

1.3 Voice Driver

Voice Driver	Voice hardware	, Voice hardware	.
Voice Hardware	VPM 400, CT-V08A, CT-V16A		.
VPM 400, CT-V08A, CT-V16A		Voice Processing feature	
•	Record	Play	
•	Play	Speed	Volume
•	Call		
•	Call Analysis		
•	DTMF		
•	Global Tone Generation	Detection	
User defined tone	CST event	event	detection
vpm_addtone()	vpm_enbtone()	가	.
Global Tone Detection		Voice Feature Guide	.
Voice Driver	Board	Board device	channel
channel device	board	subdevice	.
Scbus	SCSA(Signale Computing System Architecture) voice		Board

resource TDM Bus . Scbus board board
 Channel device .

SCbus SCbus routing Scbus Routing Guide SCbus
 Routing Function Reference .

1.4 Voice Library

Voice Library Voice Driver interface .Single thread
 Multi thread application Voice Library .

- LibVpm.lib – Main Voice Library
- LibSrl.lib – Standard Runtime Library
 “C” function library .
- Voice Board .
- Single Thread Synchronous Multithread Asynchronous Programming
 Model Application .
- Device .
- Device event
- Device information

LibVpm.lib voice library 2

Standard Runtime Library LibSrl.lib Standard Runtime Library Programmer's
 Guide for Windows 2000 . library device
 , SCT device common system function
 . device common event handler
 application .

1.4.1 Single Threaded Asynchronous Programming Model

thread programming thread
 voice channel .
 task 가 application .
 programming polled event management call back event
 management .

Standard Runtime Library Programmer's Guide for Windows 2000
 programming .

1.4.2 Multithreaded Synchronous Programming Model

multi thread programming model 가 application thread
block . channel
가 application . realtime
channel application .

Standard Runtime Library Programmer's Guide for Windows 2000
programming .

1.4.3 Extended Asynchronous Programming Model

sr_waitvtEx()
programming .
application device thread 가
.
Basic Asynchronous model thread
, Event sr_waitvtEx() 가 thread
.

Standard Runtime Library Programmer's Guide for Windows 2000
programming .

2. Using the Voice Reference Library

Voice library	Programming	description
• Voice library	category (section 2.1)	
• Voice Library	Programming	(section 2.2)

2.1 Voice Library

Voice library	voice device driver	interface
	category	
Device Management	• device	open close
Configuration	• device	configuration
Convenience	• play	record
Call Status Transition	• device	event
Event		
Route	• SCbus	SCbus receive(listen)
	Channel	SCbus time slot
		, F/W 가 download
		channel device transmit
		SCbus time slot
Global Tone Detection	• User define tone	detection 가
Global Tone Generation	• User define tone	generation 가
Speed and Volume	• Play Speed	Play Volume
Convenience	• Speed	Volume
		Convenience
Structure Clearance	• Data	structure
Extended Attribute	• Device	information
	Category	category
Voice Function Reference	, category 가	, header

2.1.1 Device Management Functions

vpm_close()	• close a board or channel
vpm_open()	• open a board or channel

Device Management device(channel) open close .
open, close , CT-V04A, CT-V08A, CT-V16A
Voice Device channel .
(04 :4, 08:8, 16:16)
analog channel, digital channel 가 .
open vpmBXCX B
C channel . VPM 400 channel 1-4 , VPM
800 channel 1-8 , CT-V16A channel 1-16 .
device library function , device open
. vpm_open() , device open , vpm_open()
device handle . device 가 open handle device
. vpm_close() handle device
close .
Device Management Function device busy , device 가 busy
idle .
NOTE : 1. Process device 가 , vpm_open()
vpm_close() device operation .
2. device handle define .
device handle Windows 2000 file descriptor 가 file
read(), write(), ioctl() .
3. process application device handle child
process , Device child process open

2.1.2 Configuration Functions

vpm_clr digbuf()	• library digit buffer
vpm_getparm()	• board/device parameter
vpm_sethook()	• hook switch
vpm_setparm()	• device parameter
vpm_wtring()	• ring

Configuration function open device configuration ,

Configuration function device 가 idle . Configuration

function device busy , idle

. busy idle 2.2.3 .

NOTE : vpm_sethook() I/O ,

2.1.3 I/O Functions

vpm_dial()	• ASCII String digit
vpm_getdig()	• channel digit buffer
vpm_play()	• digit
	• Play
vpm_playiottdata()	• Play
vpm_rec()	• destination
vpm_reciottdata()	• destination
vpm_stopch()	• I/O Stop

NOTE : 1. Global Tone Generation vpm_playtone() I/O function

, I/O function , I/O function attribute 가 .

2. Configuration vpm_sethook() I/O function

, I/O function , I/O function attribute 가 .

I/O function open idle channel

. I/O function data channel busy

, idle .

I/O function ,

가 error 가 , .

가
, termination
Standard Runtime Library Programmer's
Guide for Windows 2000

termination condition I/O function . (vpm_stopch())
)
condition event 가 I/O function
I/O function
VPMX_TERMMSK() . I/O termination 2.2.4

2.1.4 Convenience Functions

vpm_playf()	• file	Play
vpm_playvox()	• Vox file	Play
vpm_playwav()	• Wave file	Play
vpm_recf()	• file	
vpm_recvox()	• vox file	data
vpm_recwav()	• wave file	data

play record

vpm_playf() filename file play
vpm_play() , DX_IOTT structure file
entry
vpm_playf() DX_IOTT structure
, application open
vpm_playvox(), vpm_playwav(), vpm_recvox(), vpm_recwav(), vpm_recf()
vpm_playiottdata(), vpm_reciottdata(), vpm_rec() file

vpm_playf() vpm_recf() source code 3 Voice Function Reference

NOTE : vpm_playf(), vpm_playvox(), vpm_playwav(), vpm_recf(), vpm_recvox(),
vpm_recwav()

2.1.5 Call Status Transition Event Functions

vpm_getevt()	• Call status transition event
vpm_setevtmask()	• Call status transition event notification

Call Status Transition(CST) Event device Call
status transition event . Call status transition event call
ring , line on-hook, off-hook
tone detect .
Call status transition event list section 4.1.3 .

vpm_setevtmask() CST event detection 가 .

vpm_getevt() event .
event SRL Event Management function .

2.1.6 SCbus Routing Functions

SCbus device channel, time slot
function description Standard Runtime Library Programmer's
Guide for Windows 2000 .

2.1.7 Global Tone Detection Functions

vpm_addtone()	• user defined tone .
vpm_bliddt()	• dual frequency tone
vpm_bliddtcad()	• dual frequency cadence tone
vpm_bldst()	• single frequency tone
vpm_bldstcad()	• single frequency cadence tone
vpm_deltone()	• user-defined tone
vpm_enbtone()	• user-defined tone detection 가 .
vpm_distone()	• user-defined tone detection 2
vpm_setgtdamp ()	• Global Tone Detection amplitude .

Voice Driver single dual frequency tone
detection define Global Tone Detection .

tone DTMF tone .
 GTD vpm_blddt(), vpm_blddtcad(), vpm_bldst() vpm_addtone()
 channel tone . Global Tone Detection
 Voice Features Guide For Windows 2000 .

2.1.8 Global Tone Generation Functions

vpm_bldtngen()	• user-defined tone generation template
vpm_playtone()	• user-defined tone play

Single dual tone play Global Tone Generation

vpm_bldtngen() tone template structure TN_GEN ,
 vpm_playtone() tone .

Global Tone Generation Voice Features Guide For Windows 2000
 , TN_GEN structure 4 Voice Data Structure and
 Device Parameters .

2.1.9 Speed and Volume Functions

vpm_adjsv()	• Speed volume
vpm_clrsvcond()	• Speed volume , digit
vpm_setsvcond()	• Speed volume digit adjustment condition
vpm_getcursv()	• speed volume
vpm_getsvmt()	• Speed/Volume Modification Table
vpm_setsvmt()	• Speed/Volume Modification Table

Play speed volume . 21
 가 Speed Modification Table Volume Modification Table Channel
 . table speed volume 가
 .
 table vpm_setsvmt() default 가 .

vpm_adjsv(t vpm_setsvcond() speed volume
Modification table . vpm_adjsv() speed volume
, vpm_setsvcond() speed volume .
vpm_clrsvcond() speed volume condition .

vpm_getcursv() speed volume . vpm_getsvmt()
speed volume modification table .

Speed volume Voice Features Guide For
Windows 2000 .

2.1.10 Speed and Volume Convenience Functions

vpm_addspddig()	• Speed adjustment digit	.
vpm_addvoldig()	• Volume adjustment digit	.

vpm_addspddig() vpm_addvoldig() data structure ,
digit , digit
digit Convenience . Speed/Volume Modification
Table default setting .

2.1.11 PerfectCall Call Analysis Functions

vpm_chgdur()	• PerfectCall Call Analysis signal duration
vpm_chgfreq()	• PerfectCall Call Analysis signal frequency
vpm_chgrepcnt()	• PerfectCall Call Analysis signal repetition count
vpm_initcallp()	• channel PerfectCall Call Ana

2.1.12 Structure Clearance Functions

vpm_clrcap()	• DX_CAP structure	.
vpm_clrtpt()	• DV_TPT structure	.

device . vpm_clrcap() vpm_clrtpt()
DX_CAP DX_TPT voice library structure .

2.1.13 Extended Attribute Functions

VPMX_ANSRSIZ()	• Call Analysis	duration
VPMX_BDNAMEP()	• Device name string	pointer
VPMX_BDTYPE()	• Board type	
VPMX_BUFDIGS	• channel	vpm_getd
	library	digit buffer
	digit	
VPMX_CHNAMES()	• Channel name string	array
		point
VPMX_CHNUM()	• channel device handle	
	channel	
VPMX_CONTYPE()	• Call connection type	
VPMX_CPTERM()	• call analysis termination	
VPMX_CRTNID()	• 가 call analysis termination	
	tone	
VPMX_DEVTYPE()	• Device type	
VPMX_DTNFAIL()	• Call analysis	dial tone
VPMX_FWVER()	• firmware version	
VPMX_HOOKST()	• hook	
VPMX_LINEST()	• line	
VPMX_LONGLOW()	• duration	
VPMX_PHYADDR()	• physical address	
VPMX_LONGLOW()	• duration	
VPMX_SIZEHI()	• duration	
VPMX_STATE()	• Device	
VPMX_TERMMSK()	• Termination bitmap	
VPMX_TONEID()	• Tone id	
VPMX_TRCOUNT()	• , play	data

Voice Library Extended Attribute

Voice Device

Basic Call Analysis

VPMX_ANSRSIZ()

VPMX_CPTERM()

VPMX_LONGLOW()

VPMX_SHORTLOW()

VPMX_SIZEHI()

PerfectCall Call Analysis

VPMX_ANSRSIZ()

VPMX_CPTERM()

VPMX_CRTNID()

VPMX_DTNFAIL()

Call Status Transition event detection

VPMX_HOOKST()

Generation Tone Detection

VPMX_TONEID()

2.2 Voice Programming Requirements

Voice library

library

- Device open (section 2.2.1)
- Voice channel open (section 2.2.2)
- Busy and Idle Device State(section 2.2.3)
- I/O Terminations(section 2.2.4)
- Error Handling(section 2.2.5)
- Voice Library Include Files(section 2.2.6)
- Compiling Application(section 2.2.7)

2.2.1 Opening and Using Devices

Windows 2000 file open , file
descriptor . file descriptor .

```
int file_descriptor;
```

```
file_descriptor = open(filename, mode);
```

file action file descriptor ,
open action .

SCT board channel . device
operation device vpm_open() open
. vpm_open() channel open
channel operation SCT device handle .
chdev SCT channel device .

```
int chdev;
```

```
chdev = vpm_open(channel_name,mode);
```

channel voice library function , channel
SCT channel device handle chdev .
channel name channel open open action
handle chdev .
Board device open . bddev SCT Board
device handle .

NOTE :

board channel Windows 2000
device . open channel open
가 . driver channel board
.
가 Windows 2000 , board channel control 가
, SCT C language library .
channel board open close 3.2 Voice Library
function Description vpm_open() vpm_close() .

CAUTION : SCT device Windows 2000 open() open() .

2.2.2 Opening and Using Voice Files

Voice Library I/O routine . application Microsoft C
Runtime library , SCT Voice file
SCT file handling routine . file
open vpm_fileopen(), file close vpm_fileclose(), search
read, write vpm_fileseek(), vpm_fileread(), vpm_filewrite()
, “C” runtime function .

2.2.3 Busy and Idle States

library function 가 device .
device 가 idle , dialing
가 I/O function busy . state
busy .
idle busy .
state dependency .

2.2.4 I/O Termination

I/O function parameter termination condition
. Termination condition I/O
event . termination condition , VPMX_TERMMSK()
. I/O function
.

- byte 가 .

- vpm_stopch() device 가 stop .
- play .
- DTMF 가 detection delay 가 maximum .
- digit 가 maximum .
- non-silence maximum .
- silence maximum .
- DTMF 가 detection delay 가 maximum .
- digit .
- 가 .
- user-define tone on off .

I/O function event(digit disconnect
event) , termination . voice
application termination condition .
condition caller action voice
application .

termination condition DV_TPT structure field 가
.
I/O function . DV_TPT structure linked list array
termination condition .
srllib.h DV_TPT structure Standard Runtime Library
Programmer' s Guide for Windows 2000 .
DV_TPT voice board value Appendix A .

Byte Transfer Count – vpm_play() vpm_rec() file
play record . byte 가 DX_IOTT structure .
I/O byte 가 , 가
. DX_IOTT structure byte 4.1.5 DX_IOTT-
I/O transfer table .

Stop Occured – vpm_stopch() I/O . vpm_stopch()
vpm_stopch() .

End of file Reached – play .

DX_IOTTstructure io_length field –1 , play
 termination condition .
 , VPMX_TERMMSK() TM_EOD .

Maximum Delay Between Digits – digit
 DV_TPT tp_length field . digit
 time . ,
 VPMX_TERMMSK() TM_IDDTIME .

Maximum Digits Received – channel digit buffer
 digit . I/O buffer 가 ,
 digit .
 digit 1 31 DV_TPT structure
 tp_length . , VPMX_TERMMSK()
 TM_MAXDTMF .

Maximum Length of Non silence – Non-Silence silence 가 .
 sound .
 DV_TPT structure tp_length field
 . application non-silence , I/O
 . , VPMX_TERMMSK() TM_MAXNOSIL
 .

Maximum Length of Silence – DV_TPT
 tp_length . I/O 가 silence
 . , VPMX_TERMMSK() TM_MAXSIL
 .

Specific Digit Received – Application I/O 가 digit
 buffer digit . I/O digit buffer 가
 , application digit I/O 가
 digit DV_TPT structure
 tp_length field digit bit mask . digit buffer masking digit
 I/O , VPMX_TERMMSK()
 TM_DIGIT .

Maximum Function Time – DV_TPT structure tp_length field 100ms unit
time limit . I/O

. , VPMX_TERMMSK() TM_MAXTIME
.

User-Defined Tone On/Off Event Detected – Global Tone
Detection . user-defined tone ,

vpm_bld ... tone , vpm_addtone() vpm_enbtone()
, tone detection channel 가 . tone on/off
DV_TPT tp_termno field DX_TONE
, tp_data field DX_TONEON DX_TONEOFF .
, VPMX_TERMMSK() TM_TONE .

Application DV_TPT structure parameter
vpm_clrtpt() . DV_TPT Standard Runtime
Library Programmer's Guide for Windows 2000 , DV_TPT value
Appendix A .

2.2.5 Error Handling

SCT voice library .
0 .

Pointer Extended Attribute function 가
“Unknown Device” ASCIIZ string pointer .

-1 .

가 ATDV_ERRMSGP() .

Standard Runtime Library Programmer's Guide for Windows

2000 .

Voice library error Table1 Appendix B

NOTE: 1. vpm_open vpm_close() error
, error() error.h errno

2. ATDV_LASTERR() 가 EDX_SYSTEM
System Error 가 .

Window NT

Table 1. Voice Library Function Errors

Error Define	Error String
EDX_AMPLGEN	Invalid Amplitude Value in Tone Generation Template [+2dB - -40dB]
EDX_ASCII	Invalid ASCII Value in Tone Template Description
EDX_BADDEV	Invalid Device Descriptor
EDX_BADIOTT	Invalid Entry in the <i>DX_IOTT</i>
EDX_BADPARM	Invalid Parameter in Function Call
EDX_BADPROD	Function Not Supported on this Board
EDX_BADTPT	Invalid Entry in the <i>DX_TPT</i>
EDX_BADWAVFILE	Invalid WAV file
EDX_BUSY	Device is Already Busy
EDX_CADENCE	Invalid Cadence Component Values in Tone Template Description
EDX_CHANNUM	Invalid Channel Number Specified
EDX_DIGTYPE	Invalid Dig_type Value in Tone Template Description
EDX_FLAGGEN	Invalid tn_dflag field in Tone Generation Template
EDX_FREQDET	Invalid Freq Component Values in Tone Template Description
EDX_FREQGEN	Invalid Frequency Component in Tone Generation Template [200hz - 2000hz]
EDX_FWERROR	Firmware Error
EDX_IDLE	Device is Idle
Error Define	Error String
EDX_INVSUBCM	Invalid Sub Command Number
EDX_MAXTMPLT	Max number of Templates Exists
EDX_MSGSTATUS	Invalid Message Status Setting

3.VPM Function Reference

3.1 VPM Function Reference Overview

VPM board	VPM Function
VPM Function	Device Management Function, Configuration Function, I/O Function, Convenience Function, Global Tone Detection Function, Global Tone Generation Function, Speed and Volume Function, Speed and Volume Convenience Function, Smart Call Analysis Function, Structure Clearance Function, Extended Attribute Function
	Structure , Structure
dxxxlib.h srllib.h dxdigit.h	.
DV_DIGIT	: User Digit Buffer Structure
DV_TPT	: Termination Parameter Table Structure
DX_CAP	: Call Analysis Parameter Structure
<i>DX_EBLK</i>	: Event Block Structure
<i>DX_IOTT</i>	: I/O Transfer Table Structure
<i>DX_UIO</i>	: I/O User-definable I/O Structure
	.

3.2 VPM Function Description

vpm_addspddig()

sets a DTMF digit to adjust speed

Name : int vpm_addspddig(chdev,digit,adjval)

Inputs :

int chdev : VPM

char digit : DTMF digit

short adjval : speed adjustment value

Returns :

0 :

-1 :

Includes: srllib.h, smartsdk.h

Category: Speed and Volume Convenience



vpm_addspddig() play DTMF digit
speed , chdev ,

NOTE : 1. , speed , adjval
 speed value 가 speed
 vpm_clrsvcond() .

2. speed control VPM 4CH, VPM 8CH, VPM 16CH .

 vpm_setsvmt() Speed Modification Table
가 .

Parameter .

Chdev : vpm_open

Digit : adjval speed speed
 DTMF digit(0-9,*,#)

adjval : .

SV_ADD10PCT : 10% 가

SV_ADD20PCT : 20% 가

SV_ADD30PCT : 30% 가

SV_ADD40PCT :	40%	가
SV_ADD50PCT :	50%	가
SV_SUB10PCT :	10%	
SV_SUB20PCT :	20%	
SV_SUB30PCT :	30%	
SV_SUB40PCT :	40%	
SV_NORMAL :		, DTMF digit
	null	.



1. 가 . reset
 , reset . (DTMF
 digit “ 1” 가 , ,
 , DTMF digit “ 1” ,
 condition array . DTMF digit “ 1” .)
2. digit digit buffer ,
 vpm_getdig() VPMX_BUFDIGS() .
3. digit termination ,
 digit .
4. VPM 4, VPM 8, VPM 16 .

■ Example

```
#include <stdio.h>
#include <errno.h>
#include <srlib.h>
#include <smartsdk.h>
#include <windows.h>
/*
 * General Variables
 */
main()
{
int vpmdev;
/*
 * Open the Voice Channel Device and Enable a Handler
 */
```



```

if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
    perror( "vpmB1C1" );
    exit( 1 );
}
/*
* Add a Speed Adjustment Condition - increase the
* playback speed by 30% whenever DTMF key 1 is pressed.
*/
if ( vpm_addspddig(vpmdev, '1', SV_ADD30PCT ) == -1 ) {
    printf("Unable to Add a Speed Adjustment Condition\n");
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSG( vpmdev ) );
    vpm_close( vpmdev );
    exit( 1 );
}
/*
* Continue Processing
* .
* .
* .
*/
/*
* Close the opened Voice Channel Device
*/
if ( vpm_close( vpmdev ) != 0 ) {
    perror( "close" );
}
/* Terminate the Program */
exit( 0 );
}

```

Error

가 -1 , ATDV_LASTERR() ATDV_ERRMSG()
Error .

EINVAL : Invalid Argument

EBADF : Invalid file descriptor

EINTR : A signal was caught



- **vpm_addvoldig()**
- **vpm_adjsv()**
- **vpm_clrsvcond()**
- **vpm_getcursv()**
- **vpm_getsvmt()**
- **vpm_setsvcond()**
- **vpm_setsvmt()**
- **"Speed and Volume Modification Tables" (*Voice Features Guide for Windows 2000*)**
- ***DX_SVCB* data structure (*Chapter 4. Voice Data Structures and Device Parameters*)**

vpm_addtone()adds the tone

Name : int vpm_addtone(chdev,digit,adjval)

Inputs :

int chdev : VPM

unsigned char digit : bound tone optional digit

unsigned char digtype : digit type

Returns :

0 :

-1 :

Includes: srllib.h, smartsdk.h

Category: Global Tone Detection

■

vpm_addtone() 가 Global Tone Detection build tone) tone channel 가 tone vpm_distone() tone , detection , vpm_enbtone() tone detection . tone-on events ,tone off event vpm_distone() .

■ **Detection Notification**

Tone-on Tone-off event call status transition event . event asynchronous application synchronous application

Synchronous	Asynchronous
1. vpm_addtone() vpm_enbtone() 2. CST event vpm_getevt Event DX_EBLK structure	1. tone-on /off event detection , vpm_addtone(), vpm_enbtone() 2. TDX_CST event , SRL 3. DX_CST structure Sr_getevtdatap()

NOTE: vpm_addtone() vpm_enbtone() 가
 vpm_setevtmsk() , event detection 가
 , CST event .

■ **User-Defined Tones Termination** .

User-defined tone detection I/O Function termination
 . DV_TPT structure tp_termno filed DX_TONE
 , tp_data filed DX_TONEON, DX_TONEOFF .

Parameter .

Chdev : vpm_open
Digit : tone digit . t one
 , Digit DV_DIGIT digit buffer . digit
 vpm_getdigit() .(ie., digit
 DTMF)
 digit , tone DE_TONEON
 event DE_TONEOFF event .
digtype : channel digit type 가 .
 . **DG_USER1**
 . **DG_USER2**
 . **DG_USER3**
 . **DG_USER4**
 . **DG_USER5**
 type 20 digit .

-
1. channel tone vpm_addtone()
 vpm_blddt() build tone .
 , Error가 .
 2. vpm_addtone() tone .
 3. channel tone vpm_initcallp()
 8 가 , vpm_initcallp() 3 가 .

■ **Example**


```

#include <stdio.h>
#include <errno.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
#define TID_1 101
#define TID_2 102
#define TID_3 103
#define TID_4 104
main()
{
int vpmdev;
/*
* Open the Voice Channel Device and Enable a Handler
*/
if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
    perror( "vpmB1C1" );
    exit( 1 );
}
/*
* Describe a Simple Dual Tone Frequency Tone of 950-
* 1050 Hz and 475-525 Hz using leading edge detection.
*/
if ( vpm_blddt( TID_1, 1000, 50, 500, 25, TN_LEADING ) == -1 ) {
    printf( "Unable to build a Dual Tone Template\n" );
}
/*
* Bind the Tone to the Channel
*/
if ( vpm_addtone( vpmdev, NULL, 0 ) == -1 ) {
    printf( "Unable to Bind the Tone %d\n", TID_1 );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSG( vpmdev ));
    vpm_close( vpmdev );
    exit( 1 );
}
}

```



```

/*
* Describe a Dual Tone Frequency Tone of 950-1050 Hz
* and 475-525 Hz. On between 190-210 msec and off
* 990-1010 msec and a cadence of 3.
*/
if ( vpm_blddtcad( TID_2, 1000, 50, 500, 25, 20, 1, 100, 1, 3 ) == -1 ) {
    printf("Unable to build a Dual Tone Cadence Template\n" );
}
/*
* Bind the Tone to the Channel
*/
if ( vpm_addtone( vpmdev, ' A' , DG_USER1 ) == -1 ) {
    printf( "Unable to Bind the Tone %d\n", TID_2 );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSG( vpmdev ) );
    vpm_close( vpmdev );
    exit( 1 );
}
/*
* Describe a Simple Single Tone Frequency Tone of
* 950-1050 Hz using trailing edge detection.
*/
if ( vpm_bldst( TID_3, 1000, 50, TN_TRAILING ) == -1 ) {
    printf( "Unable to build a Single Tone Template\n" );
}
/*
* Bind the Tone to the Channel
*/
if ( vpm_addtone( vpmdev, ' D' , DG_USER2 ) == -1 ) {
    printf( "Unable to Bind the Tone %d\n", TID_3 );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSG( vpmdev ) );
    vpm_close( vpmdev );
    exit( 1 );
}
/*

```



```

* Describe a Single Tone Frequency Tone of 950-1050 Hz.
* On between 190-210 msec and off 990-1010 msec and
* a cadence of 3.
*/
if ( vpm_bldstcad( TID_4, 1000, 50, 20, 1, 100, 1, 3 ) == -1 ) {
    printf("Unable to build a Single Tone Cadence Template\n");
}
/*
* Bind the Tone to the Channel
*/
if ( vpm_addtone( vpmdev, NULL, 0 ) == -1 ) {
    printf( "Unable to Bind the Tone %d\n", TID_4 );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSG( vpmdev ) );
    vpm_close( vpmdev );
    exit( 1 );
}
/*
* Continue Processing
*
*
*
*/
/*
* Close the opened Voice Channel Device
*/
if ( vpm_close( vpmdev ) != 0 ) {
    perror( "close" );
}
/* Terminate the Program */
exit( 0 );
}

```


■ Error

가 -1, ATDV_LASTERR(), ATDV_ERRMSGP()

Error .

EDX_ASCII : Invalid ASCII value in tone template description
EDX_BADPARAM : Invalid parameter
EDX_CADENCE : Invalid cadence component value
EDX_DIGTYPE : Invalid Dig_Type value in tone template description
EDX_FREQDET : Invalid tone frequency
EDX_INVSUBCMD : Invalid sub-command
EDX_MAXTMPLT : Maximum number of user-defined tones for the
board
EDX_SYSTEM : Windows 2000 System error - check **errno**
EDX_TONEID : Invalid tone template ID



“Global Tone Detection” functions:

- vpm_blddt(), vpm_bldst(), vpm_blddtcad(), vpm_bldstcad()
- vpm_distone()
- vpm_enbtone()

"Global Tone Detection" Event Retrieval:

- vpm_getevt()
- **DX_CST** data structure
- sr_getevtdatap() (in the *Standard Runtime Library Programmer's Guide for Windows NT*)

Digit Retrieval:

- vpm_getdig()
- **DV_DIGIT**

vpm_advoldig() set ad DTMF digit to immediately adjust volume

Name : int vpm_addvoldig(chdev,digit,adjval)

Inputs :

int chdev : VPM
char digit : DTMF digit
short adjval : volume ajustment value

Returns :

0 :
-1 :

Includes: srllib.h, smartsdk.h

Category: Speed and Volume Convenience



vpm_addvoldig() channel play , volume
convenience function .

NOTE : 1. , volume , adjval
volume value 가 . volume
vpm_clrsvcond()
.
2. volume control VPM 4CH, VPM 8CH, VPM 16CH .
vpm_setsvmt() Volume Modification Table
가 .
vpm_setsvmt() Speed Modification Table
가 . speed volume function Speed, Volume
Modification Table , VPM feature Guide for Windows 2000
.

Parameter .

Chdev : vpm_open

Digit : adjval volume volume
DTMF digit(0-9,*,#)

adjval : volume .

SV_ADD2DB	: play volume	2DB	가
SV_ADD4DB	: play volume	4DB	가
SV_ADD6DB	: play volume	6DB	가
SV_ADD8DB	: play volume	8DB	가
SV_SUB2DB	: play volume	2DB	
SV_SUB4DB	: play volume	4DB	
SV_SUB6DB	: play volume	6DB	
SV_SUB8DB	: play volume	8DB	
SV_NORMAL	: volume		, DTMF digit
	null	.	



- 가 . reset
, reset . (DTMF
digit “1” 가 , ,
, DTMF digit “1” ,
condition array . DTMF digit “1” .)
- Volume digit digit buffer ,
vpm_getdig() VPMX_BUFDIGS() .
- Volume digit termination ,
digit .
- volume VPM 4, VPM 8, VPM 16 .

■ Example

```
#include <stdio.h>
#include <errno.h>
#include <srllib.h>/*
#include <smartsdk.h>
#include <windows.h>
/*
* General Variables
*/
main()
{
int vpmdev;
/*
```



```

* Open the Voice Channel Device and Enable a Handler
*/
if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
    perror( "vpmB1C1" );
    exit( 1 );
}
/*
* Add a Speed Adjustment Condition - decrease the
* playback volume by 2dB whenever DTMF key 2 is pressed. */
if ( vpm_addvoldig( vpmdev, ' 2' , SV_SUB2DB ) == -1 ) {
    printf( "Unable to Add a Volume Adjustment" );
    printf( " Condition\n");
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSGP( vpmdev ) );
    vpm_close( vpmdev );
    exit( 1 );
}
/*
* Continue Processing
* .
* .
* .
*/
/*
* Close the opened Voice Channel Device
*/
if ( vpm_close( vpmdev ) != 0 ) {
    perror( "close" );
}
/* Terminate the Program */
exit( 0 );
}

```

Error

가 -1 , ATDV_LASTERR() ATDV_ERRMSGP()
 Error .

EDX_BADPARG: Invalid Parameter

EDX_BADPROD : Function not supported on this board

EDX_SVADJBLKS : Invalid Number of Play Adjustment Blocks

EDX_SYSTEM : Windows 2000 system error - check **errno**



Related Speed and Volume functions:

vpm_addspddig()

vpm_adjsv()

vpm_clrsvcond()

vpm_getcursv()

vpm_getsvmt()

vpm_setsvcond()

vpm_setsvmt()

vpm_adjsv()

adjusts speeds or volume immediately

Name : int vpm_adjsv(chdev,tabletype,action,adjsize)

Inputs :

int chdev : VPM

unsigned short tabletype : table(speed or volume)

unsigned short action : speed volume
()

Returns :

0 :

-1 :

Includes: srllib.h, smartsdk.h

Category: Speed and Volume



vpm_adjsv() channel Play Speed Volume
, channel device handle ,
. Speed Volume
가 . action parameter
vpm_addjsv() play speed volume , speed
volume modification table . table speed volume
21 가 , speed volume 10 ,
10 가 . table vpm_setsvmt()
default values 가 .
table “Voice Features Guide for Windows 2000
.

NOTE : 1. vpm_setsvcond() . play speed
volume vpm_addjsv() ,
, vpm_setsvcond() .
2. play , speed volume 가
.

Parameter

Chdev : vpm_open

tabletype : speed volume table 가

SV_SPEEDTBL : Speed Modification Table

SV_VOLUMETBL : Volume Modification Table

action : type

SV_ABSPOS : table adjsize parameter position

SV_RELCURPOS : adjsize parameter step speed
volume .

SV_TOGGLE : adjsize parameter speed volume

adjsize : size . adjize action 가 .
adjsize .

Action	가	adjsize value
SV_ABSPOS	speed,volume modification table -10 - 10 가 , position 0 .	
SV_RELCURPOS	speed, volume modification table 가 step . speed volume -2 , 2step	

■ Example

```
#include <stdio.h>
#include <errno.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
```



```

int vpmdev;
/*
* Open the Voice Channel Device and Enable a Handler
*/
if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
    perror( "vpmB1C1" );
    exit( 1 );
}
/*
* Modify the Volume of the playback so that it is 4dB
* higher than normal.
*/
if ( vpm_adjsv( vpmdev, SV_VOLUMETBL, SV_ABSPOS, SV_ADD4DB ) == -1 ) {
    printf( "Unable to Increase Volume by 4dB\n" );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSGP( vpmdev ) );
    vpm_close( vpmdev );
    exit( 1 );
}
/*
* Continue Processing
* .
* .
* .
*/
/*
* Close the opened Voice Channel Device
*/
if ( vpm_close( vpmdev ) != 0 ) {
    perror( "close" );
}
/* Terminate the Program */
exit( 0 );
}

```

Error

가 -1 , ATDV_LASTERR() ATDV_ERRMSGP()
Error .

EDX_BADPARM: Invalid Parameter

EDX_BADPROD : Function not supported on this board

EDX_SYSTEM : Windows 2000 system error - check **errno**



Related Speed and Volume functions:

vpm_setsvmt()

vpm_getcursv()

vpm_getsvmt()

"Speed and Volume Modification Tables"(Voice Feature Guide for Windows 2000)

DX_SVMT structure(Chapter 4 Voice Data Structure and Device Parameters)

Setting Speed and Volume conditions

vpm_setsvcond()

vpm_clrsvcond()

vpm_blddt() defines a simple dual frequency tone

Name : int vpm_blddt(tid,freq1,fq1dev,freq2,fq2dev,mode)

Inputs :

- unsigned int tid : tone ID to assign
- unsigned int freq1 : frequency 1 in Hz
- unsigned int fq1dev : frequency 1 Deviation in Hz
- unsigned int freq2 : frequency 2 in Hz
- unsigned int fq2dev : frequency 2 Deviation in Hz
- unsigned int mode : leading or trailing edge

Returns :

- 0 :
- 1 :

Includes: srllib.h, smartsdk.h

Category: Global ToneDetection



vpm_blddt() Simple Dual Frequency tone . vpm_addtone() 가
, tone tone detection .
vpm_blddt() tone . Tone
vpm_addtone() 가
tone detection vpm_addtone .

Parameter .

tid : tone tone ID 100 – 150 가 .
freq1 : tone frequency
frq1dev : tone frequency deviation
freq2 : tone frequency
frq2dev : tone frequency deviation
mode : tone detection leading edge trailing edge
define 가 .
- TN_LEADING
- TN_TRAILING



```

1. vpm_initcallp()          3          tone          가 ,
    8          가          가 .
    vpm_blddt()          vpm_addtone()          . tone
    vpm_addtone()          가          create          , vpm_blddt()
    가          ,          tone          tone
    .          , vpm_blddt()          , vpm_addtone()
    error 가          .

```

■ Example

```

#include <stdio.h>
#include <errno.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
#define TID_1 101
main()
{
    int vpmdev;
    /*
    * Open the Voice Channel Device and Enable a Handler
    */
    if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
        perror( "vpmB1C1" );
        exit( 1 );
    }
    /*
    * Describe a Simple Dual Tone Frequency Tone of 950-
    * 1050 Hz and 475-525 Hz using leading edge detection.
    */
    if ( vpm_blddt( TID_1, 1000, 50, 500, 25, TN_LEADING ) == -1 ) {
        printf( "Unable to build a Dual Tone Template\n" );
    }
    /*
    * Continue Processing

```



```

*
*
*
*/
/*
● Close the opened Voice Channel Device
*/
if ( vpm_close( vpmdev ) != 0 ) {
    perror( "close" );
}
/* Terminate the Program */
exit( 0 );
}

```

■ Error



Global Tone Detection :

- "Global Tone Detection"(Voice Features Guide for Windows 2000)

Building Tones :

- vpm_bldst()
- vpm_blddtcad()
- vpm_bldstcad()
- Enabling Tone Detection
 - vpm_addtone()
 - vpm_distone()
 - vpm_enbtone()

vpm_blddtcad() defines a simple dual frequency cadence tone

Name : int vpm_blddtcad(tid,freq1,fq1dev,freq2,fq2dev,
ontime,ontdev,offtime,offdev,mode)

Inputs :

unsigned int tid : tone ID to assign
 unsigned int freq1 : frequency 1 in Hz
 unsigned int fq1dev : frequency 1 Deviation in Hz
 unsigned int freq2 : frequency 2 in Hz
 unsigned int fq2dev : frequency 2 Deviation in Hz
 unsigned int ontime : tone-on time(10ms unit)
 unsigned int ondev : tone-on time deviation(10ms unit)
 unsigned int offtime : tone-off time(10ms unit)
 unsigned int offdev : tone-off time deviation(10ms unit)
 unsigned int repcnt : cadence갓 ,

Returns :

0 :
 -1 :

Includes: srlib.h, smartsdk.h

Category: Global ToneDetection



vpm_blddtcad() Simple Dual Frequency Cadence tone .
 vpm_addtone() 갓 vpm_addtone()
 vpm_blddtcad() tone tone()
 . vpm_blddtcad() tone .
 Tone vpm_addtone() 갓
 tone detection vpm_addtone .

Parameter .

tid : tone tone ID 100 – 150 갓 .
freq1 : tone frequency
frq1dev : tone frequency deviation

freq2 : tone frequency
frq2dev : tone frequency deviation
ontime : Cadence가 ON Time Length(10ms unit)
ontdev : Ontime deviation
offtime : Cadence가 OFF Time Length(10ms unit)
offtdev : Offtime deviation
repcnt : cadence (, signal on off)



1. vpm_initcallp() 3 tone 가 ,
 8 가 가 .
 vpm_blddt() vpm_addtone() . tone
 vpm_addtone() 가 create , vpm_blddt()
 가 , tone tone
 . , vpm_blddt() , vpm_addtone()
 error 가 .

■ Example

```

#include <stdio.h>
#include <errno.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
#define TID_2 102
main()
{
  int vpmdev;
  /*
  * Open the Voice Channel Device and Enable a Handler
  */
  if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
    perror( "vpmB1C1" );
    exit( 1 );
  }
  /*
  
```



```

* Describe a Dual Tone Frequency Tone of 950-1050 Hz
* and 475-525 Hz. On between 190-210 msec and off
* 990-1010 msec and a cadence of 3.
*/
if ( vpm_blddtcad( TID_2, 1000, 50, 500, 25, 20, 1,00, 1, 3 ) == -1 ) {
    printf( "Unable to build a Dual Tone Cadence" );
    printf( " Template\n");
}
/*
* Continue Processing
* .
* .
*/
/*
* Close the opened Voice Channel Device
*/
if ( vpm_close( vpmdev ) != 0 ) {
    perror( "close" );
}
/* Terminate the Program */
exit( 0 );
}

```

■ Error



Global Tone Detection :

”Global Tone Detection”(Voice Features Guide for Windows 2000)

Building Tones :

- vpm_bldst()
- vpm_blddt()
- vpm_bldstcad()

Enabling Tone Detection

- vpm_addtone()
- vpm_distone()
- vpm_enbtone()

vpm_bldst() defines a simple single frequency tone

Name : int vpm_bldst(tid,freq,fqdev,,mode)

Inputs :

unsigned int tid : tone ID to assign

unsigned int freq : frequency 1 in Hz

unsigned int fqdev : frequency 1 Deviation in Hz

unsigned int mode : leading or trailing edge

Returns :

0 :

-1 :

Includes: srllib.h, smartsdk.h

Category: Global ToneDetection



vpm_bldstcad() Simple Single Frequency tone .
vpm_addtone() 가 vpm_addtone() vpm_bldst()
tone tone()
vpm_bldst() tone . Tone
vpm_addtone() 가
tone detection vpm_addtone .
Parameter .

tid : tone tone ID 100 – 150 가 .
freq : tone frequency
fqdev : tone frequency deviation
mode : tone detection leading edge trailing edge
define 가 .
- TN_LEADING
- TN_TRAILING



1. vpm_initcallp() 3 tone 가 ,
8 가 가 .


```

vpm_blddt()      vpm_addtone()      . tone
vpm_addtone()   가      create      , vpm_blddt()
               가      ,      tone      tone
               .      , vpm_blddt()      , vpm_addtone()
error 가      .

```

■ Example

```

#include <stdio.h>
#include <errno.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
#define TID_3 103
main()
{
int vpmdev;
/*
* Open the Voice Channel Device and Enable a Handler
*/
if ( ( vpmdev = vpm_open( "vpmB1C1", NULL) ) == -1 ) {
perror( "vpmB1C1" );
exit( 1 );
}
/*
* Describe a Simple Single Tone Frequency Tone of
* 950-1050 Hz using trailing edge detection.
*/
if ( vpm_bldst( TID_3, 1000, 50, TN_TRAILING ) == -1 ) {
printf( "Unable to build a Single Tone Template\n" );
}
/*
* Continue Processing
* .
* .
* .

```



```

*/
/*
* Close the opened Voice Channel Device
*/
if ( vpm_close( vpmdev ) != 0 ) {
perror( "close" );
}
/* Terminate the Program */
exit( 0 );
}

```

■ Error



Global Tone Detection :

"Global Tone Detection"(Voice Features Guide for Windows 2000)

Building Tones :

- vpm_ vpm_blddtcad()
- vpm_blddt()
- vpm_bldstcad()

Enabling Tone Detection

- vpm_addtone()
- vpm_distone()
- vpm_enbtone()

vpm_bldstcad() defines a simple dual frequency cadence tone

Name : int vpm_blddtcad(tid,freq1,fq1dev,freq2,fq2dev,
ontime,ontdev,offtime,offdev,mode)

Inputs :

unsigned int tid : tone ID to assign
 unsigned int freq : frequency in Hz
 unsigned int fqdev : frequency Deviation in Hz
 unsigned int ontime : tone-on time(10ms unit)
 unsigned int ondev : tone-on time deviation(10ms unit)
 unsigned int offtime : tone-off time(10ms unit)
 unsigned int offdev : tone-off time deviation(10ms unit)
 unsigned int repcnt : cadence갓 ,

Returns :

0 :
 -1 :

Includes: srllib.h, smartsdk.h

Category: Global Tone Detection



vpm_blddtcad() Simple Single Frequency Cadence tone .
 vpm_addtone() 갓 vpm_addtone()
 vpm_bldstcad() tone tone()
 .
 vpm_bldstcad() tone .
 Tone vpm_addtone() 갓
 tone detection vpm_addtone .
 Parameter .

tid : tone tone ID 100 – 150 갓 .
Freq : tone frequency
fqdev : tone frequency deviation
ontime : Cadence갓 ON Time Length(10ms unit)
ontdev : Ontime deviation

offtime : Cadence가 OFF Time Length(10ms unit)

offtdev : Offtime deviation

repcnt : cadence (, signal on off)



```
1. vpm_initcallp()          3          tone          가          ,
    8          가          가          .
    vpm_blddt()          vpm_addtone()          . tone
    vpm_addtone()          가          create          , vpm_blddt()
    가          ,          tone          tone
    .          , vpm_blddt()          , vpm_addtone()
    error 가          .
```

■ Example

```
#include <studio' s>
#include <errno.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
#define TID_4 104
main()
{
int vpmdev;
/*
* Open the Voice Channel Device and Enable a Handler
*/
if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
    perror( "vpmB1C1" );
    exit( 1 );
}
/*
* Describe a Single Tone Frequency Tone of 950-1050 Hz.
* On between 190-210 msec and off 990-1010 msec and
* a cadence of 3.
*/
```



```

if ( vpm_bldstcad( TID_4, 1000, 50, 20, 1, 100, 1, 3 ) == -1 ) {
    printf( "Unable to build a Single Tone Cadence" );
    printf( " Template\n");
}
/*
* Continue Processing
* .
* .
* .
*/
/*
* Close the opened Voice Channel Device
*/
if ( vpm_close( vpmdev ) != 0 ) {
    perror( "close" );
}
/* Terminate the Program */
exit( 0 );
}

```

■ Error



Global Tone Detection :

”Global Tone Detection”(Voice Features Guide for Windows 2000)

Building Tones :

- vpm_ vpm_blddtcad()
- vpm_blddt()
- vpm_bldst()

Enabling Tone Detection

- vpm_addtone()
- vpm_distone()
- vpm_enbtone()

vpm_bldtngen()

sets up tone generation template

Name : int vpm_bldtngen(tngenp,freq1,freq2,ampl1,ampl2,duraion)

Inputs :

TN_GEN *tngenp : pointer to tone generation structure

unsigned short freq1 : tone frequency

unsigned short freq2 : tone frequency

short ampl1 : tone amplitude

short ampl2 : tone amplitude

short duration : 10ms unit tone duration

Returns :

0 :

-1 : tone Candece 가 .

-2 : tone type

Includes: srllib.h, smartsdk.h

Category: PerfectCall Call Analysis



vpm_bldtngen() TN_GEN structure field

tone generation template Convenience function .

Tone generation template tngenp structure pointer ,

vpm_playtone() tone generation .

Parameter .

tngenp : tone generation template TN_GEN structure pointer

Freq1 : tone frequency(200-3000Hz)

Freq2 : tone frequency(200-3000Hz), single tone

freq2가 0

ampl1 : tone amplitude(-40 – 20db)

ampl2 : tone amplitude(-40 – 20db)

duration : tone duration 10ms unit , -1

duration 가 .



■ Example

```
#include <stdio.h>
#include <errno.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    TN_GEN tngen;
    int vpmdev;
    /*
    * Open the Voice Channel Device and Enable a Handler
    */
    if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
        perror( "vpmB1C1" );
        exit( 1 );
    }
    /*
    * Build a Tone Generation Template.
    * This template has Frequency1 = 1140,
    * Frequency2 = 1020, amplitude at -10dB for
    * both frequencies and duration of 100 * 10 msecs.
    */
    vpm_bldtnngen( &tngen, 1140, 1020, -10, -10, 100 );
    /*
    * Continue Processing
    * .
    * .
    * .
    */
    /*
    * Close the opened Voice Channel Device
    */
```



```

if ( vpm_close( vpmdev ) != 0 ) {
    perror( "close" );
}
/* Terminate the Program */
exit( 0 );
}

```

■ Error



Generation Tones :

- **TN_GEN**(chapter 4. Voice Data Structures and Device Parameters)
- **vpm_playtone()**
- **"Global Tone Generation"**(Voice Feature Guide for Windows 2000)

alters standard definition of duration component

Inputs :

```
int ontime : on duration
```

```
int offtime : off duration
```

int offdev : offtime deviation

Returns :

0:

-1 : tone does not cadence values

-2 : unknown tone type

Includes: srl.lib.h, smartsdk.h

Category: Global Tone Generation

VPM Library	Perfectcall Analysis tone	가	.
	tone	tonetype	.
vpm_chgdur()	duration component		definition

```

tone                                application
vpm_initcallp()   가   tone definition   가
가   .   , tone definition
.   ,vpm_deltone()
가   ,   vpm_initcallp()   가

```

Parameter

tonetype : tone type 가 .

- TID_BUSY1 : Busy Signal
- TID_BUSY2 : Alternate busy signal

: 480 + 620 Hz

- TID_DIAL_INTL : International dial tone
: 350 + 440 Hz
- TID_DIAL_LCL : Local Dial tone
- TID_DIAL_XTRA : Special(“ extra”) dial tone
- TID_FAX1 : Fax or Modem tone
- TID_FAX2 : Alternate fax or modem tone
: 1100 Hz(CNG tone)
- TID_RINGBK1: Ringback

ontime : tone on time 10ms unit

ondev : tone on time Maximum deviation

offtime : tone off time 10ms unit

offdev : tone off time Maximum deviation

■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    DX_CAP cap_s;
    int ddd, car;
    char *chnam, *dialstrg;
    chnam = "vpmB1C1";
    dialstrg = "L1234";
    /*
    * Open channel
    */
    if ((ddd = vpm_open( chnam, NULL )) == -1 ) {
        /* handle error */
    }
    /*
    * Delete any previous tones
    */
    if ( vpm_deltone(ddd) < 0 ) {
        /* handle error */
    }
}
```



```

}
/*
* Change Enhanced call progress default local dial tone
*/
if (vpm_chgfreq( TID_DIAL_LCL, 425, 150, 0, 0 ) < 0) {
/* handle error */
}
/*
* Change Enhanced call progress default busy cadence
*/
if (vpm_chgdur( TID_BUSY1, 550, 400, 550, 400 ) < 0) {
/* handle error */
}
if (vpm_chgrepcnt( TID_BUSY1, 4 ) < 0) {
/* handle error */
}
/*
* Now enable Enhanced call progress with above changed settings.
*/
if (vpm_initcallp( ddd )) {
/* handle error */
}
/*
* Set off Hook
*/
if ((vpm_sethook( ddd, DX_OFFHOOK, EV_SYNC )) == -1) {
/* handle error */
}
/*
* Dial
*/
if ((car = vpm_dial( ddd, dialstrg,(DX_CAP *)&cap_s, DX_CALLP|EV_SYNC))== -1) {
/* handle error */
}
switch( car ) {
case CR_NODIALTONE:

```



```

        printf(" Unable to get dial tone\n");
        break;
case CR_BUSY:
        printf(" %s engaged\n", dialstrg );
        break;
case CR_CNCT:
        printf(" Successful connection to %s\n", dialstrg );
        break;
default:
        break;
}
/*
 * Set on Hook
 */
if ((vpm_sethook( ddd, DX_ONHOOK, EV_SYNC )) == -1) {
/* handle error */
}
vpm_close( ddd );
}

```



```

                                signal . signal
vpm_deltones( , vpm_initcallp() .

```



- **vpm_chgfreq()**
- **vpm_chgrepcnt()**
- **vpm_deltones()**
- **vpm_initcallp()**

Parameter

tonetype : tone type 가 .

- TID_BUSY1 : Busy Signal
- TID_BUSY2 : Alternate busy signal
: 480 + 620 Hz
- TID_DIAL_INTL : International dial tone
: 350 + 440 Hz
- TID_DIAL_LCL : Local Dial tone
- TID_DIAL_XTRA : Special(" extra") dial tone
- TID_FAX1 : Fax or Modem tone
- TID_FAX2 : Alternate fax or modem tone
: 1100 Hz(CNG tone)
- TID_RINGBK1: Ringback

freq1 : tone frequency

freq1dev : frequency 1 maximum deviation

freq2 : tone frequency

freq2dev : frequency 2 maximum deviation

■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    DX_CAP cap_s;
    int ddd, car;
    char *chnam, *dialstrg;
    chnam = "vpmB1C1";
    dialstrg = "L1234";
    /*
    * Open channel
    */
    if ((ddd = vpm_open( chnam, NULL )) == -1 ) {
```



```

        /* handle error */
    }
    /*
    * Delete any previous tones
    */
    if ( vpm_deltone(ddd) < 0 ) {
        /* handle error */
    }
    /*
    * Change Enhanced call progress default local dial tone
    */
    if (vpm_chgfreq( TID_DIAL_LCL, 425, 150, 0, 0 ) < 0) {
        /* handle error */
    }
    /*
    * Change Enhanced call progress default busy cadence
    */
    if (vpm_chgdur( TID_BUSY1, 550, 400, 550, 400 ) < 0) {
        /* handle error */
    }
    if (vpm_chgrepcnt( TID_BUSY1, 4 ) < 0) {
        /* handle error */
    }
    /*
    * Now enable Enhanced call progress with above changed settings.
    */
    if (vpm_initcallp( ddd )) {
        /* handle error */
    }
    /*
    * Set off Hook
    */
    if ((vpm_sethook( ddd, DX_OFFHOOK, EV_SYNC )) == -1) {
        /* handle error */
    }
    /*

```



```

* Dial
*/
if ((car = vpm_dial( ddd, dialstrg,(DX_CAP *)&cap_s, DX_CALLP|EV_SYNC))== -1) {
    /* handle error */
}
switch( car ) {
case CR_NODIALTONE:
    printf(" Unable to get dial tone\n");
    break;
case CR_BUSY:
    printf(" %s engaged\n", dialstrg );
    break;
case CR_CNCT:
    printf(" Successful connection to %s\n", dialstrg );
    break;
default:
    break;
}
/*
* Set on Hook
*/
if ((vpm_sethook( ddd, DX_ONHOOK, EV_SYNC )) == -1) {
    /* handle error */
}
vpm_close( ddd );
}
■

■
· vpm_chgdur()
· vpm_chgrepcnt()
· vpm_deltone()
· vpm_initcallp()

```


vpm_chgrepcnt()

change the standard definition

Name : int vpm_chgrepcnt(tonetype,repcnt)

Inputs :

int tonetype : tone to modify

int repcnt : repetition count

Returns :

0 :

-1 : tone does not cadence values

-2 : unknown tone type

Includes: srllib.h, smartsdk.h

Category: Global Tone Generation



vpm_chgrepcnt() tone type PerfectCall Analysis tone
repetition count .

VPM Library Perfectcall Analysis tone 가 .
vpm_chgrepcnt() repetition component definition
.

tone application .
vpm_initcallp() 가 tone definition 가
가 . , tone definition
 . ,vpm_deltone()

가 , vpm_initcallp() 가

.

Parameter

tonetype : tone type 가 .
- TID_BUSY1 : Busy Signal
- TID_BUSY2 : Alternate busy signal
 : 480 + 620 Hz
- TID_DIAL_INTL : International dial tone

: 350 + 440 Hz

- TID_DIAL_LCL :Local Dial tone
- TID_DIAL_XTRA : Special(" extra") dial tone
- TID_FAX1 : Fax or Modem tone
- TID_FAX2 : Alternate fax or modem tone
- : 1100 Hz(CNG tone)
- TID_RINGBK1:Ringback

repcnt : signal

■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    DX_CAP cap_s;
    int ddd, car;
    char *chnam, *dialstrg;
    chnam = "vpmB1C1";
    dialstrg = "L1234";
    /*
    * Open channel
    */
    if ((ddd = vpm_open( chnam, NULL )) == -1 ) {
        /* handle error */
    }
    /*
    * Delete any previous tones
    */
    if ( vpm_deltone(ddd) < 0 ) {
        /* handle error */
    }
    /*
    * Change Enhanced call progress default local dial tone
    */
```



```

if (vpm_chgfreq( TID_DIAL_LCL, 425, 150, 0, 0 ) < 0) {
    /* handle error */
}
/*
* Change Enhanced call progress default busy cadence
*/
if (vpm_chgdur( TID_BUSY1, 550, 400, 550, 400 ) < 0) {
    /* handle error */
}
if (vpm_chgrepcnt( TID_BUSY1, 4 ) < 0) {
    /* handle error */
}
/*
* Now enable Enhanced call progress with above changed settings.
*/
if (vpm_initcallp( ddd )) {
    /* handle error */
}
/*
* Set off Hook
*/
if ((vpm_sethook( ddd, DX_OFFHOOK, EV_SYNC )) == -1) {
    /* handle error */
}
/*
* Dial
*/
if ((car = vpm_dial( ddd, dialstrg, (DX_CAP *)&cap_s, DX_CALLP|EV_SYNC)) == -1) {
    /* handle error */
}
switch( car ) {
case CR_NODIALTONE:
    printf(" Unable to get dial tone\n");
    break;
case CR_BUSY:
    printf(" %s engaged\n", dialstrg );

```



```

        break;
case CR_CNCT:
    printf(" Successful connection to %s\n", dialstrg );
    break;
default:
    break;
}
/*
* Set on Hook
*/
if ((vpm_sethook( ddd, DX_ONHOOK, EV_SYNC )) == -1) {
    /* handle error */
}
vpm_close( ddd );
}

```



```

                                tone . tone
vpm_deltone( ) , vpm_initcallp( ) .

```



- **vpm_chgdur()**
- **vpm_chgfreq()**
- **vpm_deltone()**
- **vpm_initcallp()**

vpm_close()

closes SCT devices

Name : int vpm_close(dev)

Inputs :

int dev : vpm channel

Returns :

0 :

-1 :

Includes: srllib.h, smartsdk.h

Category: Device Management



vpm_close() vpm_open() open VPM device close .
device handle , device 가 busy idle
handle .

NOTE : vpm_close() event .
device hook state parameter .

Parameter .

dev : vpm_open VPM channel board handle



device close , process device handle 가 action
device handle , vpm_close()

call process .

NOTE : 1.vpm_close() device .
calling process device link . device
handle 가 device link .

2. NT system close .

3. vpm_close handle event .

■ Example

```
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    DX_CAP cap_s;
    int chdev;
    if(vpm_close(chdev) == -1 )
}
```

■ Errors

가 -1 , ATDV_LASTERR() ATDV_ERRMSGP()
Error .

EINVAL : Invalid Argument

EBADF : Invalid descriptor

EINTR : A signal was caught



· vpm_open()


```

·
/* set call analysis parameters before doing call analysis */
    vpm_clrcap(&cap);
    cap.ca_nbrdna = 5; /* 5 rings before no answer */
·
·
/* continue with call analysis */
·
·
}

```

■ Errors



- **vpm_dial()**
- **DX_CAP**(chapter 4. Voice Data Structures and Device Parameters)
- **"Call Analysis"**(Voice Feature Guide for Windows 2000)

vpm_clrdigbuf() causes the digits present in the digit buffer

Name : int vpm_clrdigbuf(chdev)

Inputs :

int chdev : VPM channel handle

Returns :

0 :

-1 :

Includes: srl.lib.h, smartsdk.h

Category: Structure Clearance



vpm_clrdigbuf()	chdev	channel	digit buffer
-----------------	-------	---------	--------------

.

Parameter

chdev : vpm_open VPM channel handle



■ Example

```
#include <srl.lib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
int chdev; /* channel descriptor */
.
.
.
/* Open Channel */
if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
/* process error */
```



```

}
/* Clear digit buffer */
if (vpm_clrdigbuf(chdev) == -1) {
    /* process error*/
}
.
.
}

```

vpm_clrditbuf()

vpm_getdig(),vpm_play(),vpm_rec()

■ Errors

EDX_BADPARM : Invalid Parameter

EDX_SYSTEM : Windows 2000 System error – check errno



- **vpm_getdig()**
- **vpm_play()**
- **vpm_rec()**

vpm_clrsvcond() clear all speed or volume adjustment conditions

Name : int vpm_ clrsvcond (chdev)

Inputs : int chdev : VPM channel handle

Returns : 0 : , -1 :

Includes: srllib.h, smartsdk.h

Category: Structure Clearance



vpm_ clrsvcond ()

Parameter

chdev : vpm_open VPM channel handle



■ Example

```
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
int chdev; /* channel descriptor */

/* Open Channel */
if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
    /* process error */
}
```



```

/*
* Clear all Speed and Volume Conditions
*/
if ( vpm_clrsvcond(chdev) == -1 ) {
    printf( "Unable to Clear the Speed/Volume" );
    printf( " Conditions\n" );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR(chdev), ATDV_ERRMSGP(chdev) );
    vpm_close(chdev);
    exit( 1 );
}

/*
* Continue Processing
*/
/*
* Close the opened Voice Channel Device
*/
if ( vpm_close(chdev) !=0 ){
    perror( "close" );
}
/* Terminate the Program */
exit( 0 );
}

```

■ **Errors**

1	ATDV_LASTERR()	ATDV_ERRMSGP()
	:	
EDX_BADPARAM	:	Invalid Parameter
EDX_BADPROD	:	Function not supported on this board
EDX_SYSTEM	:	Windows 2000 system error - check errno



vpm_setsvcond()

vpm_addspddig()

vpm_addvoldig()

Speed and Volume Modification Tables (*Voice Features Guide*)

DX_SVCB structure

vpm_clrtpt()

clears all DV_TPT fields

Name : int vpm_clrtpt(tp, size)

Inputs : DV_TPT *tp : pointer of termination parameter structure
int size : number of entries to clear

Returns :

0 :

-1 :

Includes: srllib.h, smartsdk.h

Category: Structure Clearance



vpm_clrtpt() size parameter DV_TPT structure field tp_type
tp_nextp field , .
vpm_clrtpt() termination ,
DV_TPT structure field .
NOTE : DV_TPT Voice Device 가 device ,
Srllib.h . DV_TPT voice value Appendix A
 , DV_TPT structure “Standard Runtime Library
Programmer’s Guide” .
vpm_clrtpt() , DV_TPT tp_type
 .

IO_CONT : if the next DV_TPT is continuous

IO_LINK : if the next DV_TPT is linked

IO_EOT : for the last DV_TPT

tp_type IO_LINK , tp_nextp 가 DV_TPT structure
 . vpm_clrtpt() tp_type information , tp_type
IO_LINK , tp_nextp , DV_TPT structure
 . tp_type tp_nextp field , vpm_clrtpt()
 , link DV_TPT structure clear .

Parameter

tptp : DV_TPT structure pointer DV_TPT structure information, Appendix A
size : DV_TPT structure size가 0 , 0 .

■
 vpm_clrtp() DV_TPT structure access tp_type
 tp_nextp(tp_type IO_LINK) DV_TPT
 structure tp_type IO_EOT . DV_TPT structure
 , vpm_clrtp() 가 .

■ Example

```
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    DV_TPT tpt1[2];
    DV_TPT tpt2[2];
    /* Set up the links in the DV_TPTs */
    tpt1[0].tp_type = IO_CONT;
    tpt1[1].tp_type = IO_LINK;
    tpt1[1].tp_nextp = &tpt2[0];
    tpt2[0].tp_type = IO_CONT;
    tpt2[1].tp_type = IO_EOT;
    /* set up the other DV_TPT fields as required for termination */
    .
    .
    /* play a voice file, get digits, etc. */
    .
    .
    /* clear out the DV_TPT structures if required */
    vpm_clrtp(&tpt1[0],4);
    /* now set up the DV_TPT structures for the next play */
}
```



```

        .
        .
    }

```

■ Errors

size parameter DV_TPT structure , DV_TPT
 structure tp_type field IO_EOT가 , -1

.



- DV_TPT(Chapter 4. Voice Data Structures and Device Parameters)

vpm_deltone()

removes all user-defined tones

Name : int vpm_deltone(chdev)

Inputs : int chdev : VPM channel device handle

Returns :

0 :

-1 :

Includes: srllib.h, smartsdk.h

Category: General tone detection



vpm_deltone()

vpm_addtone()

user_defined tone

.

NOTE :

vpm_blddt(),vpm_bltst(),vpm_bldstcad(),vpm_blddtcad()

use defined tone

.

Parameter

.

chdev : vpm_open()

VPM channel device handle



■ Example

```
#include <stdio.h>
```

```
#include <errno.h>
```

```
#include <srllib.h>
```

```
#include <smartsdk.h>
```

```
#include <windows.h>
```

```
main()
```

```
{
```

```
int vpmdev;
```

```
/*
```

```
* Open the Voice Channel Device and Enable a Handler
```

```
*/
```



```

if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
    perror( "vpmB1C1" );
    exit( 1 );
}
/*
* Delete all Tone Templates
*/
if ( vpm_deltones( vpmdev ) == -1 ) {
    printf( "Unable to Delete all the Tone Templates\n" );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSGP( vpmdev ) );
    vpm_close( vpmdev );
    exit( 1 );
}
/*
* Continue Processing
* .
* .
* .
*/
/*
* Close the opened Voice Channel Device
*/
if ( vpm_close( vpmdev ) != 0 ) {
    perror( "close" );
}
/* Terminate the Program */
    exit( 0 );
}

```

Errors

가 -1 , ATDV_LASTERR() ATDV_ERRMSGP()
 Error .

EDX_BADPARAM : Invalid Parameter

EDX_SYSTEM : Windows 2000 System error – check errno



Adding and Enabling User-defined Tones:

- `vpm_addtone()`
- `vpm_enbtone()`

Building Tones :

- `vpm_blddt()`
- `vpm_bldst()`
- `vpm_blddtcad()`

vpm_dial()

dials and ASCIIZ string

Name : int vpm_dial(chdev,dialstrp,capp,mode)

Inputs : int chdev : VPM channel device handle

char *dialstrp : ASCIIZ dial string pointer

DX_CAP *capp : Call Analysis Parameter Structure pointer

unsigned short mode : Async,Sync mode Call Analysis
Flag

Returns :

0 : Asynchronous

>= 0 : Synchronous Call Analysis

-1 :

Includes: srl.lib.h, smartsdk.h

Category: I/O

Mode: synchronous/asynchronous



vpm_dial() ASCIIZ string 가 , open idle channel
, dial . , call Call Analysis 가

dial Call Analysis channel VPMX_STATE()

, return .

CS_DIAL : dial state(Call Analysis)

CS_CALL : Call Analysis state

NOTE : vpm_dial() HOOK . Call Analysis 가

vpm_dial() I/O ,

vpm_stopch() .

Parameter .

chdev : vpm_open() VPM channel device handle

dialstrp : ASCII dial string pointer, dialstrp ASCII character

null terminated string 가 . dialing

table .

Table 3. Valid Characters for Each Dialing Mode

Pulse Digit	Description	DTMF Digit	Description
“0” – “9”		“0” – “9”	
“*”		“*”	
“#”		“#”	
“a”		“a”	
“b”		“b”	
“c”		“c”	
“d”		“d”	
“ ”	Pause	“ ”	Pause
“&”	Flash	“&”	flash
“L”		“L”	
“I”		“I”	
“X”		“X”	
change Dial mode to :		change Dial mode to :	
“P”	Pulse mode	“P”	Pulse mode
“T”	DTMF mode	“T”	DTMF mode

dialing mode dial string “T” (DTMF Tone), “P” (Pulse dialing)
, DTMF dialing

capp : DX_CAP Call Analysis Parameter Structure pointer.(Structure 4
VPM Data Structure Parameter)
default Call Analysis Parameter capp NULL
mode DX_CALLP .

mode : ASCIIZ string 가 dial , Call Analysis 가
, 가

. mode BIT Mask .
DX_CALLP : Enable Call Analysis
EV_ASYNC : vpm_dial()
EV_SYNC : vpm_dial()
vpm_dial() Call Analysis EV_ASYNC,EV_SYNC

NOTE : vpm_dial() 가 channel ON HOOK Call
dial ,. Call Analysis .

■ Asynchronous Operation

mode field bitwise OR operation EV_ASYNC .

0 event .

, dial , call Call Analysis 가

TDX_DIAL : dial (Call Analysis)

TDX_CALLP : dial (Call Analysis)

vpm_dial 가 TDX_DIAL event ,

VPMX_TERMMSK() , TDX_CALLP event ,

VPMX_CPTERM() . Call Analysis Call Analysis

description .

■ Synchronous Operation

0 .

dialing , Call Analysis 가 Call

Process Result , 0 .

■ Call Analysis

Call Analysis call mode filed ,

dial call . vpm_dial() default Call Analysis

Parameter DX_CAP structure .

Call Analysis VPMX_CPTERM() .

vpm_dial() 가 , 가

Call Analysis Features Guide .

가 Call Analysis termination .

CR_BUSY : line was busy

CR_CNCT : Call Connected

CR_ERROR : Call Analysis error

CR_FAXTONE : fax machine or modem

CR_NOANS : No Answer

CR_NODIALTONE : no dial tone

CR_NORB : no ring back

CR_STOPD : Call Analysis Stopped due to vpm_stopch()

Call Analysis 가 , Call 가
extended Attribute function .

VPMX_ANSRSIZ() : Returns duration of answer

VPMX_CPTERM() : Returns last Call Analysis termination

VPMX_CRTNID() : Returns tone identifier

VPMX_DTNFAIL() : Returns dial tone fail character

VPMX_LONGLOW() : Returns duration of longer silence

VPMX_SHORTLOW() : Returns duration of shorter silence

VPMX_SIZEHI() : Returns duration of non silence



1. Call Analysis channel dial , vpm_stopch() dial
0 . dialstrp dial .
2. Call Analysis dial vpm_stopch() dial
Call Analysis .
dialstrp digit dial , stop information
Extended Attribute .



Example 1 : Call Analysis with user-specified parameters()

```
/* Call Analysis with user-specified parameters and synchronous mode. */  
#include <stdio.h>  
#include <srllib.h>  
#include <smartsdk.h>  
#include <windows.h>  
main()  
{  
    int cares, chdev;  
    DX_CAP capp;  
    .  
    .  
    /* open the channel using vpm_open( ). Obtain channel device descriptor in  
    * chdev  
    */
```



```

if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
    /* process error */
}
/* take the phone off-hook */
if ((vpm_sethook(chdev,DX_OFFHOOK,EV_SYNC)) == -1) {
    /* process error */
}
/* Clear DX_CAP structure */
vpm_clrcap(&capp);
/* Set the DX_CAP structure as needed for call analysis.
 * Allow 3 rings before no answer.
 */
capp.ca_nbrdna = 3;
/* Perform the outbound dial with call analysis enabled. */
if ((cares = vpm_dial(chdev,"5551212",&capp,DX_CALLP|EV_SYNC)) == -1) {
    /* perform error routine */
}
switch (cares) {
case CR_CNCT: /* Call Connected, get some additional info */
    printf("\nDuration of short low - %ld ms",
           VPMX_SHORTLOW(chdev)*10);
    printf("\nDuration of long low - %ld ms",
           VPMX_LONGLOW(chdev)*10);
    printf("\nDuration of answer - %ld ms",
           VPMX_ANSRSIZ(chdev)*10);
    break;
case CR_BUSY:
    .
    .
}
/* carry out the next state */
.
.
}

```

Example 2 : Call Analysis with default parameters()


```

/* Call Analysis with default parameters and synchronous mode. */
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
int cares, chdev;
DX_CAP capp;
/* open the channel using vpm_open( ). Obtain channel device
descriptor* in chdev*/
if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
/* process error */
}
/* take the phone off-hook */
if ((vpm_sethook(chdev,DX_OFFHOOK,EV_SYNC)) == -1) {
/* process error */
}
/* Perform the outbound dial with call analysis enabled and capp set to
* NULL
*/
if((cares=vpm_dial(chdev,"5551212",(DX_CAP *)NULL,DX_CALLP|EV_SYNC))
== -1) {
/* perform error routine */
}
/* Analyze the call analysis results as in Example 1 */
.
.
}

```

Example 3 : Call Analysis with default parameters(,Callback)

```

/* Call Analysis with user-specified parameters and asynchronous, callback mode. */
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
#define MAXCHAN 24

```



```

int dial_handler();
DX_CAP capp;
main()
{
    int i, chdev[MAXCHAN];
    char *chnamep;

    for (i=0; i<MAXCHAN; i++) {
        /* Set chnamep to the channel name, e.g., vpmB1C1, vpmB1C2 */
        /* Open the device using vpm_open( ). chdev[i] has channel device
        * descriptor. */
        if ((chdev[i] = vpm_open(chnamep, NULL)) == -1) {
            /* process error */
        }
        /* Using sr_enbhdr(), set up handler function to handle call analysis *
        completion events on this channel. */
        if (sr_enbhdr(chdev[i], TDX_CALLP, dial_handler) == -1) {
            /* process error */
        }
        /* Before issuing vpm_dial(), place the phone off-hook. */
        /* Clear DX_CAP structure */
        vpm_clrkap(&capp);
        /* Set the DX_CAP structure as needed for call analysis.
        * Allow 3 rings before no answer.
        */
        capp.ca_nbrdna = 3;
        /* Perform the outbound dial with call analysis enabled. */
        if (vpm_dial(chdev[i], "5551212", &capp, TDX_CALLP|EV_ASYNC) == -
1) {
            /* perform error routine */
        }
        /* Use sr_waitevt() to wait for the completion of call analysis.
        On receiving the completion event, TDX_CALLP, control I
        s transferred
        * to the handler function previously established using sr_enbhdr().
        */
    }
}

```



```

.
.
    }
}

int dial_handler()
{
    int chdev;
    chdev = sr_getevtdev();
    switch (VPMX_CPTERM(chdev)) {
    case CR_CNCT: /* Call Connected, get some additional info */
        printf("\nDuration of short low - %ld ms",
                VPMX_SHORTLOW(chdev)*10);
        printf("\nDuration of long low - %ld ms",
                VPMX_LONGLOW(chdev)*10);
        printf("\nDuration of answer - %ld ms",
                VPMX_ANSRSIZ(chdev)*10);

        break;
    case CR_BUSY:
        .
        .
    }
    /* Kick off next function in the state machine model. */
    .
    .
    return 0;
}

```

Example 4 : Perfect Call Analysis ()

```

#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    DX_CAP cap_s;

```



```

int ddd, car;
char *chnam, *dialstrg;
chnam = "vpmB1C1";
dialstrg = "L1234";
/*
* Open channel
*/
if ((ddd = vpm_open( chnam, NULL )) == -1 ) {
/* handle error */
}
/*
* Delete any previous tones
*/
if ( vpm_deltone(ddd) < 0 ) {
/* handle error */
}
/*
* Change Enhanced call progress default local dial tone
*/
if (vpm_chgfreq( TID_DIAL_LCL, 425, 150, 0, 0 ) < 0) {
/* handle error */
}
/*
* Change Enhanced call progress default busy cadence
*/
if (vpm_chgdur( TID_BUSY1, 550, 400, 550, 400 ) < 0) {
/* handle error */
}
if (vpm_chgrepcnt( TID_BUSY1, 4 ) < 0) {
/* handle error */
}
/*
* Now enable Enhanced call progress with above changed settings.
*/
if (vpm_initcallp( ddd )) {
/* handle error */
}

```



```

}
/*
* Set off Hook
*/
if ((vpm_sethook( ddd, DX_OFFHOOK, EV_SYNC )) == -1) {
    /* handle error */
}
/*
* Dial
*/
if ((car = vpm_dial( ddd, dialstrg,(DX_CAP *)&cap_s, DX_CALLP|EV_SYNC))
                                     ==-1) {
    /* handle error */
}
switch( car ) {
case CR_NODIALTONE:
    printf(" Unable to get dial tone\n");
    break;
case CR_BUSY:
    printf(" %s engaged\n", dialstrg );
    break;
case CR_CNCT:
    printf(" Successful connection to %s\n", dialstrg );
    break;
default:
    break;
}
/*
* Set on Hook
*/
if ((vpm_sethook( ddd, DX_ONHOOK, EV_SYNC )) == -1) {
    /* handle error */
}
vpm_close( ddd );
}

```


■ Errors

가 -1 , ATDV_LASTERR() ATDV_ERRMSGP()
Error .

EDX_BADPARM : Invalid Parameter

EDX_BUSY : Channled is busy

EDX_SYSTEM : Windows 2000 System error – check errno



- **vpm_stopch()**

Retrieving termination reasons and events for vpm_dil() with Call Analysis:

**Event Management Functions(Standard runtime Library Progrmmers Guide for
Windows 2000)**

- **VPMX_CPTERM()**

Retrieving termination reasons for vpm_dial() without Call Analysis :

- **VPMX_TERMMSK()**

Call Analysis

- **DX_CAP(chapter 4. Voice Data Structures and Device Parameters)**
- **"Call Analysis"(Voice Features Guide for Windows 2000)**
- **VPMX_ANSRSIZE()**
- **VPMX_LONGLOW()**
- **VPMX_SHORTLOW()**
- **VPMX_SIZEHI()**

vpm_distone()

disables detection of TONE ON

Name : int vpm_distone(chdev,toneid,evt_mask)

Inputs : int chdev : VPM channel device handle
int toneid : tone template
int evt_mask : event mask

Returns :

0 :

-1 :

Includes: srllib.h, smartsdk.h

Category: Global Tone Detection



vpm_distone() channel user defined tone tone-on,tone-off
event detection . vpm_addtone() 가
user-defined tone detection 가 .

Parameter .

chdev : vpm_open() VPM channel device handle

toneid : user defined tone ID , channel
user-defined tone , toneid TONEALL

evtmask : going on event going off event

parameter BIT masking .

DM_TONEON : TONE ON detection

DM_TONEOFF : TONE OFF detection

evt_mask tone template enable/disable ,

vpm_distone(),vpm_enbtone .

■ Example

```
#include <stdio.h>
```

```
#include <errno.h>
```

```
#include <srllib.h>
```

```
#include <smartsdk.h>
```

```
#include <windows.h>
```



```

#define TID_1 101
main()
{
int vpmdev;
/*
* Open the Voice Channel Device and Enable a Handler
*/
if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
    perror( "vpmB1C1" );
    exit( 1 );
}
/*
* Describe a Simple Dual Tone Frequency Tone of 950-
* 1050 Hz and 475-525 Hz using leading edge detection.
*/
if ( vpm_blddt( TID_1, 1000, 50, 500, 25, TN_LEADING ) == -1 ) {
    printf( "Unable to build a Dual Tone Template\n" );
}
/*
* Bind the Tone to the Channel
*/
if ( vpm_addtone( vpmdev, NULL, 0 ) == -1 ) {
    printf( "Unable to Bind the Tone %d\n", TID_1 );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSG( vpmdev ) );
    vpm_close( vpmdev );
    exit( 1 );
}
/*
* Disable Detection of ToneId TID_1
*/
if ( vpm_distone( vpmdev, TID_1, DM_TONEON | DM_TONEOFF ) == -1 ) {
    printf( "Unable to Disable Detection of Tone %d\n", TID_1 );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSG( vpmdev ) );
    vpm_close( vpmdev );
}

```



```

        exit( 1 );
    }
    /*
    * Continue Processing
    * .
    * .
    * .
    */
    /*
    * Close the opened Voice Channel Device
    */
    if ( vpm_close( vpmdev ) != 0 ) {
        perror( "close" );
    }
    /* Terminate the Program */
    exit( 0 );
}

```

■ Errors

가 -1, ATDV_LASTERR() ATDV_ERRMSGP()
Error .

EDX_BADPARAM : Invalid Parameter

EDX_SYSTEM : Windows 2000 System error – check errno

EDX_TNMSGSTATUS : Invalid Message status setting

EDX_TONEID : Bad tone ID



General one Detection functions :

- vpm_addtone()
- vpm_blddt(),vpm_bldst(),vpm_blddtcad(),vpm_bldstcad()
- vpm_enbtone()
- "Global Tone Detection"(Voice Features Guide for Windows 2000)

Event Retrieval :

- vpm_getevt()
- DX_CST data structure

· **sr_getevtdatap()(int the Standard Runtime Library Programmer' s Guide for
Windows 2000)**

vpm_enbtone()

enables detection of TONE ON

Name : int vpm_enbtone(chdev,toneid,evt_mask)

Inputs : int chdev : VPM channel device handle
int toneid : tone template
int evt_mask : event mask

Returns :

0 :

-1 :

Includes: srllib.h, smartsdk.h

Category: Global Tone Detection



vpm_enbtone() channel user defined tone tone-on,tone-off
event detection 가 . vpm_addtone() 가
user-defined tone detection 가 .

, tone-on,tone-off event 가
vpm_addtone() .
vpm_distone() disable tone enable , .

Parameter .

chdev : vpm_open() VPM channel device handle

toneid : user defined tone ID , channel
user-defined tone , toneid TONEALL

evtmask : going on event going off event

parameter BIT masking .

DM_TONEON : TONE ON detection

DM_TONEOFF : TONE OFF detection

evt_mask tone template enable/disable ,

vpm_distone(),vpm_enbtone .



Example

#include <stdio.h>


```

#include <errno.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
#define TID_1 101
main()
{
int vpmdev;
/*
* Open the Voice Channel Device and Enable a Handler
*/
if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
    perror( "vpmB1C1" );
    exit( 1 );
}
/*
* Describe a Simple Dual Tone Frequency Tone of 950-
* 1050 Hz and 475-525 Hz using leading edge detection.
*/
if ( vpm_blddt( TID_1, 1000, 50, 500, 25, TN_LEADING ) == -1 ) {
    printf( "Unable to build a Dual Tone Template\n" );
}
/*
* Bind the Tone to the Channel
*/
if ( vpm_addtone( vpmdev, NULL, 0 ) == -1 ) {
    printf( "Unable to Bind the Tone %d\n", TID_1 );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSGP( vpmdev ) );
    vpm_close( vpmdev );
    exit( 1 );
}
/*
* Enable Detection of Toneld TID_1
*/
if ( vpm_enbtone( vpmdev, TID_1, DM_TONEON | DM_TONEOFF ) == -1 ) {

```



```

        printf( "Unable to Enable Detection of Tone %d\n", TID_1 );
        printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSGP( vpmdev ) );
        vpm_close( vpmdev );
        exit( 1 );
    }
    /*
    * Continue Processing
    * .
    * .
    * .
    */
    /*
    * Close the opened Voice Channel Device
    */
    if ( vpm_close( vpmdev ) != 0 ) {
        perror( "close" );
    }
    /* Terminate the Program */
    exit( 0 );
}

```



Errors

가 -1 , ATDV_LASTERR() ATDV_ERRMSGP()
 Error .

EDX_BADPARAM : Invalid Parameter

EDX_SYSTEM : Windows 2000 System error – check errno

EDX_TNMSGSTATUS : Invalid Message status setting

EDX_TONEID : Bad tone ID



General one Detection functions :

- **vpm_addtone()**
- **vpm_blddt(),vpm_bldst(),vpm_blddtcad(),vpm_bldstcad()**
- **vpm_enbtone()**
- **"Global Tone Detection"(Voice Features Guide for Windows 2000)**

Event Retrieval :

- **vpm_getevt()**
- **DX_CST data structure**
- **sr_getevtdatap()(int the Standard Runtime Library Programmer' s Guide for Windows 2000)**

closes the file associated with the handle

Category: File Management

vpm_fileclose() vpm_fileopen() open file handle close

. Microsoft Visual C++ Run-Time Library

_close() .

```
_close, vpm_fileclose()
```

```
if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
```



```

/* process error */
}
/* set up DX_IOTT */
iott.io_type = IO_DEV|IO_EOT;
iott.io_bufp = 0;
iott.io_offset = 0;
iott.io_length = -1; /* play till end of file */
if((iott.io_handle = vpm_fileopen("prompt.vox",
O_RDONLY|O_BINARY)) == -1) {
/* process error */
}

/* set up DV_TPT */
vpm_clrtpt(&tpt,1);
tpt.tp_type = IO_EOT; /* only entry in the table */
tpt.tp_termno = DX_MAXDTMF; /* Maximum digits */
tpt.tp_length = 4; /* terminate on four digits */
tpt.tp_flags = TF_MAXDTMF; /* Use the default flags */
/* clear previously entered digits */
if (vpm_clrdigbuf(chdev) == -1) {
/* process error */
}
/* Now play the file */
if (vpm_play(chdev,&iott,&tpt,EV_SYNC) == -1) {
/* process error */
}
/* get digit using vpm_getdig( ) and continue processing. */
.
.

if (vpm_fileclose(iott.io_handle) == -1) {
/* process error */
}

```

Errors

가 -1, errno invalid file-handle parameter

EBADF .



- **vpm_fileopen()**
- **vpm_fileseek()**
- **vpm_fileread()**
- **vpm_filewrite()**

vpm_fileopen()

opens the file specified by filep

Name : int vpm_fileopen(filep,flags,pmode)

Inputs : const char *filep : filename

int flags : Operation Type

int pmode : Permission mode

Returns :

file handle :

-1 :

Category: File Management



vpm_fileopen() filep file open , flags field

Microsoft Visual C++ Run-Time Library

_open()



_open , vpm_fileopen()

■ Example

/* Play a voice file. Terminate on receiving 4 digits or at end of file*/

#include <fcntl.h>

#include <srllib.h>

#include <smartsdk.h>

#include <windows.h>

main()

{

int chdev;

DX_IOTT iott;

DV_TPT tpt;

DV_DIGIT dig;

.

.

/* Open the device using vpm_open(). Get channel device descriptor in

* chdev.


```

*/
if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
/* process error */
}
/* set up DX_IOTT */
iott.io_type = IO_DEV|IO_EOT;
iott.io_bufp = 0;
iott.io_offset = 0;
iott.io_length = -1; /* play till end of file */
if((iott.io_handle = vpm_fileopen("prompt.vox", O_RDONLY|O_BINARY)) == -1) {
/* process error */
}
/* set up DV_TPT */
vpm_clrtpt(&tpt,1);
tpt.tp_type = IO_EOT; /* only entry in the table */
tpt.tp_termno = DX_MAXDTMF; /* Maximum digits */
tpt.tp_length = 4; /* terminate on four digits */
tpt.tp_flags = TF_MAXDTMF; /* Use the default flags */
/* clear previously entered digits */
if (vpm_clrdigbuf(chdev) == -1) {
/* process error */
}
/* Now play the file */
if (vpm_play(chdev,&iott,&tpt,EV_SYNC) == -1) {
/* process error */
}
/* get digit using vpm_getdig( ) and continue processing. */
.
.
if (vpm_fileclose(iott.io_handle) == -1) {
/* process error */
}
}

```

■ Errors

가 -1 , errno
가 .

EACCES : Read only file write open ,
 path가 directory
 EEXIT : flag _O_CREAT _O_EXCL ,
 .
 EINVAL : Invalid flags or pmode argument
 EMFILE : file open open .
 ENOENT :file path 가 .

■ See Also

- vpm_fileclose()
- vpm_fileseek()
- vpm_fileread()
- vpm_filewrite()

vpm_fileread() turns number of bytes read by appliation

Name : int vpm_fileread(handle,buffer,count)

Inputs : int handle : vpm_fileopen() file handle
void *buffer : stroage location for data
unsigned int count : maximum number of bytes

Returns :

byte :

-1 :

Category: File Management

■

```

vpm_fileread()      application      byte      .      file
handle      file      count      buffer      .      file
count      byte 가 , text mode      file      open      ,
      byte      count      .
      Microsoft Visual C++ Run-Time Library
_read()      .

```

■

```

_read      , vpm_fileread()      .

```

■ Example

```

#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
int cd; /* channel descriptor */
DX_UIO myio; /* user definable I/O structure */
/*
* User defined I/O functions
*/
int my_read(fd,ptr,cnt)
int fd;
char * ptr;
unsigned cnt;

```



```

{
    printf("My read\n");
    return(vpm_fileread(fd,ptr,cnt));
}
/*
* my write function
*/
int my_write(fd,ptr,cnt)
int fd;
char * ptr;
unsigned cnt;
{
    printf("My write \n");
    return(vpm_filewrite(fd,ptr,cnt));
}
/*
* my seek function
*/
long my_seek(fd,offset,whence)
int fd;
long offset;
int whence;
{
    printf("My seek\n");
    return(vpm_fileseek(fd,offset,whence));
}
void main(argc,argv)
int argc;
char *argv[];
{
    .
    . /* Other initialization */
    .
    DX_UIO uioblk;
    /* Initialize the UIO structure */
    uioblk.u_read=my_read;

```



```

uioblk.u_write=my_write;
uioblk.u_seek=my_seek;
/* Install my I/O routines */
vpm_setuio(devhandle,uioblk);
vodat_fd = vpm_fileopen("JUNK.VOX",O_RDWR|O_BINARY);
/*This block uses standard I/O functions */
iott->io_type = IO_DEV|IO_CONT
iott->io_fhandle = vodat_fd;
iott->io_offset = 0;
iott->io_length = 20000;
/*This block uses my I/O functions */
iott++;
iott->io_type = IO_DEV|IO_UIO|IO_CONT
iott->io_fhandle = vodat_fd;
iott->io_offset = 20001;
iott->io_length = 20000;
/*This block uses standard I/O functions */
iott++;
iott->io_type = IO_DEV|IO_CONT
iott->io_fhandle = vodat_fd;
iott->io_offset = 20002;
iott->io_length = 20000;
/*This block uses my I/O functions */
iott->io_type = IO_DEV|IO_UIO|IO_EOT
iott->io_fhandle = vodat_fd;
iott->io_offset = 10003;
iott->io_length = 20000;
devhandle = vpm_open("vpmB1C1", 0);
vpm_sethook(devhandle, DX_ONHOOK,EV_SYNC)
vpm_wtrng(devhandle,1,DX_OFFHOOK,EV_SYNC);
vpm_clrdigbuf;
if(vpm_rec(devhandle,iott,(DX_TPT*)NULL,RM_TONE|EV_SYNC) == -1) {
perror("");
exit(1);
}
vpm_clrdigbuf(devhandle);

```



```

if(vpm_play(devhandle,iott,(DX_TPT*)NULL,EV_SYNC) == -1 {
perror("");
exit(1);
}
vpm_close(devhandle);

```

■ Errors

가 -1 , errno invalid file-handle , EBADF 가 .

■ See Also

- vpm_fileclose()
- vpm_fileopen()
- vpm_fileseek()
- vpm_filewrite()

vpm_fileseek() moves file pointer associated with handle

Name : long vpm_fileseek(handle,offset,origin)

Inputs : int handle : vpm_fileopen() file handle
 long offset : number of bytes from the origin
 int origin : initial position

Returns :

byte :

-1 :

Category: File Management

■

vpm_fileseek()	handle	file	Pointer	offset
	.			offset
byte	.			

Microsoft Visual C++ Run-Time Library

_lseek()

■

_lseek , vpm_fileseek()

■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
int cd; /* channel descriptor */
DX_UIO myio; /* user definable I/O structure */
/*
 * User defined I/O functions
 */
int my_read(fd,ptr,cnt)
int fd;
char * ptr;
unsigned cnt;
{
```



```

        printf("My read\n");
        return(vpm_fileread(fd,ptr,cnt));
    }
    /*
    * my write function
    */
    int my_write(fd,ptr,cnt)
    int fd;
    char * ptr;
    unsigned cnt;
    {
        printf("My write \n");
        return(vpm_filewrite(fd,ptr,cnt));
    }
    /*
    * my seek function
    */
    long my_seek(fd,offset,whence)
    int fd;
    long offset;
    int whence;
    {
        printf("My seek\n");
        return(vpm_fileseek(fd,offset,whence));
    }
    void main(argc,argv)
    int argc;
    char *argv[];
    {
        .
        . /* Other initialization */
        .
        DX_UIO uioblk;
        /* Initialize the UIO structure */
        uioblk.u_read=my_read;
        uioblk.u_write=my_write;

```



```

uioblk.u_seek=my_seek;
/* Install my I/O routines */
vpm_setuio(devhandle,uioblk);
vodat_fd = vpm_fileopen("JUNK.VOX",O_RDWR|O_BINARY);
/*This block uses standard I/O functions */
iott->io_type = IO_DEV|IO_CONT
iott->io_fhandle = vodat_fd;
iott->io_offset = 0;
iott->io_length = 20000;
/*This block uses my I/O functions */
iott++;
iott->io_type = IO_DEV|IO_UIO|IO_CONT
iott->io_fhandle = vodat_fd;
iott->io_offset = 20001;
iott->io_length = 20000;
/*This block uses standard I/O functions */
iott++
iott->io_type = IO_DEV|IO_CONT
iott->io_fhandle = vodat_fd;
iott->io_offset = 20002;
iott->io_length = 20000;
/*This block uses my I/O functions */
iott->io_type = IO_DEV|IO_UIO|IO_EOT
iott->io_fhandle = vodat_fd;
iott->io_offset = 10003;
iott->io_length = 20000;
devhandle = vpm_open("vpmB1C1", NULL);
vpm_sethook(devhandle, DX_ONHOOK,EV_SYNC)
vpm_wtring(devhandle,1,DX_OFFHOOK,EV_SYNC);
vpm_clrdigbuf;
if(vpm_rec(devhandle,iott,(DX_TPT*)NULL,RM_TONE|EV_SYNC) == -1) {
perror("");
exit(1);
}
vpm_clrdigbuf(devhandle);
if(vpm_play(devhandle,iott,(DX_TPT*)NULL,EV_SYNC) == -1 {

```



```

perror("");
exit(1);
}
vpm_close(devhandle);

```

■ Errors

가 -1, errno

EBADF : invalid file-handle parameter, closed file, locked file

EINVAL : origin value 가 , offset position file

■ See Also

- vpm_fileclose()
- vpm_fileopen()
- vpm_fileread()
- vpm_filewrite()

vpm_filewrite() writes count bytes from buffer into files associated with handle

Name : long vpm_filewrite(handle,t,origin)

Inputs : int handle : vpm_fileopen() file handle
 void *buffer : stroage location for data
 unsigned int count : maximum number of bytes

Returns :

byte :

-1 :

Category: File Management



vpm_filewrite()	handle	file	buffer	count	.
write operation		file pointer			.
	Append	open		, write operation	
	. write operation	file pointer		byte	가

Microsoft Visual C++ Run-Time Library
 _write()



_write , vpm_filewrite()

■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
int cd; /* channel descriptor */
DX_UIO myio; /* user definable I/O structure */
/*
 * User defined I/O functions
 */
int my_read(fd,ptr,cnt)
int fd;
char * ptr;
```



```

unsigned cnt;
{
    printf("My read\n");
    return(vpm_fileread(fd,ptr,cnt));
}
/*
my write function
*/
int my_write(fd,ptr,cnt)
int fd;
char * ptr;
unsigned cnt;
{
    printf("My write \n");
    return(vpm_filewrite(fd,ptr,cnt));
}
/*
* my seek function
*/
long my_seek(fd,offset,whence)
int fd;
long offset;
int whence;
{
    printf("My seek\n");
    return(vpm_fileseek(fd,offset,whence));
}
void main(argc,argv)
int argc;
char *argv[];
{
    .
    . /* Other initialization */
    .
    DX_UIO uioblk;
    /* Initialize the UIO structure */

```



```

uioblk.u_read=my_read;
uioblk.u_write=my_write;
uioblk.u_seek=my_seek;
/* Install my I/O routines */
vpm_setuio(devhandle,uioblk);
vodat_fd = vpm_fileopen("JUNK.VOX",O_RDWR|O_BINARY);
/*This block uses standard I/O functions */
iott->io_type = IO_DEV|IO_CONT
iott->io_fhandle = vodat_fd;
iott->io_offset = 0;
iott->io_length = 20000;
/*This block uses my I/O functions */
iott++;
iott->io_type = IO_DEV|IO_UIO|IO_CONT
iott->io_fhandle = vodat_fd;
iott->io_offset = 20001;
iott->io_length = 20000;
/*This block uses standard I/O functions */
iott++
iott->io_type = IO_DEV|IO_CONT
iott->io_fhandle = vodat_fd;
iott->io_offset = 20002;
iott->io_length = 20000;
/*This block uses my I/O functions */
iott->io_type = IO_DEV|IO_UIO|IO_EOT
iott->io_fhandle = vodat_fd;
iott->io_offset = 10003;
iott->io_length = 20000;
devhandle = vpm_open("vpmB1C1", NULL);
vpm_sethook(devhandle, DX_ONHOOK,EV_SYNC)
vpm_wtring(devhandle,1,DX_OFFHOOK,EV_SYNC);
vpm_clrdigbuf;
if(vpm_rec(devhandle,iott,(DX_TPT*)NULL,RM_TONE|EV_SYNC) == -1) {
perror("");
exit(1);
vpm_clrdigbuf(devhandle);

```



```

if(vpm_play(devhandle,iott,(DX_TPT*)NULL,EV_SYNC) == -1 {
perror("");
exit(1);
}
vpm_close(devhandle);

```

■ Errors

가 -1 , errno

EBADF : invalid file-handle parameter, file write open .
 ENOSPC : write operation .

■ See Also

- vpm_fileclose()
- vpm_fileopen()
- vpm_fileseek()
- vpm_fileread()

vpm_getcursv() returns the specified channel' s current speed

Name : long vpm_getcursv(chdev,curvolp,curspeedp)
Inputs : int chdev : VPM channel device handle
int *curvolp : volume pointer
int *curspeedp : speed가 pointer
Returns :
0 :
-1 :
Includes: srllib.h, smartsdk.h
Category: Speed and Volume

■

vpm_getcursv() channel speed volume
curvolp curspeedp . vpm_getcursv()
가 DTMF ditit speed volume ,
.
Play speed volume DTMF digit vpm_setsvcond()
vpm_addspddig(),vpm_addvoldig() .
Parameter .

chdev : vpm_open() VPM channeld device handle
curvolp : channel volume value
pointer -40dB 20dB 가 .
curspeedp : channel speed value
pointer -50% - 100% 가 .

■

■ **Example**

```
#include <stdio.h>
#include <errno.h>
#include <srllib.h>
#include <smartsdk.h>
```



```

#include <windows.h>
/*
* General Variables
*/
main()
{
int vpmdev;
int curspeed, curvolume;
/*
* Open the Voice Channel Device and Enable a Handler
*/
if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
    perror( "vpmB1C1" );
    exit( 1 );
}
/*
* Get the Current Volume and Speed Settings
*/
if ( vpm_getcursv( vpmdev, &curvolume, &curspeed ) == -1 ) {
    printf( "Unable to Get the Current Speed and" );
    printf( " Volume Settings\n" );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSG( vpmdev ) );
    vpm_close( vpmdev );
    exit( 1 );
} else {
    printf("Volume = %d Speed = %d\n", curvolume, curspeed );
}
/*
* Continue Processing
* .
* .
* .
*/
/*
* Close the opened Voice Channel Device

```



```

*/
if ( vpm_close( vpmdev ) != 0 ) {
    perror( "close" );
}
/* Terminate the Program */
exit( 0 );
}

```

■ Errors

가 -1, ATDV_LASTERR() ATDV_ERRMSGP()
Error .

EDX_BADPARAM : Invalid Parameter

EDX_SYSTEM : Windows 2000 System error – check errno



Related to Speed and Volume :

- vpm_adjsv()
- vpm_addspddig()
- vpm_addvoldig()
- vpm_setsvmt()
- vpm_getsvmt()
- vpm_setsvcond()
- vpm_clrsvcond()
- "Speed and Volume Modification Tables"(Voice Features Guide for Windows NT)
- DX_SVMT structure(Chapter 4.Voice Data Structure and Device Parameters)

vpm_getdig() initiates the collection of digits

Name : int vpm_getdig(chdev,tptp,digitp,mode)
Inputs : int chdev : VPM channel device handle
DV_TPT *tptp : Termination Parameter Structure Pointer
DV_DIGIT *digitp : User Digit Buffer Structure Pointer
unsigned short mode : /
Returns :
0 : initial
digit : digit 1
-1 :
Includes: srllib.h, smartsdk.h
Category: I/O
Mode: synchronous/asynchronous

■
vpm_getdig() open channel digit buffer digit
. 가 digit ASCIIZ format DV_DIGIT
structure local buffer .
digit type vpm_addtone() digit detection
.

vpm_addtone() chapter , DV_DIGIT
structure 4 Voice Data Structure Device Parameter .

termination DV_TPT structure , structure
tptp parameter .

■ Asynchronous Operation

mode field EV_ASYNC .
가 0
, digit termination event
.
termination event SRL event Management Function
.
Event Management Function Appendix A

digit TDX_GETDIG Event
vpm_getdig()가 , VPMX_TERMMSK()

■ Synchronous Operation

digit 0
digit 가
VPMX_TERMMSK()

Parameter

chdev : vpm_open() VPM channel device handle

tptp : Termination Termination Parameter Table
Structure pointer
Termination
DX_MAXDTMF : Maximum number of digits received
DX_MAXSIL : Maximum silence
DX_MAXNOSIL : Maximum non-silence
DX_IDDTIME : Inter-digit delay
DX_MAXTIME : Function Time
DX_DIGMASK : Digit mask termination
DX_TONE : Tone-off or Tone-on detection

digitp : digit digit type digit bufferstructure
pointer digits digit type array DV_DIGIT
structure digit type
DG_DTMF : DTMF digit
DG_USER1 : User defined digit
DG_USER2 : User defined digit
DG_USER3 : User defined digit
DG_USER4 : User defined digit
DG_USER5 : User defined digit

mode : vpm_getdig()
가
EV_ASYNC :
EV_SYNC :

digit buffer 31 FIFO . digit
buffer digit overwrite vpm_clrdigibuf() digit
. DG_MAXDIGS vpm_getdig() 가 digit
define .
NOTE : vpm_getdigbuf() 31 digit ,
digit .



1. vpm_setsvcond() , speed volume digit
vpm_getdig() . termination .
digit termination speed/volume
speed/volume Control digit .
2. 가 , digit buffer duration
.
3. channel idle . EDX_BUSY
.
4. 가 , buffer digit , return
value 1 , digit buffer가 null termination .



Example 1 : Using vpm_getdig() in synchronous mode

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
DV_TPT tpt[3];
DV_DIGIT digp;
int chdev, numdigs, cnt;
/* open the channel with vpm_open( ). Obtain channel device descriptor
* in chdev
*/
if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
/* process error */
}
```



```

/* initiate the call */
.
.
/* Set up the DV_TPT and get the digits */
vpm_clrtpt(tpt,3);
tpt[0].tp_type = IO_CONT;
tpt[0].tp_termno = DX_MAXDTMF; /* Maximum number of digits */
tpt[0].tp_length = 4; /* terminate on 4 digits */
tpt[0].tp_flags = TF_MAXDTMF; /* terminate if already in buf. */
tpt[1].tp_type = IO_CONT;
tpt[1].tp_termno = DX_LCOFF; /* LC off termination */
tpt[1].tp_length = 3; /* Use 30 ms (10 ms resolution
* timer) */
tpt[1].tp_flags = TF_LCOFF|TF_10MS; /* level triggered, clear history,
* 10 ms resolution */
tpt[2].tp_type = IO_EOT;
tpt[2].tp_termno = DX_MAXTIME; /* Function Time */
tpt[2].tp_length = 100; /* 10 seconds (100 ms resolution
* timer) */
tpt[2].tp_flags = TF_MAXTIME; /* Edge-triggered */
/* clear previously entered digits */
if (vpm_clrdigbuf(chdev) == -1) {
/* process error */
}
if ((numdigs = vpm_getdig(chdev,tpt, &digp, EV_SYNC)) == -1) {
/* process error */
}
for (cnt=0; cnt < numdigs; cnt++) {
    printf("\nDigit received = %c, digit type = %d",
        digp.dg_value[cnt], digp.dg_type[cnt]);
}
/* go to next state */
.
.
}

```


■ Example 2 : Using vpm_getdig() in asynchronous mode

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
#define MAXCHAN 24
int digit_handler();
DV_TPT stpt[3];
DV_DIGIT digp[256];
main()
{
    int i, chdev[MAXCHAN];
    char *chnamep;
    for (i=0; i<MAXCHAN; i++) {
        /* Set chnamep to the channel name - e.g., vpmB1C1 */
        /* open the channel with vpm_open( ). Obtain channel device
        * descriptor in chdev[i]
        */
        if ((chdev[i] = vpm_open(chnamep, NULL)) == -1) {
            /* process error */
        }
        /* Using sr_enbhdr(), set up handler function to handle vpm_getdig()
        * completion events on this channel.
        */
        if (sr_enbhdr(chdev[i], TDX_GETDIG, digit_handler) == -1) {
            /* process error */
        }
        /* initiate the call */
        .
        .
        /* Set up the DV_TPT and get the digits */
        vpm_clrtpt(tpt, 3);
        tpt[0].tp_type = IO_CONT;
        tpt[0].tp_termno = DX_MAXDTMF; /* Maximum number of digits */
        tpt[0].tp_length = 4; /* terminate on 4 digits */
        tpt[0].tp_flags = TF_MAXDTMF; /* terminate if already in buf*/
    }
```



```

tpt[1].tp_type = IO_CONT;
tpt[1].tp_termno = DX_LCOFF; /* LC off termination */
tpt[1].tp_length = 3; /* Use 30 ms (10 ms resolution
* timer) */
tpt[1].tp_flags = TF_LCOFF|TF_10MS; /* level triggered, clear
* history, 10 ms resolution */
tpt[2].tp_type = IO_EOT;
tpt[2].tp_termno = DX_MAXTIME; /* Function Time */
tpt[2].tp_length = 100; /* 10 seconds (100 ms resolution
* timer) */
tpt[2].tp_flags = TF_MAXTIME; /* Edge triggered */
/* clear previously entered digits */
if (vpm_clrdigbuf(chdev[i]) == -1) {
/* process error */
}
if (vpm_getdig(chdev[i], tpt, &digp[chdev[i]], EV_ASYNC) == -1) {
/* process error */
}
}
/* Use sr_waitevt() to wait for the completion of vpm_getdig().
* On receiving the completion event, TDX_GETDIG, control is transferred
* to the handler function previously established using sr_enbhdlr().
*/
.
.
}
int digit_handler()
{
int chfd;
int cnt, numdigs;
chfd = sr_getevtdev();
numdigs = strlen(digp[chfd].dg_value);
for(cnt=0; cnt < numdigs; cnt++) {
printf("\nDigit received = %c, digit type = %d",
digp[chfd].dg_value[cnt], digp[chfd].dg_type[cnt]);
}
}

```



```

/* Kick off next function in the state machine model. */
.
.
return 0;
}

```

■ Errors

가 -1 , ATDV_LASTERR() ATDV_ERRMSGP()
 Error .

EDX_BADPARM : Invalid Parameter

EDX_BADTPT : Invalid DV_TPT entry

EDX_BUSY : Channel busy

EDX_SYSTEM : Windows 2000 System error – check errno



Setting User-Defined Digits :

- vpm_addtone()

Collectiong Digits :

- DV_DIGIT

- vpm_sethook()

used to synchronously monitor channels

Name : int vpm_getevt(chdev, eblkp, timeout)

Inputs : int chdev : VPM channel device handle
 DX_EBLK *eblp : Event Block Structure Pointer
 int timeout : timeout value

Returns :

0:

-1 :

Includes: srllib.h, smartsdk.h

Category: Call Status Transition Event

■

vpm_getevt() vpm_setevtmask() Call Status

Transition Event .

vpm_getevt() vpm_setevtmask() event mask

channel event , return .

DX_EBLK structure event .

Parameter .

```

chdev :    vpm_open()                VPM channelled device handle
eblkp :                event                Event Block Structure    DX_EBLK
        pointer
timeout :    가
                가 .
        # of seconds : vpm_getevt 가 ,
                error .
        -1 : vpm_getevt()    event가    block .
                time out .
        0 :                event                -1 .
NOTE :    timeout    block    timeout    vpm_getevt()
        , error    error    EDX_TIMEOUT

```




vpm_getevt()	channel	,	process	.
	Process	channel	vpm_getevt	
vpm_getevt()	가	event	.	

■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
int chdev; /* channel descriptor */
int timeout; /* timeout for function */
DX_EBLK eblk; /* Event Block Structure */
.
.
.
/* Open Channel */
if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
/* process error */
}
/* Set RINGS or WINK as events to wait on */
if (vpm_setevtmask(chdev,DM_RINGS|DM_WINK) == -1) {
/* process error */
}
/* Set timeout to 5 seconds */
timeout = 5;
if (vpm_getevt(chdev,&eblk,timeout) == -1){
/* process error */
if (ATDV_LASTERR(chdev) == EDX_TIMEOUT) { /* check if timed out */
printf("Timed out waiting for event.\n");
}
else {
/* further error processing */
.

```



```

·
}
}
switch (eblk.ev_event) {
case DE_RINGS:
    printf("Ring event occurred.\n");
    break;
case DE_WINK:
    printf("Wink event occurred.\n");
    break;
}
·
·
}

```

■ Errors

가 -1 , ATDV_LASTERR() ATDV_ERRMSGP()
Error .

EDX_BADPARAM : Invalid Parameter

EDX_SYSTEM : Windows 2000 System error – check errno

EDX_TIMEOUT :Timeout time is reached



- **vpm_setevtmsk()**
- **DX_EBLK(chapter 4. Voice Data Structures and Device Parameters)**

vpm_getparm() obtains the current parameter settings

Name : int vpm_getparm(dev,param.valuep)

Inputs : int dev : VPM channel device handle board handle
 unsigned long parm : parameter type
 void *valuep : parameter value가 pointer

Returns :

0 :

-1 :

Includes: srl.lib.h, smartsdk.h

Category: Configuration



vpm_getparm() open device parameter .
 vpm_getparm() parameter .
 channel idle .(, I/O function
)

Parameter .

dev : vpm_open() VPM channel device handle Board handle

parm : parameter type define , valuep .

Board Channel paramter type define default

Table10 , Table 11 .

valuep : parm parameter pointer

NOTE : valuep void *cast

valuep parameter size 가 .

parameter size dxllib.h , define

parameter size PM_SHORT,PM_BYTE,PM_INT,PM_LONG 가

, parameter type short .



Parameter vpm_getparm()
 variable Clear . address가

parameter

가

.

■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    int bddev;
    unsigned short parmval;
    /* open the board using vpm_open( ). Obtain board device descriptor in
    * bddev
    */
    if ((bddev = vpm_open("vpmB1",NULL)) == -1) {
        /* process error */
    }
    parmval = 0; /* CLEAR parmval */
    /* get the number of channels on the board. DXBD_CHNUM is of type
    * unsigned short as specified by the PM_SHORT define in the definition
    * for DXBD_CHNUM in dxxplib.h. The size of the variable parmval is
    * sufficient to hold the value of DXBD_CHNUM.
    */
    if (vpm_getparm(bddev, VPMBD_CHNUM, (void *)&parmval) == -1) {
        /* process error */
    }
    printf("\nNumber of channels on board = %d",parmval);
    .
    .
}
```

■ Errors

가 -1 , ATDV_LASTERR() ATDV_ERRMSGP()
Error .

EDX_BADPARAM : Invalid Parameter

EDX_SYSTEM : Windows 2000 System error – check errno

EDX_BUSY : Channel is busy



- **vpm_setparm()**

vpm_getsvmt() returns contents of Speed or volume Modification Table

Name : int vpm_getsvmt(chdev,tabletype,svmtp)

Inputs : int chdev : VPM channel device handle
unsigned short tabletype : table(speed and volume table)
DX_SVMT *svmtp : DX_SVMT structure pointer

Returns :

0 :

-1 :

Includes: srllib.h, smartsdk.h

Category: Speed and Volume



vpm_getsvmt() DX_SVMT structure speed volume modification Table

Speed Volume Modification table Voice Feature Guide

DX_SVMT structure

Parameter

dev : vpm_open() VPM channeled device handle

tabletype : speed modification table volume modification table

table 가 .

SV_SPEEDTBL : Speed Modification Table

SV_VOLUMETBL : Volume Modification Table

svmtp : speed,volume modification table DX_SVMT structure pointer



■ Example

```
#include <stdio.h>
```

```
#include <errno.h>
```

```
#include <srllib.h>
```

```
#include <smartsdk.h>
```

```
#include <windows.h>
```



```

/*
* General Variables
*/
main()
{
    DX_SVMT svmt;
    int vpmdev, index;
/*
* Open the Voice Channel Device and Enable a Handler
*/
if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
    perror( "vpmB1C1" );
    exit( 1 );
}
/*
* Get the Current Volume Modification Table
*/
memset( &svmt, 0, sizeof( DX_SVMT ) );
if ( vpm_getsvmt( vpmdev, SV_VOLUMETBL, &svmt ) == -1 ){
    printf( "Unable to Get the Current Volume" );
    printf( " Modification Table\n" );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSG( vpmdev ) );
    vpm_close( vpmdev );
    exit( 1 );
} else {
    printf( "Volume Modification Table is:\n" );
    for ( index = 0; index < 10; index++ ) {
        printf( "decrease[ %d ] = %d\n", index,
            svmt.decrease[ index ] );
    }
    printf( "origin = %d\n", svmt.origin );
    for ( index = 0; index < 10; index++ ) {
        printf( "increase[ %d ] = %d\n", index,
            svmt.increase[ index ] );
    }
}
}

```



```

}
/*
* Continue Processing
* .
* .
* .
*/
/*
* Close the opened Voice Channel Device
*/
if ( vpm_close( vpmdev ) != 0 ) {
    perror( "close" );
}
/* Terminate the Program */
exit( 0 );
}

```

■ Errors

가 -1 , ATDV_LASTERR() ATDV_ERRMSGP()
Error .

EDX_BADPARAM : Invalid Parameter

EDX_SYSTEM : Windows 2000 System error – check errno

EDX_SPDVOL : table type SV_SPEEDTBL SV_VOLUMETBL



- vpm_addspddig()
- vpm_addvoldig()
- vpm_adjsv()
- vpm_clrsvcond()
- vpm_getcursv()
- vpm_setsvcond()
- vpm_setsvmt()
- "Speed and Volume Modification Tables"(Voice Features Guide for Windows NT)
- DX_SVMT(Chapter 4. Voice Data Structures and Device Parameters)

vpm_gtcallid()

Name:	int gtcallid (chdev , *buffer))		
Input:	int chdev	channel device handle	
	Char *buffer	call id	buffer
Returns:	0 if success		
	-1 if failure		
Includes:	srllib.h		
	smartsdk.h		



buffer Call ID 가 . Buffer MDMF
Data (MDMF .)

Parameter	Description
chdev:	vpm_open() valid channel device handle
Buffer:	call id buffer .

■ **Example**

#include “ smartSdk.h”

```
void main()
{
    int ret;
    int dev;

    If((dev = vpm_open(“ vpmB1C1,NULL)) == -1){
        Return;
    }

    printf(“ \n channel open = %d” ,dev);

    int value = 1;
    if(vpm_setparm(dev, VPM_CALLID, (void *)&value) < 0){
        printf(“ v[pm_setparm error\n”];
    }
}
```



```

    }
    value = 2;
    if(vpm_setparm(dev, VPMCH_RINGCNT ,(void *)&value) < 0{
        printf(" vpm_setparm error\n");
    }
    if(vpm_setparm(dev, DM_RINGS) == -1)
    {
        ;//ERROR Handling
    }
    vpm_sethook(dev,DX_ONHOOK, EV_SYNC);

    printf("\n ring wait .. %d ch\n",dev);

    unsigned char buffer[256];
    memset(buffer,0,sizeof(buffer));

    DX_EBLK eblk;
    if(vpm_getevt(dev,&ebk,-1) != -1){
        if(eblk.ev_event = DM_RINGS){
            Printf(" Ring event occurrence.\n");
        }
    }
    if(vpm_gtcallid(dev,buffer) <0){
        printf(" vpm_gtcallid ID Error\n");
    }
    else
        printf(" Caller ID : %s\n", buffer);
    if(vpm_wtring(dev,1,DX_OFFHOOK, -1) == -1){
        printf("\n dl wtring error");
        return;
    }
    printf("\n OFFHOOK %d ch \n,dev);
}

```



vpm_gtextcallid()

vpm_wtcallid()

"Speed and Volume Modification Tables" (*Voice Features Guide for Windows 2000*)

DX_SVMT (Chapter 4. Voice Data Structures and Device Parameters)

MDMF

Data

TYPE	BINARY VALUE	HEXADECIMAL VALUE
Message Type(MDMF)	1000 0000	80H
hMessage Length	0001 1111	1FH
Parameter Type(Date/Time)	0000 0001	01H
Parameter Length	0000 1000	08H
Date&Time (3 21 2 05 PM)	0011 0000	30H
	0011 0011	33H
	0011 0010	32H
	0011 0001	31H
	0011 0001	31H
	0011 0100	34H
	0011 0000	30H
	0011 0101	35H
Parameter Type(Number)	0000 0010	02H
Parameter Length	0000 1010	0AH
CLID (914-555-1234)	0011 1001	39H
	0011 0001	31H
	0011 0100	34H
	0011 0101	35H
	0011 0101	35H
	0011 0101	35H
	0011 0001	31H
	0011 0010	32H
	0011 0011	33H
	0011 0100	34H
Parameter Type(Name)	0000 0111	07H
Parameter Length	0000 0111	07H
Name (DOE JOE)	0100 0100	44H
	0100 1111	4FH
	0100 1010	45H
	0100 0000	20H
	0100 1010	4AH
	0100 1111	4FH
	0100 0101	45H
Checksum	1101 0110	D6H

vpm_gtextcallid()

Name:	int vpm_gtextcallid (chdev , infotype , buffer);
Input:	int chdev channel device handle
	Char infotype message
	Char buffer buffer
Returns:	0 if success -1 if failure
Includes:	srllib.h smartsdk.h



message 가 MDM(multi data message)

message

Parameter	Description
Int chdev	channel device handle
Char infotype	message
	parameter .
	• MCLASS_DN : Callid ID 가
	• MCLASS_NAME : Caller NAME 가
	• MCLASS_DATETIME : Date/Time 가
Char buffer	message buffer .

■ Example

#include " smartSdk.h"

```
void main()
{
    int ret;
    int dev;
    If((dev = vpm_open(" vpmB1C1" ,NULL)) == -1){
        Return;
```



```

}
printf("\n channel open = %d",dev);

int value = 1;
if(vpm_setparm(dev, VPM_CALLID, (void *)&value) < 0){
printf(" v[pm_setparm error\n" ];
}
value = 2;
if(vpm_setparm(dev, VPMCH_RINGCNT ,(void *)&value) < 0{
printf(" vpm_setparm error\n" );
}
if(vpm_setparm(dev, DM_RINGS) == -1)
{
        ;//ERROR Handling
}
vpm_sethook(dev,DX_ONHOOK, EV_SYNC);

printf("\n ring wait .. %d ch\n", dev);

unsigned char buffer[256];
memset(buffer,0,sizeof(buffer));

DX_EBLK eblk;
If(vpm_getevt(dev,&eblk, -1){
Printf(" Ring event occurrence.\n" );
}
}

int infotype = CLIDINFO_RFAMETYPE;

if(vpm_gtextcallid(dev,infotype,buffer) <0){
printf(" vpm_gtextcallid error\n" );
}
if(infotype = CLIDINFO_FRAMETYPE){
if(buffer[0] == CLASSFRAME_MDM){
//Get Caller ID
if(vpm_gtextcallid(dev , MCALSS_DN, buffer) != -1){

```



```

        printf(" Callid ID = %s" , buffer);
    }

    /Get Caller Name
    if(vpm_gtextcallid(dev , MCALSS_NAME,buffer) != -1){
        printf("\nCaller NAME = %s, buffer);

        //Get Date and Time
        if(vpm_gtextcallid(dev , MCLASS_DATETIME, buffer) != -1){
            printf("\n Data/Time = &s" , buffer);
        }
    }
    else{
        printf(" Not MDM Message" );
    }
}
else{
    printf(" Caller ID : %s" , buffer);
}
if(vpm_wtring(dev, 1, DX_OFFHOOK, -1) == -1){
    printf("\n dl wtring error");
    return;
}
printf("\n OFFHOOK %dch \n" ,dev);

}

```



vpm_gtcallid()

vpm_wtcallid()

"Speed and Volume Modification Tables" (*Voice Features Guide for Windows 2000*)

DX_SVMT (Chapter 4. Voice Data Structures and Device Parameters)

vpm_initcallp() initializes and activates PerfectCall Call Analysis

Name : int vpm_initcallp(chdev)

Inputs : int chdev : VPM channel device handle

Returns :

0 :

-1 :

Includes: srllib.h, smartsdk.h

Category: PerfectCall Call Analysis



vpm_initcallp() chdev channel PerfectCall Call Analysis
 , . Call Analysis Tone
 Global Tone Detection Template .

PerfectCall Call Analysis vpm_initcallp()
 vpm_dial() .
 vpm_dial() vpm_initcallp() Call Analysis
 channel Basic Mode .

PerfectCall Call Analysis 가 dial tone, 2 busy
 signals, ringback fax modem tone . Call
 , live voice answering machine 가
 . vpm_initcallp()가 , tone channel
 .

VPM DLL signal default
 . Application vpm_chgdur(),vpm_chgfreq(),vpm_chrrepcnt()
 . vpm_initcallp() tone 가
 , channel .

channel tone ,
 vpm_deltone() tone 가 ,
 vpm_initcallp() 가 . vpm_deltone()
 PerfectCall call Analysismf 가 .

NOTE : vpm_deltone() vpm_initcallp() channel
 tone use defined tone .

Parameter	Description
chdev:	specifies the channel device handle.

■ Example

```
#include <stdio.h>

#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>

main()
{
    DX_CAP cap_s;
    int ddd, car;
    char *chnam, *dialstrg;

    chnam = "vpmB1C1";
    dialstrg = "L1234";

    /*
     * Open channel
     */
    if ((ddd = vpm_open( chnam, NULL )) == -1 ) {
        /* handle error */
    }

    /*
     * Delete any previous tones
     */
    if ( vpm_deltone(ddd) < 0 )
        /* handle error */
    }
```



```

/*
    * Change Enhanced call progress default local dial tone
    */
    if (vpm_chgfreq( TID_DIAL_LCL, 425, 150, 0, 0 ) < 0) {
        /* handle error */
    }

/*
    * Change Enhanced call progress default busy cadence
    */
    if (vpm_chgdur( TID_BUSY1, 550, 400, 550, 400 ) < 0) {
        /* handle error */
    }

    if (vpm_chgrepcnt( TID_BUSY1, 4 ) < 0) {
        /* handle error */
    }

/*
    * Now enable Enhanced call progress with above changed settings.
    */

    if (vpm_initcallp( ddd )) {
        /* handle error */
    }

/* Set off Hook*/
    if ((vpm_sethook( ddd, DX_OFFHOOK, EV_SYNC )) == -1) {
        /* handle error */
    }

/* Dial*/
    if ((car = vpm_dial( ddd,dialstrg,(DX_CAP*)&cap_s,
        DX_CALLP|EV_SYNC))== -1) {
        /* handle error */
    }

```



```

switch( car ) {
case CR_NODIALTONE:
    printf(" Unable to get dial tone\n");
    break;

case CR_BUSY:
    printf(" %s engaged\n", dialstrg );
    break;

case CR_CNCT:
    printf(" Successful connection to %s\n", dialstrg );
    break;

default:
    break;
}

/* Set on Hook */
if ((vpm_sethook( ddd, DX_ONHOOK, EV_SYNC )) == -1) {
    /* handle error */
}

vpm_close( ddd );
}
■
        idle
        .

■
• vpm_chgdur( )
• vpm_chgfreq( )
• vpm_chgrepcnt( )
• vpm_deltone( )

```


vpm_open()

Category: PerfectCall Call Analysis

Process	child process	handle
, child process	device 가 open	.


```

#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    int chdev; /* channel descriptor */
    .
    .
    .
    /* Open Channel */
    if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
        /* process error */
    }
    .
    .
}

```

■ Errors

−1

error check .

EINVAL	Invalid Argument
EBADF	Invalid file descriptor
EINTR	A signal was caught
EIO	Error during a Windows 2000 STREAMS open

■ See Also

- vpm_close()

plays recorded voice data	vpm_play()
---------------------------	-------------

Name : int vpm_play(chdev,iottp,tptp,mode)

Inputs :

int chdev	valid SCT channel device handle
DX_IOTT *iottp	I/O Transfer Table Structure
DV_TPT *tptp	Termination Parameter Table Structure
unsigned short mode	play asynchronous/synchronous playing mode bit mask

Returns: 0 :
-1 :

Includes: srllib.h
smartsdk.h

Category: I/O

Mode : synchronous/asynchronous

■

vpm_play Voice data channel play . Voice data data file memory custom device . Play Voice data iottp 가 가 DX_IOTT structure , DX_IOTT structure 4 Voice data structure and Device Parameters .

NOTE : file play , vpm_play() . DX_IOTT structure setup 가 . vpm_play() description .

■ **Asynchronous Operation**

mode field EV_ASYNC . 0 event . play DV_TPT structure . DX_IOTT data 가 play DV_TPT play .

vpm_play 가 current channel
status information Extended Attribute function
play TDX_PLAY Event
vpm_play() VPMX_TERMMSK()
event SRL Event Management function

■ synchronous Operation

가
0
play DV_TPT structure DX_IOTT
data 가 play DV_TPT
play
play 0
vpm_play 가 , VPMX_TERMMSK()

The vpm_play()

Parameter Description

chdev :	vpm_open	open	valid
	channel device handle		
iott :	play voice data	가	I/O Transfer Table
	Structure DX_IOTT	pointer	
tpt :	play	Termination Parameter Table Structure	
	DV_TPT	pointer vpm_play()	DV_TPT

DX_MAXDTMF : Maximum number of digits received

DX_MAXSIL · Maximum silence

DX_MAXNOSIL · Maximum non-silence

DX_IDDTIME : Inter-digit delay

DX_MAXTIME : Function Time

DX_DIGMASK :Digit mask termination

DX_TONE · Tone-off or Tone-on detection

DV_TPT Structure

standard

Runtime Library

NOTE : DV_TPT

maximum byte count,

vpm_stopch() file

VPMX_TERMMSK

Mode : play mode async / sync mode play mode
parameter Bit mask

Choose one only:

EV_ASYNC : Run vpm_play() asynchronously.

EV_SYNC : Run vpm_play() synchronously (default).

PM_TONE : Play tone mode 가
tone . tone



vpm_play() 가

vpm_setsvcond()

vpm_adjsv()

가

가



Example 1: Using vpm_play() in synchronous mode.

/* Play a voice file. Terminate on receiving 4 digits or at end of file*/

#include <fcntl.h>

#include <srllib.h>

#include <smartsdk.h>

#include <windows.h>

main()

{

int chdev;

DX_IOTT iott;

DV_TPT tpt;

DV_DIGIT dig;

.

.

/* Open the device using vpm_open(). Get channel device descriptor in

* chdev.

*/


```

if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
    /* process error */
}
/* set up DX_IOTT */
iott.io_type = IO_DEV|IO_EOT;
iott.io_bufp = 0;
iott.io_offset = 0;
iott.io_length = -1; /* play till end of file */
if((iott.io_fhandle = vpm_fileopen("prompt.vox", O_RDONLY|O_BINARY))
    == -1) {
    /* process error */
}
/* set up DV_TPT */
vpm_clrtpt(&tpt,1);
tpt.tp_type = IO_EOT; /* only entry in the table */
tpt.tp_termno = DX_MAXDTMF; /* Maximum digits */
tpt.tp_length = 4; /* terminate on four digits */
tpt.tp_flags = TF_MAXDTMF; /* Use the default flags */
/* clear previously entered digits */
if (vpm_clrdigbuf(chdev) == -1) {
    /* process error */
}
/* Now play the file */
if (vpm_play(chdev,&iott,&tpt,EV_SYNC) == -1) {
    /* process error */
}
/* get digit using vpm_getdig( ) and continue processing. */
.
.
}

```

■ **Example 2: Using vpm_play() in asynchronous mode.**

```

#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>

```



```

#define MAXCHAN 24
int play_handler();
DX_IOTT prompt[MAXCHAN];
DV_TPT tpt;
DV_DIGIT dig;
main()
{
    int chdev[MAXCHAN], index, index1;
    char *chname;
    int i, voxfd;
    /* initialize all the DX_IOTT structures for each individual prompt */
    .
    .
    /* Open the vox file to play; the file descriptor will be used
    * by all channels.
    */
    if ((voxfd = vpm_fileopen("prompt.vox", O_RDONLY|O_BINARY)) == -1) {
        /* process error */
    }
    /* For each channel, open the device using vpm_open(), set up a DX_IOTT
    * structure for each channel, and issue vpm_play() in asynchronous mode. */
    for (i=0; i<MAXCHAN; i++) {
        /* Set chname to the channel name, e.g., vpmB1C1, vpmB1C2,... */
        /* Open the device using vpm_open( ). chdev[i] has channel device
        * descriptor.
        */
        if ((chdev[i] = vpm_open(chname, NULL)) == -1) {
            /* process error */
        }
        /* Use sr_enbhdr() to set up handler function to handle play
        * completion events on this channel.
        */
        if (sr_enbhdr(chdev[i], TDX_PLAY, play_handler) == -1) {
            /* process error */
        }
    }
    /*

```



```

    * Set the DV_TPT structures up for MAXDTMF. Play until one digit is
    * pressed or the file is played
    */
    vpm_clrtpt(&tpt,1);
    tpt.tp_type = IO_EOT; /* only entry in the table */
    tpt.tp_termno = DX_MAXDTMF; /* Maximum digits */
    tpt.tp_length = 1; /* terminate on the first digit */
    tpt.tp_flags = TF_MAXDTMF; /* Use the default flags */
    prompt[i].io_type = IO_DEV|IO_EOT; /* play from file */
    prompt[i].io_bufp = 0;
    prompt[i].offset = 0;
    prompt[i].io_length = -1; /* play till end of file */
    prompt[i].io_nextp = NULL;
    prompt[i].io_fhandle = voxfd;
    /* play the data */
    if (vpm_play(chdev[i],&prompt[i],&tpt,EV_ASYNC)
        == -1) {
        /* process error */

    }
}
/* Use sr_waitevt to wait for the completion of vpm_play().
   * On receiving the completion event, TDX_PLAY, control is transferred
   * to the handler function previously established using sr_enbhdr().
   */
.
.
}
int play_handler()
{
    long term;
    /* Use VPMX_TERMMSK() to get the reason for termination. */
    term = VPMX_TERMMSK(sr_getevtdev());
    if (term & TM_MAXDTMF) {
        printf("play terminated on receiving DTMF digit(s)\n");
    } else if (term & TM_EOD) {

```



```

        printf("play terminated on reaching end of data\n");
    } else {
        printf("Unknown termination reason: %x\n", term);
    }
    /* Kick off next function in the state machine model. */
    .
    .
    return 0;
}

```

■ Errors

	-1	error
	ATDV_LASTERR()	ATDV_ERRMSG()
EDX_BADPARAM	Invalid Parameter	
EDX_BADIOTT	Invalid <i>DX_IOTT</i> entry	
EDX_BADTPT	Invalid <i>DX_TPT</i> entry	
EDX_BUSY	Busy executing I/O function	
EDX_SYSTEM	Windows 2000 system error - check errno	



Related function

- **vpm_play()**
- **vpm_rec()**
- **vpm_recf()**

Setting Speed and Volume:

- **vpm_adjsv()**
- **vpm_setsvcond()**

Setting Order and Location for Voice Data:

- *DX_IOTT* (Chapter 4. Voice Data Structures and Device Parameters)

Retrieving and Handling Play Termination Events:

- Event Management functions (*Standard Runtime Library Programmer's Guide for Windows 2000* and *Appendix A* of this guide)
- **VPMX_TERMMSK()**
- *DV_TPT* (*Appendix A*)

synchronously plays voice data vpm_playf()

Name : int vpm_playf(chdev,fnamep,tptp,mode)	
Inputs : int chdev	valid SCT channel device handle
char *fnamep	play file pointer
DV_TPT *tptp	pointer to Termination Parameter Table Structure
unsigned short mode	playing mode bit mask for this play session
Returns : 0 :	
1:	
Includes: srllib.h	
smartsdk.h	
Category: Convenience	

■

vpm_playf() voice data play convenience

.

vpm_playf() DX_IOTT structure 가 single file entry

vpm_play() . file DX_IOTT structure

single file data play .

vpm_playf() 가 application file open close

 vpm_playf() fnamep file close open

.

Parameter	Description
fnamep:	voice data 가 play file pointer

argument **vpm_play()** .

■ **Source Code**

```
/******  
*                      NAME: int vpm_playf(devd,filep,tptp,mode)  
* DESCRIPTION: This function opens and plays a
```


- * named file.
- * INPUTS: devd - channel descriptor
- * tptp - pointer to the termination control block
- * filep - pointer to file name

- * OUTPUTS: Data is played.
- * RETURNS: 0 - success -1 - failure
- * CALLS: open() vpm_play() close()
- * CAUTIONS: none.

*****/

```
int vpm_playf(devd,filep,tptp,mode)
    int devd;
    char *filep;
    DV_TPT *tptp;
    USHORT mode;
{
    DX_IOTT iott;
    int rval;

    /*
    * If Async then return Error
    * Reason: IOTT' s must be in scope for the duration of the play
    */
    if ( mode & EV_ASYNC ) {
        return( -1 );
    }

    /* Open the File */
    if ((iott.io_fhandle = vpm_fileopen(filep,O_RDONLY)|O-BINARY) == -1) {
        return -1;
    }

    /* Use vpm_play() to do the Play */
    iott.io_type = IO_EOT | IO_DEV;
    iott.io_offset = (unsigned long)0;
    iott.io_length = -1;
```



```

rval = vpm_play(devd,&iott,tptp,mode);

if (vpm_fileclose(iott.io_fhandle) == -1) {
    return -1;
}

return rval;
}

```

■ Example

```

#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>

main()
{
    int chdev;
    DV_TPT tpt[2];

    /* Open the channel using vpm_open( ). Get channel device descriptor in
    * chdev.
    */
    if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
        /* process error */
    }

    /*
    * Set up the DV_TPT structures for MAXDTMF. Play until one digit is
    * pressed or the file has completed play
    */
    vpm_clrtpt(tpt,1);
    tpt[0].tp_type = IO_EOT; /* only entry in the table */
    tpt[0].tp_termno = DX_MAXDTMF; /* Maximum digits */
    tpt[0].tp_length = 1; /* terminate on the first digit */
}

```



```

tpt[0].tp_flags = TF_MAXDTMF; /* Use the default flags */

if (vpm_playf(chdev,"weather.vox",tpt,EV_SYNC) == -1) {
    /* process error */
}
.
.
}

```

■ Errors

-1
error
ATDV_LASTERR() **ATDV_ERRMSGP()** .

EDX_BADPARAM	Invalid Parameter
EDX_BADIOTT	Invalid <i>DX_IOTT</i> entry
EDX_BADTPT	Invalid <i>DX_TPT</i> entry
EDX_BUSY	Busy executing I/O function
EDX_SYSTEM	Windows 2000 system error - check errno



Related Functions:

- **vpm_rec()**
- **vpm_recf()**
- **vpm_setparm()**, **vpm_getparm()**

Setting Speed and Volume:

- **vpm_adjsv()**
- **vpm_setsvcond()**

Setting and Handling Play Termination:

- **VPMX_TERMMSK()**
- **DV_TPT** (*Appendix A*)

- | | | | | | | | | |
|----|---------|--------|------------|-----------|--------|-------|-------------|------------------|
| 1. | DX_IOTT | table | | file | | file | format | |
| | | DX_XPB | | | | | | . |
| 2. | DX_IOTT | table | | file | DX_XPB | | | type data |
| | | | | | | | | . |
| 3. | VOX | file | playing | recording | | data | format | DX_XPB structure |
| | | | mode field | | | | | . |
| 4. | WAVE | file | play | DX_XPB | | field | | . |
| 5. | WAVE | file | play | | | | data format | dl play |

■ Example

```
#include "srllib.h"
#include "smartsdk.h"

int chdev;          /* channel descriptor */
int fd;             /* file descriptor for file to be played */
DX_IOTT iott;       /* I/O transfer table */
DV_TPT tpt;         /* termination parameter table */
DX_XPB xpb;         /* I/O transfer parameter block */
.
.
.
/* Open channel */
if ((chdev = vpm_open("vpmB1C1",0)) == -1) {
    printf("Cannot open channel\n");
    printf("errno = %d\n",errno);
    exit(1);
}

/* Set to terminate play on 1 digit */
tpt.tp_type = IO_EOT;
tpt.tp_termno = DX_MAXDTMF;
tpt.tp_length = 1;
tpt.tp_flags = TF_MAXDTMF;
/* Open VOX file to play */
if ((fd = vpm_fileopen("HELLO.VOX",O_RDONLY|O_BINARY)) == -1) {
```



```

        printf("File open error\n");
        exit(2);
    }
    /* Set up DX_IOTT */
    iott.io_fhandle = fd;
    iott.io_bufp = 0;
    iott.io_offset = 0;
    iott.io_length = -1;
    iott.io_typ = IO_DEV | IO_EOT;
    /*
    * Specify VOX file format for ADPCM at 8KHz
    */
    xpb.wFileFormat = FILE_FORMAT_VOX;
    xpb.wDataFormat = DATA_FORMAT_ADPCM;
    xpb.nSamplesPerSec = DRT_8KHZ;
    xpb.nBitsPerSample = 4;

    /* Wait forever for phone to ring and go offhook */
    if (vpm_wtring(chdev,1,DX_OFFHOOK,-1) == -1) {
        printf("Error waiting for ring - %s\n", ATDV_LASTERR(chdev));
        exit(3);
    }
    /* Start playback */
    if (vpm_playiottdata(chdev,&iott,&tpt,&xpb,EV_SYNC)==-1) {
        printf("Error playing file - %s\n", ATDV_ERRMSGP(chdev));
        exit(4);
    }

    void set_xpb(DX_XPB * xpb, int msg_type)
    {
        switch(msg_type){
            case VOX24K:
                xpb->wFileFormat = FILE_FORMAT_VOX;
                xpb->wDataFormat = DATA_FORMAT_ADPCM;
                xpb->nSamplesPerSec = DRT_8KHZ;
                xpb->wBitsPerSample = 3;

```



```

        break;
    case VOX16K:
        xpb->wFileFormat = FILE_FORMAT_VOX;
        xpb->wDataFormat = DATA_FORMAT_ADPCM;
        xpb->nSamplesPerSec = DRT_8KHZ;
        xpb->wBitsPerSample = 2;
        break;
    case WAVE64K_MULAW:
        xpb->wFileFormat = FILE_FORMAT_WAVE;
        xpb->wDataFormat = DATA_FORMAT_MULAW;
//DATA_FORMAT_ALAW;
        xpb->nSamplesPerSec = DRT_8KHZ;
        xpb->wBitsPerSample = 8;
        break;
    case WAVE64K_RAW:
        xpb->wFileFormat = FILE_FORMAT_WAVE;
        xpb->wDataFormat = DATA_FORMAT_MULAW_RAW;
//DATA_FORMAT_ALAW_RAW;
        xpb->nSamplesPerSec = DRT_8KHZ;
        xpb->wBitsPerSample = 8;
        break;
    case VOX32K:
    default:
        xpb->wFileFormat = FILE_FORMAT_VOX;
        xpb->wDataFormat = DATA_FORMAT_ADPCM;
        xpb->nSamplesPerSec = DRT_8KHZ;
        xpb->wBitsPerSample = 4;
        break;
}

```

Errors

asynchronous mode		TDX_PLAY event
.	VPMX_LASTTERM()	.
	TDX_ERROR event 가	error
ATDV_LASTERR()	.	

synchronous mode

-1

ATDV_LASTERR()

..:

Equate	Returned When
EDX_BUSY	Channel is busy
EDX_XPBPARAM	Invalid <i>DX_XPB</i> setting
EDX_BADIOTT	Invalid <i>DX_IOTT</i> setting
EDX_SYSTEM	System I/O errors
EDX_BADWAVFILE	Invalid WAV file
EDX_SH_BADCMD	Unsupported command or WAV file format



vpm_playwav()

vpm_playvox()

plays tone defined by TN_GEN template

Name	:	int vpm_playtone(chdev,tngenp,tptp,mode)	
Inputs	:	int chdev	valid SCT channel device handle
		<i>TN_GEN</i> *tngenp	pointer to the <i>TN_GEN</i> structure
		DV_TPT*tptp	pointer to the DV_TPT structure
		int mode	asynchronous/synchronous
Returns	:	0 :	
		-1:	
Includes	:	srllib.h	
		smartsdk.h	
Category	:	Global Tone Generation	
Mode	:	asynchronous/synchronous	

vpm_playtone()	frequency, duration, amplitude	define
play	TN_GEN template	tone play .

■ Asynchronous Operation

mode field EV_ASYNC .
0
event .
DV_TPT structure . structure
tptp pointer 가 .
TDX_PLAYTONE event .
event SRL Event Management . Event
Management Appendix A .
vpm_playtone()
VPMX_TERMMSK() .

■ **synchronous Operation**

0 .

DV_TPT structure . structure


```

#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>

#define TID_1 101

main()
{
    TN_GEN tngen;
    DV_TPT tpt[ 5 ];
    int vpmdev;

    /*
    * Open the Voice Channel Device and Enable a Handler
    */
    if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
        perror( "vpmB1C1" );
        exit( 1 );
    }

    /*
    * Describe a Simple Dual Tone Frequency Tone of 950-
    * 1050 Hz and 475-525 Hz using leading edge detection.
    */
    if ( vpm_blddt( TID_1, 1000, 50, 500, 25, TN_LEADING ) == -1 ) {
        printf( "Unable to build a Dual Tone Template\n" );
    }

    /*
    * Bind the Tone to the Channel
    */
    if ( vpm_addtone( vpmdev, NULL, 0 ) == -1 ) {
        printf( "Unable to Bind the Tone %d\n", TID_1 );
        printf( "LastError = %d Err Msg = %s\n",
            ATDV_LASTERR( vpmdev ), ATDV_ERRMSGP( vpmdev ) );
        vpm_close( vpmdev );
    }
}

```



```

    exit( 1 );
}

/*
 * Enable Detection of ToneId TID_1
 */
if ( vpm_enbtone( vpmdev, TID_1, DM_TONEON | DM_TONEOFF ) == -1 ) {
    printf( "Unable to Enable Detection of Tone %d\n", TID_1 );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSG( vpmdev ) );
    vpm_close( vpmdev );
    exit( 1 );
}

/*
 * Build a Tone Generation Template.
 * This template has Frequency1 = 1140,
 * Frequency2 = 1020, amplitude at -10dB for
 * both frequencies and duration of 100 * 10 msecs.
 */
vpm_bldtngen( &tngen, 1140, 1020, -10, -10, 100 );

/*
 * Set up the Terminating Conditions
 */
tpt[0].tp_type = IO_CONT;
tpt[0].tp_termno = DX_TONE;
tpt[0].tp_length = TID_1;
tpt[0].tp_flags = TF_TONE;
tpt[0].tp_data = DX_TONEON;

tpt[1].tp_type = IO_CONT;
tpt[1].tp_termno = DX_TONE;
tpt[1].tp_length = TID_1;
tpt[1].tp_flags = TF_TONE;
tpt[1].tp_data = DX_TONEOFF;

```



```

tpt[2].tp_type = IO_EOT;
tpt[2].tp_termno = DX_MAXTIME;
tpt[2].tp_length = 6000;
tpt[2].tp_flags = TF_MAXTIME;

if (vpm_playtone( vpmdev, &tngen, tpt, EV_SYNC ) == -1 ){
    printf( "Unable to Play the Tone\n" );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSGP( vpmdev ) );
    vpm_close( vpmdev );
    exit( 1 );
}

/*
 * Continue Processing
 * .
 * .
 * .
 */

/*
 * Close the opened Voice Channel Device
 */
if ( vpm_close( vpmdev ) != 0 ) {
    perror( "close" );
}

/* Terminate the Program */
exit( 0 );
}

```

Errors

	-1		error
ATDV_LASTERR()	ATDV_ERRMSGP()	.	

EDX_BADPARAM	Invalid parameter
EDX_BADPROD	Function not supported on this board
EDX_BADTPT	Invalid DV_TPT entry
EDX_BUSY	Busy executing I/O function
EDX_AMPLGEN	Invalid amplitude value in <i>TN_GEN</i> structure
EDX_FREQGEN	Invalid frequency component in <i>TN_GEN</i> structure
EDX_FLAGGEN	Invalid <i>tn_dflag</i> field in <i>TN_GEN</i> structure
EDX_SYSTEM	Windows 2000 system error - check errno



Related to Tone Generation:

- **vpm_bldtngen()**
- *TN_GEN* (Chapter 4. Voice Data Structures and Device Parameters)
- "Global Tone Generation" (*Voice Features Guide for Windows 2000*)

Handling and Retrieving **vpm_playtone()** Termination Events:

- Event Management functions (*Standard Runtime Library Programmer's*

Guide for Windows 2000 and Appendix A)

- DV_TPT (*Appendix A*)
- **VPMX_TERMMSK()**

plays voice data stored in a single VOX file

vpm_playvox()

Name: SHORT vpm_playvox(chdev, filenamep, tptp, xpbp, mode)

Inputs: int chdev valid SCT channel device handle

 char *filenamep play file pointer

 DV_TPT *tptp termination parameter block pointer

 DX_XPB *xpbp I/O transfer parameter block pointer

 unsigned short mode play mode

Returns: 0 :
 -1 :

Includes: smartsdk.h

Category: Convenience function

Mode: synchronous



vpm_playvox() convenience function single VOX file voice data

 play . **xpbp** 가 NULL data ADPCM 8KHz

 .

Parameter	Description
chdev	Channel device descriptor
tcbp	termination parameter table pointer
filenamep	play file pointer
xpbp	I/O transfer parameter block Pointer (See the <i>DX_XPB</i> data structure)
mode	specifies the play mode:



Example

```
#include "srllib.h"
#include "smartsdk.h"
```



```

int chdev;          /* channel descriptor */
DV_TPT tpt;        /* termination parameter table */.
.
.
/* Open channel */
if ((chdev = vpm_open("vpmB1C1",0)) == -1) {
    printf("Cannot open channel\n");
    printf("errno = %d\n",errno);
    exit(1);
}
/* Set to terminate play on 1 digit */
tpt.tp_type = IO_EOT;
tpt.tp_termno = DX_MAXDTMF;
tpt.tp_length = 1;
tpt.tp_flags = TF_MAXDTMF;
/* Wait forever for phone to ring and go offhook */
if (vpm_wtring(chdev,1,DX_OFFHOOK,-1) == -1) {
    printf("Error waiting for ring - %s\n", ATDV_LASTERR(chdev));
    exit(3);
}
/* Start 8KHz ADPCM playback */
if (vpm_playvox(chdev,&tpt,"HELLO.VOX",NULL,0) == -1) {
    printf("Error playing file - %s\n", ATDV_ERRMSGP(chdev));
    exit(4);
}

```

Errors

	-1
ATDV_LASTERR()	가. ...

Equate	Returned When
EDX_BUSY	Channel is busy
EDX_XBPARM	Invalid <i>DX_XPB</i> setting
EDX_BADIOTT	Invalid <i>DX_IOTT</i> setting
EDX_SYSTEM	System I/O errors

EDX_BADWAVFILE	Invalid WAV file
EDX_SH_BADCMD	Unsupported command or WAV file format



- **vpm_playiottdata()**
- **vpm_playwav()**

vpm_playwav() plays voice data stored in a single WAVE file

Name:	SHORT vpm_playwav(chdev, filenamep, tptp, mode)		
Inputs:	int chdev	valid SCT channel device	
		handle	
	char *filenamep	play file	pointer
	DV_TPT*tptp	termination parameter block	pointer
	unsigned short mode	play mode	
Returns:	0 :		
	-1 :		
Includes:	srllib.h		
	smartsdk.h		
Category:	Convenience function		
Mode:	synchronous		



vpm_playwav() convenience function single WAVE file voice data
play **vpm_playiottdata()** .
WAVE file format *DX_XPB* structure

Parameter	Description			
chdev	Channel device descriptor			
tcbp	termination parameter table	pointer		
filenamep	play file	pointer		
mode	specifies the play mode:			
	PM_TONE	200ms	tone	play
	EV_SYNC	synchronous operation		
	(must be specified)			
	NOTE: PM_TONE	EV_SYNC OR bitmask	..	




```

                                data waveform      play                                error
                                data waveform      .
8 and mu-law 8-bit PCM (WAVE_FORMAT_MULAW)
8 and a-law 8-bit PCM (WAVE_FORMAT_ALAW)

```

■ Example

```

#include "srllib.h"
#include "smartsdk.h"

int chdev;                                /* channel descriptor */
DV_TPT tpt;                              /* termination parameter table */
.
.
.

/* Open channel */
if ((chdev = vpm_open("vpmB1C1",0)) == -1) {
    printf("Cannot open channel\n");
    printf("errno = %d\n",errno);
    exit(1);
}

/* Set to terminate play on 1 digit */
tpt.tp_type = IO_EOT;
tpt.tp_termno = DX_MAXDTMF;
tpt.tp_length = 1;
tpt.tp_flags = TF_MAXDTMF;
/* Wait forever for phone to ring and go offhook */
if (vpm_wtring(chdev,1,DX_OFFHOOK,-1) == -1) {
    printf("Error waiting for ring - %s\n", ATDV_LASTERR(chdev));
    exit(3);
}

/* Start playback */
if (vpm_playwav(chdev,&tpt,"HELLO.WAV",EV_SYNC) == -1) {
    printf("Error playing file - %s\n", ATDV_ERRMSGP(chdev));
    exit(4);
}

```


■ Errors

가 -1

ATDV_LASTERR()

.

Equate

Returned When

EDX_BUSY	Channel is busy
EDX_XPBPARM	Invalid <i>DX_XPB</i> setting
EDX_BADIOTT	Invalid <i>DX_IOTT</i> setting
EDX_SYSTEM	System I/O errors
EDX_BADWAVFILE	Invalid WAV file
EDX_SH_BADCMD	Unsupported command or WAV file format



- **vpm_playiottdata()**
- **vpm_playvox()**

records voice data from a single channel vpm_rec()

Name:	int vpm_rec(chdev,iottp,tptp,mode)		
Inputs:	int chdev	valid SCT channel device	
		handle	
	<i>DX_IOTT</i> *iottp	I/O Descriptor Table	pointer
	DV_TPT *tptp	Termination Parameter Table	
		Structure	pointer
	unsigned short mode	asynchronous/synchronous	
		setting and recording mode bit	
		mask for this record session	
Returns:	0 :		
	-1:		
Includes:	srllib.h		
	smartsdk.h		
Category:	I/O		
Mode:	synchronous/asynchronous		



vpm_rec() voice data record . data data
file, custom device record . voice data 가 record
DX_IOTTstructure .
가 , DX_IOTT structure 가
.
vpm_rec() 가 vpm_stopch()가 DX_IOTT structure
DV_TPT 가
recording . vpm_rec() 가
Extended Attribute function

NOTE: single file record vpm_recf() DX_IOTT structure
.
vpm_recf()

■ Asynchronous Operation

EV_ASYNC MODE field 0

event

DV_TPT structure structure

tptp pointer

recording TDX_RECORD event

event SRL Event Management Event

Management Appendix A

vpm_rec() 가 VPMX_TERMMSK()

■ synchronous Operation

0

DV_TPT structure structure

tptp pointer

vpm_rec() 가 VPMX_TERMMSK()

Parameter	Description
chdev:	vpm_open() poen valid channel device handle
iottp:	points to the I/O Transfer Table Structure, <i>DX_IOTT</i> , which specifies the order in which and media onto which the voice data will be recorded.
tptp:	recording DV_TPT Termination Parameter Table Structure pointer
	DX_MAXDTMF : Maximum number of digits received DX_MAXSIL : Maximum silence DX_MAXNOSIL : Maximum non-silence DX_IDDTIME : Inter-digit delay DX_MAXTIME : Function Time

DX_DIGMASK :Digit mask termination
 DX_TONE : Tone-off or Tone-on detection

structure SRL *Appendix*
 A .

NOTE: DV_TPT termination byte count,
 vpm_stopch(), file .
 VPMX_TERMMSK() .

mode: recording mode .
 bit mask .

Choose one only
 EV_ASYNC : Run **vpm_rec()** asynchronously.
 EV_SYNC : Run **vpm_rec()** synchronously (default).

Choose one or more
 RM_TONE : tone . mode 가
 t one play .(default)
 RM_AGC : auto gain control .



■ Example 1: Using vpm_rec() in synchronous mode

```
#include <fcntl.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
#define MAXLEN 10000
main()
{
    DV_TPT tpt;
    DX_IOTT iott[2];
    int chdev;
```



```

char basebufp[MAXLEN];
/*
* open the channel using vpm_open( )
*/
if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
    /* process error */
}
/*
* Set up the DV_TPT structures for MAXDTMF
*/
vpm_clrtpt(&tpt,1);
tpt.tp_type = IO_EOT;           /* last entry in the table */
tpt.tp_termno = DX_MAXDTMF;     /* Maximum digits */
tpt.tp_length = 1;              /* terminate on the first digit */
tpt.tp_flags = TF_MAXDTMF;      /* Use the default flags */
/*
* Set up the DX_IOTT. The application records the voice data to memory
* allocated by the user.
*/
iott[0].io_type = IO_MEM|IO_CONT; /* Record to memory */
iott[0].io_bufp = basebufp;       /* Set up pointer to buffer */
iott[0].io_offset = 0;             /* Start at beginning of buffer */
iott[0].io_length = MAXLEN;        /* Record 10,000 bytes of voice data */
iott[1].io_type = IO_DEV|IO_EOT;   /* Record to file, last DX_IOTT
                                     /* entry */
iott[1].io_bufp = 0;              /* Set up pointer to buffer */
iott[1].io_offset = 0;            /* Start at beginning of buffer */
iott[1].io_length = MAXLEN;        /* Record 10,000 bytes of voice
                                     /* data */

if((iott[1].io_fhandle = vpm_fileopen("file.vox",
    O_RDWR|O_CREAT|O_TRUNC|O_BINARY,0666)) == -1) {
    /* process error */
}
/* clear previously entered digits */
if (vpm_clrdigbuf(chdev) == -1) {
    /* process error */
}

```



```

    }
    if (vpm_rec(chdev,&iott[0],&tpt,RM_TONE|EV_SYNC) == -1) {
        /* process error */
    }
    /* Analyze the data recorded */
    .
    .
}

```

■ Example 2: Using vpm_rec() in asynchronous mode

```

#include <stdio.h>
#include <fcntl.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
#define MAXLEN 10000
#define MAXCHAN 24
int record_handler();
DV_TPT tpt;
DX_IOTT iott[MAXCHAN];
int chdev[MAXCHAN];
char basebufp[MAXCHAN][MAXLEN];
main()
{
    int i;
    char *chname;

    /* Start asynchronous vpm_rec() on all the channels. */
    for (i=0; i<MAXCHAN; i++) {
        /* Set chname to the channel name, e.g., vpmB1C1, vpmB1C2,... */
        /*
        * open the channel using vpm_open( )
        */
        if ((chdev[i] = vpm_open(chname,NULL)) == -1) {
            /* process error */

```



```

}
/* Using sr_enbhdr(), set up handler function to handle record
 * completion events on this channel.
 */
if (sr_enbhdr(chdev[i], TDX_RECORD, record_handler) == -1) {
    /* process error */
}
/*
 * Set up the DV_TPT structures for MAXDTMF
 */
vpm_clrtpt(&tpt,1);
tpt.tp_type = IO_EOT;                /* last entry in the table */
tpt.tp_termno = DX_MAXDTMF;          /* Maximum digits */
tpt.tp_length = 1;                   /* terminate on the first digit */
tpt.tp_flags = TF_MAXDTMF;           /* Use the default flags */
/*
 * Set up the DX_IOTT. The application records the voice data to memory
 * allocated by the user.
 */
iott[i].io_type = IO_MEM|IO_EOT;      /* Record to memory, last DX_IOTT
                                         /* entry */
iott[i].io_bufp = basebufp[i];        /* Set up pointer to buffer */
iott[i].io_offset = 0;                 /* Start at beginning of buffer */
iott[i].io_length = MAXLEN;           /* Record 10,000 bytes voice data */
/* clear previously entered digits */
if (vpm_clrdigbuf(chdev) == -1) {
    /* process error */
}
/* Start asynchronous vpm_rec() on the channel */
if (vpm_rec(chdev[i],&iott[i],&tpt, RM_TONE|EV_ASYNC) == -1) {
    /* process error */
}
}
}
/* Use sr_waitevt to wait for the completion of vpm_rec().
 * On receiving the completion event, TDX_RECORD, control is transferred
 * to a handler function previously established using sr_enbhdr().

```



```

    */
    .
    .
}
int record_handler()
{
    long term;
    /* Use VPMX_TERMMSK() to get the reason for termination. */
    term = VPMX_TERMMSK(sr_getevtdev());
    if (term & TM_MAXDTMF) {
        printf("record terminated on receiving DTMF digit(s)\n");
    } else if (term & TM_NORMTERM) {
        printf("normal termination of vpm_rec()\n");
    } else {
        printf("Unknown termination reason: %x\n", term);
    }
    /* Kick off next function in the state machine model. */
    .
    .
    return 0;
}

```

■ Errors

가	-1	error
ATDV_LASTERR()	ATDV_ERRMSGP()	.
EDX_BADDEV	Invalid Device Descriptor	
EDX_BADPARAM	Invalid Parameter	
EDX_BADIOTT	Invalid <i>DX_IOTT</i> entry	
EDX_BADTPT	Invalid <i>DX_TPT</i> entry	
EDX_BUSY	Busy executing I/O function	
EDX_SYSTEM	Windows 2000 system error - check errno	



Related Functions:

- vpm_recf()

- vpm_play()
- vpm_playf()
- vpm_setparm(), vpm_getparm()

Setting Order and Location for Voice Data:

- *DX_IOTT (Chapter 4. Voice Data Structures and Device Parameters)*

Retrieving and Handling Record Termination Events:

- Event Management functions (*Standard Runtime Library Programmer's Guide for Windows 2000 and Appendix A of this guide*)
- VPMX_TERMMSK()
- DV_TPT (*Appendix A*)

vpm_recf()

Name	:	int vpm_recf(chdev,fnamep,tptp,mode)	
Inputs	:	int chdev	valid SCT channel device handle
		char *fnamep	pointer to file to record to
		DV_TPT *tptp	Termination Parameter Table Structure pointer
		unsigned short mode	recording mode bit mask for this record session
Returns	:	0 :	
		-1 :	
Includes	:	srllib.h	
		smartsdk.h	
Category	:	Convenience	
Mode	:	synchronous/Asynchronous	

```

vpm_recf()      voice data 가 channel      single file      record
      . vpm_recf()      single file      DX_IOTT structure      가
vpm_recf()      .      file      DX_IOTT structure
      data      recording      . vpm_recf()      가
application      file      open      close      vpm_recf()
      file      close      open      .

```

Parameter	Description
fname:	voice data 가 record file pointer

arguments **vpm_rec()**

■ Source Code

```

/*****
*      NAME: int vpm_recf(devd,filep,tptp,mode)
* DESCRIPTION: Record data to a file

```



```

*      INPUTS: devd - channel descriptor
*
*              ttp - TPT pointer
*
*              filep - ASCIIZ string for name of file to read into
*
*              mode - tone initiation flag
*
*      OUTPUTS: Data stored in file, status in CSB pointed to by csbp
*
*      RETURNS: 0 or -1 on error
*
*      CALLS: open() vpm_rec() close()
*
*      CAUTIONS: none.

```

```

*****

```

```

*/

```

```

int vpm_recf(devd,filep,ttp,mode)

```

```

    int devd;

```

```

    char *filep;

```

```

    DV_TPT *ttp;

```

```

    USHORT mode;

```

```

{

```

```

    int rval;

```

```

    DX_IOTT iott;

```

```

    /*

```

```

    * If Async then return Error

```

```

    * Reason: IOTT' s must be in scope for the duration of the record

```

```

    */

```

```

    if ( mode & EV_ASYNC ) {

```

```

        return( -1 );

```

```

    }

```

```

    /* Open the File */

```

```

    if ((iott.io_fhandle =

```

```

vpm_fileopen(filep,(O_WRONLY|O_CREAT|O_TRUNC),0666))==

```

```

        1) {

```

```

            return -1;

```

```

        }

```

```

    /* Use vpm_rec() to do the record */

```

```

    iott.io_type = IO_EOT | IO_DEV;

```

```

    iott.io_offset = (long)0;

```



```

iott.io_length = -1;

rval = vpm_rec(devd,&iott,tptp,mode);

if (vpm_fileclose(iott.io_fhandle) == -1) {
    return -1;
}
return rval;
}

```

■ Example

```

#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>

main()
{
    int chdev;
    long termtype;
    DV_TPT tpt[2];

    /* Open the channel using vpm_open( ). Get channel device descriptor in
    * chdev
    */
    if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
        /* process error */
    }

    /* Set the DV_TPT structures up for MAXDTMF and MAXSIL */
    vpm_clrtpt(tpt,2);
    tpt[0].tp_type = IO_CONT;
    tpt[0].tp_termno = DX_MAXDTMF; /* Maximum digits */
    tpt[0].tp_length = 1; /* terminate on the first digit */
    tpt[0].tp_flags = TF_MAXDTMF; /* Use the default flags */
}

```



```

/*
 * If the initial silence period before the first non-silence period
 * exceeds 4 seconds then terminate. If a silence period after the
 * first non-silence period exceeds 2 seconds then terminate.
 */
tpt[1].tp_type = IO_EOT;           /* last entry in the table */
tpt[1].tp_termno = DX_MAXSIL;     /* Maximum silence */
tpt[1].tp_length = 20;            /* terminate on 2 seconds of
                                   * continuous silence */
tpt[1].tp_flags = TF_MAXSIL|TF_SETINIT; /* Use the default flags and
                                   * initial silence flag */
tpt[1].tp_data = 40;             /* Allow 4 seconds of initial
                                   * silence */
if (vpm_recf(chdev,"weather.vox",tpt,RM_TONE) == -1) {
    /* process error */
}
termtype = VPMX_TERMMSK(chdev);   /* investigate termination reason */
if (termtype & TM_MAXDTMF) {
    /* process DTMF termination */
}
...
}

```

■ Errors

가	-1	error
	ATDV_LASTERR()	ATDV_ERRMSGP()
EDX_BADPARAM	Invalid Parameter	
EDX_BADIOTT	Invalid <i>DX_IOTT</i> entry	
EDX_BADTPT	Invalid <i>DX_TPT</i> entry	
EDX_BUSY	Busy executing I/O function	
EDX_SYSTEM	Windows 2000 system error - check errno	

■ See Also

Related Functions:

- `vpm_rec()`
- `vpm_play()`
- `vpm_playf()`
- `vpm_setparm(), vpm_getparm()`

Setting and Handling Record Termination:

- `VPMX_TERMMSK()`
- `DV_TPT` (*Appendix A*)

records voice data to multiple destinations, vpm_reciottdata()

Name:	short vpm_reciottdata(chdev, iottp, tptp, xpbp, mode)		
Inputs:	int chdev	valid SCT channel device	
		handle	
	<i>DX_IOTT</i> *iottp	I/O Transfer Table	pointer
	DV_TPT *tptp	Termination Parameter Table	
		Stucture	pointer
	<i>DX_XPB</i> *xpbp	I/O Transfer Parameter block	pointer
	unsigned short mode	play mode	
Returns:	0 :		
	-1:		
Includes:	srllib.h		
	smartsdk.h		
Category:	I/O function		
Mode:	synchronous or asynchronous		



vpm_reciottdata() data files, memory, custom devices 가
voice data record .

Parameter	Description
chdev	channel device descriptor.
iottp	Pointer to <i>DX_IOTT</i> table that specifies the order and media onto which the voice data will be recorded.
tptp	Termination Parameter Table structure pointer
xpbp	I/O transfer parameter block pointer
mode	specifies the record mode: EV_SYNCH synchronous mode EV_ASYNC asynchronous mode



1. *DX_IOTT* table *DX_XPB* file format
 2. record file *DX_XPB* data encoding rate
- 가 ..

■ Example

```
#include "srllib.h"
```

```
#include "smartsdk.h"
```

```
int chdev;      /* channel descriptor */
int fd;         /* file descriptor for file to be played */
DX_IOTT iott;   /* I/O transfer table */
DV_TPT tpt;     /* termination parameter table */
DX_XPB xpb;     /* I/O transfer parameter block */

.
.

/* Open channel */
if ((chdev = vpm_open("vpmB1C1",0)) == -1) {
    printf("Cannot open channel\n");
    printf("errno = %d\n",errno);
    exit(1);
}

/* Set to terminate play on 1 digit */
tpt.tp_type = IO_EOT;
tpt.tp_termno = DX_MAXDTMF;
tpt.tp_length = 1;
tpt.tp_flags = TF_MAXDTMF;

/* Open file */
if ((fd = vpm_fileopen("MESSAGE.VOX",O_RDWR|O_BINARY)) == -1) {
    printf("File open error\n");
    exit(2);
}

/* Set up DX_IOTT */
iott.io_fhandle = fd;
iott.io_bufp = 0;
```



```

iott.io_offset = 0;
iott.io_length = -1;
iott.io_typ = IO_DEV | IO_EOT;

/*
 * Specify VOX file format for PCM at 8KHz.
 */
xpb.wFileFormat = FILE_FORMAT_VOX;
xpb.wDataFormat = DATA_FORMAT_ADPCM;
xpb.nSamplesPerSec = DRT_8KHZ;
xpb.nBitsPerSample = 8;

/* Wait forever for phone to ring and go offhook */
if (vpm_wtrng(chdev,1,DX_OFFHOOK,-1) == -1) {
    printf("Error waiting for ring - %s\n", ATDV_LASTERR(chdev));
    exit(3);
}
/* Play intro message */
if (vpm_playwav(chdev,&tpt,"HELLO.WAV",EV_SYNC) == -1) {
    printf("Error playing file - %s\n", ATDV_ERRMSGP(chdev));
    exit(4);
}
/* Start recording */
if (vpm_reciottdata(chdev,&iott,&tpt,&xpb,PM_TONE|EV_SYNC) == -1) {
    printf("Error recording file - %s\n", ATDV_ERRMSGP(chdev));
    exit(4);
}
void set_xpb(DX_XPB * xpb, int msg_type)
{
    switch(msg_type){
        case VOX24K:
            xpb->wFileFormat = FILE_FORMAT_VOX;
            xpb->wDataFormat = DATA_FORMAT_ADPCM;
            xpb->nSamplesPerSec = DRT_8KHZ;
            xpb->wBitsPerSample = 3;
            break;
    }
}

```



```

case VOX16K:
    xpb->wFileFormat = FILE_FORMAT_VOX;
    xpb->wDataFormat = DATA_FORMAT_ADPCM;
    xpb->nSamplesPerSec = DRT_8KHZ;
    xpb->wBitsPerSample = 2;
    break;
case WAVE64K_MULAW:
    xpb->wFileFormat = FILE_FORMAT_WAVE;
    xpb->wDataFormat = DATA_FORMAT_MULAW;
//DATA_FORMAT_ALAW;
    xpb->nSamplesPerSec = DRT_8KHZ;
    xpb->wBitsPerSample = 8;
    break;
case WAVE64K_RAW:
    xpb->wFileFormat = FILE_FORMAT_WAVE;
    xpb->wDataFormat = DATA_FORMAT_MULAW_RAW;
//DATA_FORMAT_ALAW_RAW;
    xpb->nSamplesPerSec = DRT_8KHZ;
    xpb->wBitsPerSample = 8;
    break;
case VOX32K:
default:
    xpb->wFileFormat = FILE_FORMAT_VOX;
    xpb->wDataFormat = DATA_FORMAT_ADPCM;
    xpb->nSamplesPerSec = DRT_8KHZ;
    xpb->wBitsPerSample = 4;
    break;
}

```

ERROR

```

asynchronous mode . TDX_RECORD event
. VPMX_TERMMSK( )
가 TDX_ERROR event 가 .error
ATDV_LASTERR( )
synchronous mode 1 ATDV_LASTERR( )
error .

```


Equate	Returned When
EDX_BUSY	Channel is busy
EDX_XPBPARM	Invalid <i>DX_XPB</i> setting
EDX_BADIOTT	Invalid <i>DX_IOTT</i> setting
EDX_SYSTEM	System I/O errors
EDX_BADWAVFILE	Invalid WAV file
EDX_SH_BADCMD	Unsupported command or WAV file format



- **vpm_recwav()**
- **vpm_recvox()**

records voice data to a single VOX file

vpm_recvox() convenience function single VOX file voice data record
 . xbp 가 NULL data 8KHz I ADPCM

chdev	Channel device descriptor
tcbp	Termination Parameter Table pointer
filenamep	record file pointer
xpbp	I/O Transfer Parameter Block pointer
	(See the <i>DX_XPB</i> data structure)
mode	specifies the play mode:
	RM_TONE : record 200ms
	tone play
	EV_SYNC synchronous operation

(must be specified)

NOTE: RM_TONE EV_SYNC OR bitmask .



VOX file playing recording data format vpm_recvox() DX_XPB
mode parameter .

■ Example

```
#include "srllib.h"
```

```
#include "smartsdk.h"
```

```
int chdev;    /* channel descriptor */  
DV_TPT tpt;    /* termination parameter table */  
DX_XPB xpb;    /* I/O transfer parameter block */
```

```
.
```

```
.
```

```
.
```

```
/* Open channel */
```

```
if ((chdev = vpm_open("vpmB1C1",0)) == -1) {
```

```
    printf("Cannot open channel\n");
```

```
    printf("errno = %d\n",errno);
```

```
    exit(1);
```

```
}
```

```
/* Set to terminate play on 1 digit */
```

```
tpt.tp_type = IO_EOT;
```

```
tpt.tp_termno = DX_MAXDTMF;
```

```
tpt.tp_length = 1;
```

```
tpt.tp_flags = TF_MAXDTMF;
```

```
/* Wait forever for phone to ring and go offhook */
```

```
if (vpm_wtrng(chdev,1,DX_OFFHOOK,-1) == -1) {
```

```
    printf("Error waiting for ring - %s\n", ATDV_LASTERR(chdev));
```

```
    exit(3);
```

```
}
```

```
/* Start playback */
```

```
if (vpm_playwav(chdev,&tpt,"HELLO.WAV",EV_SYNC) == -1) {
```

```
    printf("Error playing file - %s\n", ATDV_ERRMSGP(chdev));
```



```

        exit(4);
    }
    /* clear digit buffer */
    vpm_clrdigbuf(chdev);
    /* Start 6KHz ADPCM recording */
    if (vpm_recvox(chdev,"MESSAGE.VOX", &tpt, NULL,PM_TONE|EV_SYNC) == -1) {
        printf("Error recording file - %s\n", ATDV_ERRMSGP(chdev));
        exit(4);
    }
}

```

Errors

-1
 ATDV_LASTERR() .

Equate	Returned When
EDX_BUSY	Channel is busy
EDX_XPBPARAM	Invalid <i>DX_XPB</i> setting
EDX_BADIOTT	Invalid <i>DX_IOTT</i> setting
EDX_SYSTEM	System I/O errors
EDX_SH_BADCMD	Unsupported command or VOX file format



- vpm_reciottdata()
- vpm_recwav()

records voice data to a single WAVE file vpm_recwav()

Name:	SHORT vpm_recwav(chdev, filenamep, tptp, xpbp, mode)		
Inputs:	int chdev	valid SCT channel device	
		handle	
	char *filenamep	record file	
		pointer	
	DV_TPT*tptp	Termination Parameter Table	
		pointer	
	DX_XPB *xpbp	I/O Transfer Parameter Block	
		pointer	
	unsigned short mode	play mode	
Returns:	0 :		
	-1:		
Includes:	srllib.h		
	smartsdk.h		
Category:	Convenience function		
Mode:	synchronous		

■ **Description**

vpm_recwav() convenience function voice data single WAVE file record
 . xpbp 가 NULL 8KHz linear 8-bit Mulaw PCM
 record . vpm_reciottdata() .

Parameter	Description
-----------	-------------

chdev	channel device descriptor
tcbp	termination parameter table pointer
filenamep	play file pointer
xpbp	I/O Transfer Parameter Block pointer
mode	specifies the play mode:
	RM_TONE play 200 ms audible tone
	EV_SYNC synchronous operation
	(must be specified)

NOTE: RM_TONE EV_SYNC OR bitmask



None.

■ Example

```
#include "srllib.h"
#include "smartsdk.h"

int chdev; /* channel descriptor */
DV_TPT tpt; /* termination parameter table */
DX_XPB xpb; /* I/O transfer parameter block */

.
.
.

/* Open channel */
if ((chdev = vpm_open("vpmB1C1",0)) == -1) {
    printf("Cannot open channel\n");
    printf("errno = %d\n",errno);
    exit(1);
}

/* Set to terminate play on 1 digit */
tpt.tp_type = IO_EOT;
tpt.tp_termno = DX_MAXDTMF;
tpt.tp_length = 1;
tpt.tp_flags = TF_MAXDTMF;

/* Wait forever for phone to ring and go offhook */
if (vpm_wtring(chdev,1,DX_OFFHOOK,-1) == -1) {
    printf("Error waiting for ring - %s\n", ATDV_LASTERR(chdev));
    exit(3);
}

/* Start playback */
if (vpm_playwav(chdev,&tpt,"HELLO.WAV",EV_SYNC) == -1) {
    printf("Error playing file - %s\n", ATDV_ERRMSG(chdev));
    exit(4);
}

/* clear digit buffer */
vpm_clrdigbuf(chdev);
```



```

/* Start 11KHz PCM recording */
if (vpm_recwav(chdev,"MESSAGE.WAV", &tpt, (DX_XPB
                *)NULL,RM_TONE|EV_SYNC) == -1) {
    printf("Error recording file - %s\n", ATDV_ERRMSGP(chdev));
    exit(4);
}

```

■ Errors

−1

ATDV_LASTERR()

Equate	Returned When
EDX_BUSY	Channel is busy
EDX_XPBPARAM	Invalid <i>DX_XPB</i> setting
EDX_BADIOTT	Invalid <i>DX_IOTT</i> setting
EDX_SYSTEM	System I/O errors
EDX_BADWAVFILE	Invalid WAV file
EDX_SH_BADCMD	Unsupported command or WAV file format



- vpm_reciottdata()
- vpm_recvox()

enables detection of Call Status Transition (CST) event

Name: int vpm_setevtmsk(chdev,mask)

Returns: unsigned int mask event mask of events to wait for

0 :
-1:

Category: Call Status Transition Event

■ Description

NOTE:	function	tone	CST event
	가	.	vpm_addtone()
	vpm_enbtone()	.	

function parameter

Parameter	Description
-----------	-------------

chdev:	channel	vpm_open()	open
	valid channel device handle		.
mask:	event(s)	vpm_getevt()	mask 가
	가		.
	DM_LCOFF		wait for loop current to be off
	DM_LCON		wait for loop current to be on
	DM_RINGS		wait for rings
	DM_RNGOFF		wait for ring to drop (hang-up)
	multiple event	polling	event
	bit mask	OR operate	.
	CST event 가		.

asynchronous mode ring off event
vpm_setevtmsk(DM_RNGOFF) , event
 vpm_getevt() . Event
 multitasking function **vpm_stopch()** .

ring
VPMX_LINEST() .
vpm_stopch() .

table synchronous asynchronous CST event

Synchronous	Asynchronous
1. Call vpm_setevtmsk() to enable CST event(s)	Call vpm_setevtmsk() to enable CSTevent(s)
2. Call vpm_getevt() to wait for CST event(s). Events are returned to the <i>DX_EBLK</i> structure.	Use SRL to asynchronously wait for for TDX_CST event(s).
3.	Use sr_getevtdatap() to retrieve <i>DX_CST</i> structure.



event **vpm_getevt()** function call
 event **vpm_setevtmsk()** **vpm_wtring()**
 . event mask 가 **vpm_wtring()**

■ Example 1: Using vpm_setevtmsk() to wait for ring events -synchronous processing

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
```



```

main()
{
    int chdev;
    DX_EBLK eblk;

    .
    .
    /* open a channel with chdev as descriptor */
    if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
        /* process error */
    }

    .
    .
    /* Set event mask to receive ring events */
    if (vpm_setevtsk(chdev, DM_RINGS) == -1) {
        /* error setting event */
    }

    .
    .
    /* check for ring event, timeout set to 20 seconds */
    if (vpm_getevt(chdev,&eblk,20) == -1) {
        /* error timeout */
    }
    if(eblk.ev_event==DE_RINGS) {
        printf("Ring event occurred\n");
    }

    .
    .
}

```

■ **Example 2: Using vpm_setevtsk() to handle call status transition events - asynchronous processing**

```

#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>

```



```
#define MAXCHAN 24
```

```
int cst_handler();
```

```
main()
```

```
{
```

```
    int chdev[MAXCHAN];
```

```
    char *chname;
```

```
    int i, srlmode;
```

```
    for (i=0; i<MAXCHAN; i++) {
```

```
        /* Set chname to the channel name, e.g., vpmB1C1, vpmB1C2,... */
```

```
        /* Open the device using vpm_open( ). chdev[i] has channel device  
        * descriptor.
```

```
        */
```

```
        if ((chdev[i] = vpm_open(chname,NULL)) == -1) {
```

```
            /* process error */
```

```
        }
```

```
        /* Use vpm_setevtsk() to enable call status transition events  
        * on this channel.
```

```
        */
```

```
        if (vpm_setevtsk(chdev[i],
```

```
            DM_LCOFF|DM_LCON|DM_RINGS|DM_SILOFF|DM_SILON) == -1) {
```

```
            /* process error */
```

```
        }
```

```
        /* Using sr_enbhdr(), set up handler function to handle call status  
        * transition events on this channel.
```

```
        */
```

```
        if (sr_enbhdr(chdev[i], TDX_CST, cst_handler) == -1) {
```

```
            /* process error */
```

```
        }
```

```
        /* Use sr_waitvt to wait for call status transition event.
```



```

    * On receiving the transition event, TDX_CST, control is transferred
    * to the handler function previously established using sr_enbhdr().
    */
        .
        .
    }
}

int cst_handler()
{
    DX_CST *cstp;

    /* sr_getevtdatap() points to the event that caused the call status
    * transition.
    */
    cstp = (DX_CST *)sr_getevtdatap();

    switch (cstp->cst_event) {
        case DE_RINGS:
            printf("Ring event occurred on channel %s\n",
                VPMX_NAMEP(sr_getevtdev()));
            break;
        case DE_WINK:
            printf("Wink event occurred on channel %s\n",
                VPMX_NAMEP(sr_getevtdev()));
            break;
        case DE_LCON:
            printf("Loop current ON event occurred on channel %s\n",
                VPMX_NAMEP(sr_getevtdev()));
            break;
        case DE_LCOFF:
            .
            .
    }

    /* Kick off next function in the state machine model. */

```



```

    .
    .
    return 0;
}

```

■ Errors

function 가 -1 error

ATDV_LASTERR() **ATDV_ERRMSGP()** .

EDX_BADPARAM Invalid Parameter
 EDX_SYSTEM Windows 2000 system error - check
 errno



CST Event Handling and Retrieval:

- vpm_getevt() - synchronous operation
- sr_getevtdatap() - asynchronous operation (*Standard Runtime Library*
· Programmer's Guide for Windows 2000)
- DX_CST data structure

Enabling User-Defined Tone Detection:

- vpm_addtone()

sets up the amplitudes

vpm_setgtdamp()

Name: void vpm_setgtdamp(gtd minampl1, gtd maxampl1, gtd minampl2, gtd maxampl2)

Inputs: short int gtd_minampl1 Minimum amplitude frequency
short int gtd_maxampl1 Maximum amplitude frequency
short int gtd_minampl2 Minimum amplitude frequency
short int gtd_maxampl2 Maximum amplitude frequency

Returns: void

Includes: srllib.h
smartsdk.h

Category: GTD Function

■ **Description**

vpm_setgtdamp() function general tone detection
amplitude set up . vpm_bld...() functions vpm_addtone()
.
vpm_bld...() function call .
function 가 vpm_addtone()
minimum amplitude -30, maximum amplitude 0 .

Default Value	Description	
GT_MIN_DEF	gtd_minampl* parameter Default	minimum GTD amplitude
GT_MAX_DEF	gtd_maxampl* parameter Default	maximum GTD amplitude

Parameter	Description
<hr/>	

gtd minampl1:	tone 1	minimum amplitude	dB	.
gtd maxampl1:	tone 1	maximum amplitude	dB	.
gtd minampl2:	tone 2	minimum amplitude	dB	.
gtd maxampl2:	tone 2	maximum amplitude	dB	.

■ Cautions

```
function 가 , tone amplitude
. amplitude default ,
GT_MIN_DEF GT_MAX_DEF .
```

■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
#include "voxlib.h" /* SCT voice library header file */

#define TID 1; /* Tone ID */

.
.
.
/*
* Set amplitude for GTD;
* freq1 -30dBm to 0 dBm
* freq2 -30dBm to 0 dBm
*/
vpm_setgtdamp(-30,0,-30,0);

/*
* Build temporary simple dual tone frequency tone of
* 950-1050 Hz and 475-525 Hz. using trailing edge detection, and
* -30dBm to 0dBm.
if (vpm_blddt(TID1, 1000, 50, 500, 25, TN_LEADING) == -1) {
    printf("Error building temporary tone: %d\n",vpm_errno);
    exit(3);
```


}

.
.
.

■ Errors

None.

vpm_sethook() provides control of the hookswitch status

Name:	int vpm_sethook(chdev, hookstate, mode)		
Inputs:	int chdev	valid SCT channel device	
		handle	
	int hookstate	hook state (on-hook or off-hook)	
	unsigned short mode	asynchronous/synchronous	
Returns:	0:		
	-1:		
Includes:	srllib.h		
	smartsdk.h		
Category:	Configuration		
Mode:	asynchronous/synchronous		

■ **Description**

vpm_sethook() function hookswitch . Hookswitch
on-hook off-hook 가 .

■ **Asynchronous Operation**

vpm_sethook() mode field EV_ASYNC
. function 0
event . event
SRL Event Management function .
TDX_SETHOOK event
. data structure cst_event field .
hookstate 가 on-hook DX_ONHOOK
hookstate 가 off-hook DX_OFFHOOK
DX_CST structure pointer sr_getevtdatap() .
Event Management Appendix A .
VPMX_HOOKST() hookstate event type .

■ **Synchronous Operation**

vpm_sethook() 0

The

Parameter	Description
chdev:	vpm_open() valid channel device handle
hookstate:	on-hook off-hook hookstate
mode:	DX_ONHOOK: set to on-hook state DX_OFFHOOK: set to off-hook state vpm_sethook() EV_ASYNC: Run vpm_sethook() asynchronously. EV_SYNC: Run vpm_sethook() synchronously (default).



None.

■ Example 1: Using vpm_sethook() in synchronous mode

```
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>

main()
{
    int chdev;
    /* open a channel with chdev as descriptor */
    if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
        /* process error */
    }

    /* put the channel on-hook */
    if (vpm_sethook(chdev,DX_ONHOOK,EV_SYNC) == -1) {
```



```

/* error setting hook state */
}
.
.
/* take the channel off-hook */
if (vpm_sethook(chdev,DX_OFFHOOK,EV_SYNC) == -1) {
/* error setting hook state */
}
.
.
}

```

■ Example 2: Using vpm_sethook() in asynchronous mode

```

#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>

#define MAXCHAN 24

int sethook_hdlr();

main()
{
    int i, chdev[MAXCHAN];
    char *chnamep;
    int srlmode;

    for (i=0; i<MAXCHAN; i++) {

        /* Set chnamep to the channel name - e.g, vpmB1C1, vpmB1C2,... */

        /* open a channel with chdev[i] as descriptor */
        if ((chdev[i] = vpm_open(chnamep,NULL)) == -1) {
            /* process error */
        }
    }
}

```



```

/* Using sr_enbhdr(), set up handler function to handle sethook
 * events on this channel.
 */
if (sr_enbhdr(chdev[i], TDX_SETHOOK, sethook_hdlr) == -1) {
    /* process error */
}

/* put the channel on-hook */
if (vpm_sethook(chdev[i], DX_ONHOOK, EV_ASYNC) == -1) {
    /* error setting hook state */
}
}

/* Use sr_waitevt() to wait for the completion of vpm_sethook().
 * On receiving the completion event, TDX_SETHOOK, control is transferred
 * to the handler function previously established using sr_enbhdr().
 */
.
.
}

int sethook_hdlr()
{
    DX_CST *cstp;

    /* sr_getevtdatap() points to the call status transition
     * event structure, which contains the hook state of the
     * device.
     */
    cstp = (DX_CST *)sr_getevtdatap();

    switch (cstp->cst_event) {
    case DX_ONHOOK:
        printf("Channel %s is ON hook\n", VPMX_NAMEP(sr_getevtdev()));
        break;

```



```

case DX_OFFHOOK:
    printf("Channel %s is OFF hook\n", VPMX_NAMEP(sr_getevtdev()));
    break;
default:
    /* process error */
    break;
}

/* Kick off next function in the state machine model. */
.
return 0;
}

```

■ Errors

function	–1	error
	ATDV_LASTERR()	ATDV_ERRMSGP()
EDX_BADPARAM	Invalid Parameter	
EDX_SYSTEM	Windows 2000 system error - check errno	

■ See Also

- *DX_CST* structure
- *sr_getevtdatap()* (*Standard Runtime Library Programmer's Guide for Windows 2000*, and *Appendix A*)
- *VPMX_HOOKST()*
- *DV_TPT* (*Appendix A*)

allows you to set the physical parameters vpm_setparm()

Name:	int vpm_setparm(dev,parm,valuep)		
Inputs:	int dev	valid SCT channel or board device handle	
	unsigned long parm	parameter type	
	void *valuep	parameter value	pointer
Returns:	0:		
	-1:		
Includes:	srllib.h smartsdk.h		
Category:	Configuration		



vpm_setparm() function off-hook delay, pause flash Character

parm 가 4 Voice Data

Structures and Device Parameters

vpm_setparm() idle

resources . board parameters

,

board parameters

- board open
- board channel close

function parameters

Parameter	Description		
dev:	channel	board 가 vpm_open()	open
		valid channel	board device handle

parm: board parameters . Board
channel parameter ,defaults 5.2. Clearing

Voice Structures

NOTE: parm parameter device 가 close reopen

valuep: channel board parameter pointer

NOTE: valuep driver
void* cast .



1. constant valuep .
valuep void* variable cast adress
pass .
2. channel parameters setting open idle state
..



Example

```
#include <srllib.h>
```

```
#include <smartsdk.h>
```

```
#include <windows.h>
```

```
main()
```

```
{
```

```
    int bddev, parmval;
```

```
    /* Open the board using vpm_open( ). Get board device descriptor in
```

```
    * bddev.
```

```
    */
```

```
    if ((bddev = vpm_open("vpmB1",NULL)) == -1) {
```

```
        /* process error */
```

```
    }
```

```
    /* Set the inter-ring delay to 6 seconds (default = 8) */
```

```
    parmval = 6;
```

```
    If (vpm_setparm(bddev, DXBD_R_IRD, (void *)&parmval) == -1) {
```

```
        /* process error */
```

```
    }
```

```
    /* now wait for an incoming ring */
```



```
    . . .  
}
```

■ Errors

ATDV_LASTERR() -1 error
ATDV_ERRMSGP() .

EDX_BADPARAM Invalid Parameter
EDX_SYSTEM Windows 2000 system error - check errno



- vpm_getparm()

structure . 20 DX_SVCB block .

NOTES:

- 1. function vpm_adjsv() . play
vpm_adjsv() . play
vpm_setsvcond() .
vpm_adjsv()
2. play 가 가

Parameter	Description
chdev:	vpm_open() valid channel device handle .
numblk:	DX_SVCB blocks 1 20
svcbp:	DX_SVCB structure pointer

■ **Cautions**

- 1. Condition block 가 .(20)
reset remove clear
reset play-speed 가
DTMF digit “1” “1” call
- 2. play volume digit digit buffer pass
vpm_getdig() VPMX_BUFDIGS()
- 3. play Digit
digit 가 play adjustment digitrk termination

■ **Example**

```
#include <stdio.h>
#include <errno.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
```



```

/*
* General Variables
*/
DX_SVCB svcb[ 10 ] = {
    /* BitMask AdjustmentSize AsciiDigit DigitType */
    { SV_SPEEDTBL | SV_RELCURPOS, 1, ' 1', 0 }, /* 1 */
    { SV_SPEEDTBL | SV_ABSPOS, -4, ' 2', 0 }, /* 2 */
    { SV_VOLUMETBL | SV_ABSPOS, 1, ' 3', 0 }, /* 3 */
    { SV_SPEEDTBL | SV_ABSPOS, 1, ' 4', 0 }, /* 4 */
    { SV_SPEEDTBL | SV_ABSPOS, 1, ' 5', 0 }, /* 5 */
    { SV_VOLUMETBL | SV_ABSPOS, 1, ' 6', 0 }, /* 6 */
    { SV_SPEEDTBL | SV_RELCURPOS, -1, ' 7', 0 }, /* 7 */
    { SV_SPEEDTBL | SV_ABSPOS, 6, ' 8', 0 }, /* 8 */
    { SV_VOLUMETBL | SV_RELCURPOS, -1, ' 9', 0 }, /* 9 */
    { SV_SPEEDTBL | SV_ABSPOS, 10, ' 0', 0 }, /* 10 */
};

main()
{
    int vpmdev;

    /*
    * Open the Voice Channel Device and Enable a Handler
    */
    if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
        perror( "vpmB1C1" );
        exit( 1 );
    }

    /*
    * Set Speed and Volume Adjustment Conditions
    */
    if ( vpm_setsvcond( vpmdev, 10, svcb ) == -1 ) {
        printf( "Unable to Set Speed and Volume" );
        printf( " Adjustment Conditions\n" );
        printf( "Lasterror = %d Err Msg = %s\n",
            ATDV_LASTERR( vpmdev ), ATDV_ERRMSGP( vpmdev ) );
    }
}

```



```

    vpm_close( vpmdev );
    exit( 1 );
}

/*
 * Continue Processing
 * .
 * .
 * .
 */

/*
 * Close the opened Voice Channel Device
 */
if ( vpm_close( vpmdev ) != 0 ) {
    perror( "close" );
}
/* Terminate the Program */
exit( 0 );
}

```

■ Errors

function	-1	error
	ATDV_LASTERR()	ATDV_ERRMSGP()
EDX_BADPARAM	Invalid Parameter	
EDX_BADPROD	Function not supported on this board	
EDX_SVADJBLKS	Invalid Number of Speed/Volume Adjustment blocks	
EDX_SYSTEM	Windows 2000 system error - check errno	



Setting Speed and Volume conditions:

- vpm_clrsvcond()
- *DX_SVCB* (Chapter 4. Voice Data Structures and Device Parameters)

Related to Speed and Volume:

- vpm_setsvmt()

- vpm_getcursv()
- vpm_getsvmt()
- "Speed and Volume Modification Tables" (*Voice Features Guide for Windows 2000*)
- vpm_adjsv()

updates the speed or volume

Name:	int vpm_setsvmt(chdev,tabletype,svmtp,flag)	
Inputs:	int chdev	valid channel device handle
	unsigned short tabletype	table to update (speed or volume)
	<i>DX_SVMT</i> * svmtp	pointer to <i>DX_SVMT</i>
	unsigned short flag	optional modification flag
Returns:	0 if success -1 if failure	
Includes:	srllib.h smartsdk.h	
Category:	Speed and Volume	

NOTE: default Speed Volume Modification
Voice Features Guide for Windows 2000 DX_SVMT
structure DX_SVMT -speed/volume modification table structure

function

Speed

Volume Modification Table

• speed volume

가 play volume

maximum level , minimum level

• Speed/Volume Modification Table default .default

Speed Volume Modification Table

the *Voice Features Guide for Windows 2000*

Parameter	Description
-----------	-------------

chdev: vpm_open() valid channel device handle

tabletype: Speed Volume Modification Table

SV_SPEEDTBL Update the Speed Modification Table values

SV_VOLUMETBL Update the Volume Modification Table values

sump: speed volume Speed/Volume Modification
Table pointer

flag: SV_WRAPMOD speed volume top level
bottom level

SV_SETDEFAULT Table default
table default

the *Voice Features Guide for Windows 2000*

DX_SVMT structure
svmtp

NOTE: SV_WRAPMOD SV_SETDEFAULT
flag 0

■ Example

```
#include <stdio.h>
#include <errno.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
```

```
/*
 * General Variables
 */
```

```
main()
```



```

{
    DX_SVMT svmt;
    int vpmdev, index;

    /*
    * Open the Voice Channel Device and Enable a Handler
    */
    if ( ( vpmdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
        perror( "vpmB1C1" );
        exit( 1 );
    }

    /*
    * Set up the Speed/Volume Modification
    */
    memset( &svmt, 0, sizeof( DX_SVMT ) );
    svmt.decrease[ 0 ] = -128;
    svmt.decrease[ 1 ] = -128;
    svmt.decrease[ 2 ] = -128;
    svmt.decrease[ 3 ] = -128;
    svmt.decrease[ 4 ] = -128;
    svmt.decrease[ 5 ] = -20;
    svmt.decrease[ 6 ] = -16;
    svmt.decrease[ 7 ] = -12;
    svmt.decrease[ 8 ] = -8;
    svmt.decrease[ 9 ] = -4;
    svmt.origin = 0;
    svmt.increase[ 0 ] = 4;
    svmt.increase[ 1 ] = 8;
    svmt.increase[ 2 ] = 10;
    svmt.increase[ 3 ] = -128;
    svmt.increase[ 4 ] = -128;
    svmt.increase[ 5 ] = -128;
    svmt.increase[ 6 ] = -128;
    svmt.increase[ 7 ] = -128;
    svmt.increase[ 8 ] = -128;

```



```

svmt.increase[ 9 ] = -128;

/*
 * Update the Volume Modification Table without Wrap Mode.
 */
if (vpm_setsvmt( vpmdev, SV_VOLUMETBL, &svmt, 0 ) == -1){
    printf( "Unable to Set the Volume" );
    printf( " Modification Table\n");
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( vpmdev ), ATDV_ERRMSG( vpmdev ) );
    vpm_close( vpmdev );
    exit( 1 );
}

/*
 * Continue Processing
 * .
 * .
 */

/*
 * Close the opened Voice Channel Device
 */
if ( vpm_close( vpmdev ) != 0 ) {
    perror( "close" );
}
/* Terminate the Program */
exit( 0 );
}

```

Errors

function	-1	error
	ATDV_LASTERR()	ATDV_ERRMSG()
		.
EDX_BADPARM	Invalid Parameter	
EDX_BADPROD	Function not supported on this board	

EDX_NONZEROSIZE	Reset to Default was Requested but size was non-zero
EDX_SPDVOL	Must Specify either SV_SPEEDTBL or SV_VOLUMETBL
EDX_SVMTRANGE	An Entry in <i>DX_SVMT</i> was out of Range
EDX_SVMTSIZE	Invalid Table Size Specified
EDX_SYSTEM	Windows 2000 system error - check errno



vpm_adjsv()

vpm_getcursv()

vpm_getsvmt()

"Speed and Volume Modification Tables" (*Voice Features Guide for Windows 2000*)

DX_SVMT (Chapter 4. Voice Data Structures and Device Parameters)

vpm_setuio()	allows an application to install a user I/O routine	
<hr/>		
Name:	int vpm_setuio(uioblk)	
Inputs:	uioblk	<i>DX_UIO</i> structure
Returns:	0: -1:	
Includes:	srllib.h smartsdk.h	
Category:	miscellaneous function	
<hr/>		

■
 vpm_setuio() function application I/O routine read(), write(), lseek()
 functions install . functions vpm_play() vpm_rec() functions
 storage media read write .

Application *DX_UIO* structure read(),
 write() lseek() function address .
 vpm_setuio() function install .

application *DX_IOTT* structure io_type field IO_UIO flag
 standard I/O function override . (Chapter
 4. Voice Data Structures and Device Parameters .)

NOTE: IO_UIO flag
 IO_DEV flag OR .

Record vpm_setuio() function write()
 Function . read() lseek() functions
 option .

play vpm_setuio() function read() function
 . write() lseek() functions option .

■
 application , I/O function
 .

■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
#include "VOXLIB.H"                /* SCT voice library header file */
int cd;                           /* channel descriptor */
DX_UIO myio;                      /* user definable I/O structure */
/*
 * User defined I/O functions
 */
int my_read9(fd,ptr,cnt)
int fd;
char * ptr;
unsigned cnt;
{
    printf("My read\n");
    return(read(fd,ptr,cnt));
}
/*
 * my write function
 */
int my_write(fd,ptr,cnt)
int fd;
char * ptr;
unsigned cnt;
{
    printf("My write \n");
    return(write(fd,ptr,cnt));
}
/*
 * my seek function
 */
long my_seek(fd,offset,whence)
int fd;
long offset;
```



```

int whence;
{
    printf("My seek\n");
    return(lseek(fd,offset,whence));
}
void main(argc,argv)
int argc;
char *argv[];
{
    .
    . /* Other initialization */
    .
    DX_UIO uioblk;
/* Initialize the UIO structure */
uioblk.u_read=my_read;
uioblk.u_write=my_write;
uioblk.u_seek=my_seek;
/* Install my I/O routines */
vpm_setuio(uioblk);
vodat_fd = vpm_fileopen("JUNK.VOX",O_RDWR|O_BINARY);
/*This block uses standard I/O functions */
iott->io_type = IO_DEV|IO_CONT
iott->io_fhandle = vodat_fd;
iott->io_offset = 0;
iott->io_length = 20000;
/*This block uses my I/O functions */
iott++;
iott->io_type = IO_DEV|IO_UIO|IO_CONT
iott->io_fhandle = vodat_fd;
iott->io_offset = 20001;
iott->io_length = 20000;
/*This block uses standard I/O functions */
iott++;
iott->io_type = IO_DEV|IO_CONT
iott->io_fhandle = vodat_fd;
iott->io_offset = 20002;

```



```

iott->io_length = 20000;
/*This block uses my I/O functions */
iott->io_type = IO_DEV|IO_UIO|IO_EOT
iott->io_fhandle = vodat_fd;
iott->io_offset = 10003;
iott->io_length = 20000;
devhandle = vpm_open("vpmB1C1", NULL);
vpm_sethook(devhandle, DX-ONHOOK,EV_SYNC)
vpm_wtring(devhandle,1,DX_OFFHOOK,EV_SYNC);
vpm_clrdigbuf;
    if(vpm_rec(devhandle,iott,(DX_TPT*)NULL,RM_TONE|EV_SYNC) == -1) {
        perror("");
        exit(1);
    }
vpm_clrdigbuf(devhandle);
    if(vpm_play(devhandle,iott,(DX_TPT*)NULL,EV_SYNC) == -1 {
        perror("");
        exit(1);
    }
    vpm_close(devhandle);

```

Errors

None.

forces termination of currently active I/O functions

vpm_stopch()

Name:	int vpm_stopch(chdev,mode)		
Inputs:	int chdev	valid SCT channel device handle	
	unsigned short mode	Reserved for future use	
Returns:	0 if success -1 if failure		
Includes:	srllib.h smartsdk.h		
Category:	I/O		
Mode:	asynchronous/synchronous		

■Description

vpm_stopch() function I/O function

. Busy idle 가 . chdev

idle vpm_stopch() .

process

vpm_stopch() .

process processing block

. process service .

vpm_stopch()가 I/O VPMX_TERMMSK()

TM_USRSTOP . call Analysis 가

vpm_dial() function Call Analysis

VPMX_CPTERM() . Call Analysis 가 vpm_stopch()

CR_STOPD .

.

Parameter	Description
chdev:	vpm_open() open

valid channel device handle .

mode: EV_ASYNC . stop .

driver vpm_stopch() 가 idle

sleep wait .



1. vpm_stopch()
 - vpm_dial() without Call Analysis enabled
 - vpm_wink()
2. Call Analysis 가 dialling channel vpm_stopch()

가 Call Analysis stop dial . stop

Call Analysis Extended Attribute function
3. vpm_stopch()가 I/O function

vpm_stopch()가 .
4. signal handler vpm_stopch() mode EV_ASYNC

■ Example

```
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>

main()
{
    int chdev, srlmode;

    /* Open the channel using vpm_open( ). Get channel device descriptor in
    * chdev.
    */
    if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
        /* process error */
    }
}
```



```

}

/* continue processing */
.
.

/* Force the channel idle. The I/O function that the channel is
 * executing will be terminated, and control passed to the handler
 * function previously enabled, using sr_enbhdr(), for the
 * termination event corresponding to that I/O function.
 * In the asynchronous mode, vpm_stopch() returns immediately,
 * without waiting for the channel to go idle.
 */
if ( vpm_stopch(chdev, EV_ASYNC) == -1) {
    /* process error */
}
}

```

■ Errors

	function	-1	error
	ATDV_LASTERR()	ATDV_ERRMSG()	
	.		
EDX_BADPARAM	Invalid Parameter		
EDX_SYSTEM	Windows 2000 system error - check errno		

■ See Also

Related I/O functions:

- vpm_dial()
- vpm_getdig()
- vpm_play()
- vpm_playf()
- vpm_playtone()

- vpm_rec()
- vpm_recf()
- vpm_wink()

Retrieving I/O termination reason due to vpm_stopch():

- VPMX_TERMMSK()
- VPMX_CPTERM() - vpm_dial() with Call Analysis

Name:	int vpm_wtcallid(chdev, rings, timeout,buffer)		
Input:	int chdev	channel device handle	
	Int rings	ring	
	Int timeout	timeout	
	Char buffer		buffer
Returns:	0 if success		
	-1 if failure		
Includes:	srllib.h		
	smartsdk.h		

```

      . (Message
      vpm_gtcallid      MDMF      .)
-   vpm_setpevtmsk()
-   vpm_getevt()
-   vpm_gtcallid()

```

Parameter	Description
Int chdev	channel device handle
Int rings	ring parameter
Int timeout	timeout parameter
char buffer	buffer

■ Example

```
#include "smartSdk.h"
```

```
void main()
{
    int ret;
    int dev;

    if((dev= vpm_open(" vpmB1C1" , NULL)) == -1) {
        return;
    }
}
```



```

}
printf("\n channel open = %d", dev);

int value = 1
if(vpm_setparm(dev, VPM_CALLID, (void *)&value) < 0){
    printf(" v[pm_setparm error\n" );
}
value = 2;
if(vpm_setparm(dev, VPMCH_RINGCNT ,(void *)&value) < 0{
    printf(" vpm_setparm error\n" );
}
if(vpm_setparm(dev, DM_RINGS) == -1)
{
    ;//ERROR Handling
}
vpm_sethook(chdev[ch],DX_ONHOOK, EV_SYNC);

printf("\n ring wait .. %d ch\n", dev);

unsigned char buffer[256];
memset(buffer,0,sizeof(buffer));
if(vpm_wtcallid(dev,1,-1,buffer) < 0{//2      ring      caller id
//
    Printf(" Caller ID Fail in ArsService()!!!\n" );
}
else{
    printf("\n ring detected..%d ch\n",dev);
    printf(" Caller ID : %s\n", buffer);

    if(vpm_wtring(dev,1,DX_OFFHOOK,-1) == -1){
        printf("\n dl_wtring error" );
        return;
    }
    printf("\n OFFHOOK %dch \n", dev);
}
}

```




vpm_gtcallid()

vpm_gtextcallid()

"Speed and Volume Modification Tables" (*Voice Features Guide for Windows 2000*)

DX_SVMT (Chapter 4. Voice Data Structures and Device Parameters

vpm_wtrng()

■ Description

```

vpm_wtring( ) function
on-hook off-hook . vpm_wtring( )
ring vpm_setevtsk( ), vpm_getevtsk( ), vpm_sethook( )
vpm_wtring( ) event 가 DM_RINGS

```

247

0 : ring event 가
-1 . vpm_wtring()



1. vpm_wtring() DM_RINGS event 가 detection event
process A 가 vpm_setevtmsk()
DM_SILON event , process B DM_RINGS
A DM_RINGS event .
2. channel on hook ring . Ring
vpm_wtring() vpm_sethook() . Ring
count .

NOTE: vpm_wtring()가 ring nring .
vpm_wtring()
application ring
application .

3. Ring Windows 2000 sigset() system call .

■ Example

```
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
```

```
main()
{
    int chdev; /* channel descriptor */
    .
    .
    /* Open Channel */
    if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
        /* process error */
    }

    /* Wait for two rings on this channel - no timeout */
    if (vpm_wtring(chdev,2,DX_OFFHOOK,-1) == -1) {
```



```

        /* process error */
    }
    .
    .
}

```

Errors

function	–1	error
	ATDV_LASTERR()	ATDV_ERRMSGP()
EDX_BADPARAM	Invalid Parameter	
EDX_SYSTEM	Windows 2000 system error - check errno	
EDX_TIMEOUT	Timeout limit is reached	



- `vpm_setevtmsk()`
- `vpm_getevt()`
- `vpm_sethook()`
- *DX_EBLK (Chapter 4. Voice Data Structures and Device Parameters)*

returns the duration of the answer VPMX_ANSRSIZ()

Name:	long	VPMX_ANSRSIZ(chdev)
Inputs:	int chdev	SCT device handle
Returns:	10ms unit	answer duration : AT_FAILURE :
Includes:	srllib.h smartsdk.h	
Category:	Extended Attribute	

■ **Description**

VPMX_ANSRSIZ() channel Basic Call Analysis 가
vpm_dial() . Ring Cadence 가
connection .
connection event 가 Ring Cadence 가
call ,
answering maching
가 Voice Features
Guide For Windows 2000 .
parameter .

Parameter	Description
chdev:	vpm_open() open valid channel device handle



■ **Example**

/* Call Analysis with user-specified parameters */


```

#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    int cares, chdev;
    DX_CAP capp;
    .
    .
    /* open the channel using vpm_open( ). Obtain channel device descriptor in
    * chdev
    */
    if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
        /* process error */
    }
    /* take the phone off-hook */
    if (vpm_sethook(chdev,DX_OFFHOOK,EV_SYNC) == -1) {
        /* process error */
    }
    /* Set the DX_CAP structure as needed for call analysis. Perform the
    * outbound dial with call analysis enabled
    */
    if ((cares = vpm_dial(chdev,"5551212",&capp,DX_CALLP|EV_SYNC)) == -1) {
        /* perform error routine */
    }
    switch (cares) {
    case CR_CNCT: /* Call Connected, get some additional info */
        printf("\nDuration of short low - %ld ms",
                VPMX_SHORTLOW(chdev)*10);
        printf("\nDuration of long low - %ld ms",
                VPMX_LONGLOW(chdev)*10);
        printf("\nDuration of answer - %ld ms",VPMX_ANSRSIZ(chdev)*10);
        break;
    case CR_BUSY:
        .

```



```

        .
    }
}

```

■ Errors

chdev handle , AT_FAILURE .



Related Call Analysis :

- vpm_dial()
- DX_CAP(chapter 4. Voice Data Structures and Device Parameters)
- “Call Analysis” (Voice Feature Guide For Windows 2000)

VPMX_BDNAMEP()

Name:	char * VPMX_ BDNAMEP (chdev)
Inputs:	int chdev SCT device handle
Returns:	Board name string pointer : “Unknown Device” ASCII string pointer :
Includes:	srllib.h smartsdk.h
Category:	Extended Attribute

■ Description

```

VPMX_BDNAMEP()      chdev 가      name
                     channel      open
                     .
                     parameter
                     .

```

Parameter	Description
chdev:	vpm_open() channel device handle
	open
	valid

■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
```



```

int chdev, bddev;
char *bdnamep;

.
.
/* Open the channel device */
if ((chdev = vpm_open("vpmB1C1", NULL)) == -1) {
    /* Process error */
}

/* Display board name */
bdnamep = VPMX_BDNAMEP(chdev);
printf("The board device is: %s\n", bdnamep);
/* Open the board device */
if ((bddev = vpm_open(bdnamep, NULL)) == -1) {
    /* Process error */
}

.
.
}

```

Errors

chdev handle , AT_FAILURE .

returns the device type

VPMX_BDTYPE()

Name:	long	VPMX_	BDTYPE	(chdev)
Inputs:	int	chdev		SCT device handle
Returns:	Board	channel	device type	: AT_FAILURE :
Includes:	srllib.h			smartsdk.h
Category:	Extended Attribute			

■ Description

VPMX_ BDTYPE () board channel device type .

device 가

parameter .

Parameter	Description		
chdev:	vpm_open()	open	valid
	channel device handle		
가	return value	or bit mask	.
	• SBT_VPM : VPM	.	
	• SBT_SPM : SPM	.	
	• SBT_FPM : FPM	.	
	• SHT_BOARD : Board	.	
	• SHT_CHANNEL : Channel	.	
VPM Board	open	SBT_VPM SHT_BOARD	가



■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
#define ON 1
main()
{
    int bddev;
    long bdtype;
    int call_analysis=0;
    /* Open the board device */
    if ((bddev = vpm_open("vpmB1",NULL)) == -1) {
        /* Process error */
    }
    if((bdtype = VPMX_BDTYPE(bddev)) == AT_FAILURE) {
        /* Process error */
    }
    if(bdtype == SBT_VPM|SHT_BOARD) {
        printf("Device is a VPM Board\n");
        call_analysis = ON;
    }
    .
    .
}
```

■ Errors

chdev handle , AT_FAILURE .

returns the number of unclooeected digits

VPMX_BUFDIGS()

Name:

long VPMX_ BUFDIGS (chdev)

Inputs:

int chdev SCT device handle

Returns:

digit buffer digit :

AT_FAILURE :

Includes:

srllib.h

smartsdk.h

Category:

Extended Attribute

■
 Description

VPMX_ BUFDIGS ()

chdev

channel

library

digit buffer

digit

.

vpm_getdigs()

digit

, vpm_clrdigbuf()

digit buffer

.

parameter

.

Parameter	Description
chdev:	vpm_open() channel device handle
	open
	valid



speed

volume

digit

digit buffer

.

■
 Example

```

#include <fcntl.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    int chdev;
```



```

long bufdigs;
DX_IOTT iott;
DV_TPT tpt[2];
/* Open the device using vpm_open( ). Get channel device descriptor in
* chdev. */
if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
    /* process error */
}
/* set up DX_IOTT */
iott.io_type = IO_DEV|IO_EOT;
iott.io_bufp = 0;
iott.io_offset = 0;
iott.io_length = -1; /* play till end of file */
if((iott.io_fhandle = vpm_fileopen("prompt.vox", O_RDONLY)) == -1) {
    /* process error */
}
/* set up DV_TPT */
vpm_clrtp(tpt,2);
tpt[0].tp_type = IO_CONT;
tpt[0].tp_termno = DX_MAXDTMF; /* Maximum digits */
tpt[0].tp_length = 4; /* terminate on 4 digits */
tpt[0].tp_flags = TF_MAXDTMF; /* Use the default flags */
tpt[1].tp_type = IO_EOT;
tpt[1].tp_termno = DX_DIGMASK; /* Digit termination */
tpt[1].tp_length = DM_5; /* terminate on the digit "5" */
tpt[1].tp_flags = TF_DIGMASK; /* Use the default flags */
/* Play a voice file. Terminate on receiving 4 digits, the digit "5" or
* at end of file.*/
if (vpm_play(chdev,&iott,tpt,EV_SYNC) == -1) {
    /* process error */
}
/* Check # of digits collected and continue processing. */
if((bufdigs=VPMX_BUFDIGS(chdev))==AT_FAILURE) {
    /* process error */
}
.

```



```

        .
        .
    }

```

■ Errors

chdev handle , AT_FAILURE .



Other digit functions :

- vpm_getdig()
- vpm_clrdigbuf()

VPMX_CHNAMES()

Name:	char ** VPMX_ CHNAMES (bddev)
Inputs:	int bddev SCT Board device handle
Returns:	channel name array pointer : “Unknown Device” 가 array pointer :
Includes:	srllib.h smartsdk.h
Category:	Extended Attribute

■ Description

```
VPMX_ CHNAMES ()          bddev          channel    name array    pointe
.
                                board device      channel device
.
                                parameter          .
```

Parameter	Description
bddev:	vpm_open() board device handle

■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
```



```

int bddev, cnt;
char **chnames;
long subdevs;

.
.
/* Open the board device */
if ((bddev = vpm_open("vpmB1",NULL)) == -1) {
    /* Process error */
}

.
.
/* Display channels on board */
chnames = VPMX_CHNAMES(bddev);
subdevs = ATDV_SUBDEVS(bddev); /* number of sub-devices on board */
printf("Channels on this board are:\n");
for(cnt=0; cnt<subdevs; cnt++) {
    printf("%s\n",*(chnames + cnt));
}
/* Call vpm_open( ) to open each of the
 * channels and store the device descriptors
 */
.
.
}

```

Errors

bddev	,	"Unknown Device"
array pointer	.	

returns the channel number

VPMX_CHNUM()

Name: long VPMX_ CHNUM (chdev)
Inputs: int chdev SCT channel device handle
Returns: channel number :
AT_FAILURE :
Includes: srllib.h
smartsdk.h
Category: Extended Attribute

■ **Description**

VPMX_ CHNUM () VPM Board chdev channel .
channel 1 .
channel number channel
array index .
parameter .

Parameter	Description
chdev:	vpm_open() channel open valid channel device handle



■ **Example**

```
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    int chdev;
```



```

long chno;

.
/* Open the channel device */
if ((chdev = vpm_open("vpmB1C1", NULL)) == -1) {
/* Process error */
}
/* Get Channel number */
if((chno = VPMX_CHNUM(chdev)) == AT_FAILURE) {
/* Process error */
}
/* Use chno for application-specific purposes */
.
.
}

```

Errors

chdev handle , AT_FAILURE .

returns the connection

VPMX_CONTYPE()

Name:	long	VPMX_ CONTYPE (chdev)
Inputs:	int chdev	SCT channel device handle
Returns:	connection type	: AT_FAILURE :
Includes:	srllib.h	smartsdk.h
Category:	Extended Attribute	

■ Description

VPMX_ CONTYPE () channel device call , call
type . Call Analysis 가 vpm_dial() ADTX_CPTERM()
CR_CNCT , .

가 .
CON_CAD : Cadence 가 break Connection
CON_PVD : Positive Voice Detection Connection.

parameter .

Parameter	Description
chdev:	vpm_open() channel open valid channel device handle



■ Example

```
#include <stdio.h>
#include <errno.h>
#include <srllib.h>
#include <smartsdk.h>
```



```

#include <windows.h>
main()
{
    int chdev;
    int cares;
    /*
    * Open the Voice Channel Device and Enable a Handler
    */
    if ( (chdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
        perror( "vpmB1C1" );
        exit( 1 );
    }
    /*
    * Delete any previous tones
    */
    if ( vpm_deltone(chdev) < 0 ) {
        /* handle error */
    }
    /*
    * Now enable Enhanced call progress with above changed settings.
    */
    if (vpm_initcallp(chdev)) {
        /* handle error */
    }
    /*
    * Take the phone off-hook
    */
    if ( vpm_sethook(chdev, DX_OFFHOOK, EV_SYNC ) == -1 ) {
        printf( "Unable to set the phone off-hook\n" );
        printf( "Lasterror = %d Err Msg = %s\n",
            ATDV_LASTERR(chdev), ATDV_ERRMSG(chdev) );
        vpm_close(chdev);
        exit( 1 );
    }
    /*
    * Perform an outbound dial with call analysis, using

```



```

* the default call analysis parameters.
*/
if ((cares=vpm_dial(chdev, ",84",(DX_CAP *)NULL, DX_CALLP ) ) == -1 ) {
    printf( "Outbound dial failed - reason = %d\n",
    vpm_close(chdev);
    exit( 1 );
}
printf( "Call Analysis returned %d\n", cares );
if ( cares == CR_CNCT ) {
    switch ( VPMX_CONNTYPE(chdev) ) {
    case CON_CAD:
        printf( "Cadence Break\n" );
        break;
    case CON_PVD:
        printf( "Positive Voice Detection\n" );
        break;
    default:
        printf( "Unknown connection type\n" );
        break;
    }
}
/*
* Continue Processing
* .
* .
* .
*/
/*
* Close the opened Voice Channel Device
*/
if ( vpm_close(chdev) != 0 ) {
    perror( "close" );
}
/* Terminate the Program */
exit( 0 );
}

```


■ Errors

chdev handle , AT_FAILURE .



Related to Call Analysis :

- vpm_dial()
- VPMX_CPTerm()
- DX_CAP structure(Chapter 4. Voice Data Structures and Device Parameters)
- “Call Analysis” (Voice Feature for Windows 2000)

returns last Call Analysis termination

VPMX_CPTERM()

Name: long VPMX_CPTERM(chdev)
Inputs: int chdev SCT channel device handle
Returns: call analysis termination :
AT_FAILURE :
Includes: srllib.h
smartsdk.h
Category: Extended Attribute

■ Description

VPMX_CPTERM () channel device Call Analysis
termination . Call Anlaysis 가 dila Call Status

parameter .

Parameter Description

chdev: vpm_open() channel open valid
channel device handle

가 .

CR_BUSY	• Busy
CR_CNCT	• Connect
CR_FAXTONE	• Called line answered by fax machine or modem
CR_NOANS	• No answer
CR_NODIALTONE	• Timeout occurred while waiting for dial tone
CR_NORB	• No ringback
CR_STOPD	• Stopped
CR_ERROR	• Error



■ Example

```
/* Call Analysis with user-specified parameters */
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    int chdev;
    DX_CAP capp;
    .
    .
    /* open the channel using vpm_open( ). Obtain channel device descriptor
    in
    * chdev
    */
    if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
        /* process error */
    }
    /* take the phone off-hook */
    if (vpm_sethook(chdev,DX_OFFHOOK,EV_SYNC) == -1) {
        /* process error */
    } else {
        /* Clear DX_CAP structure */
        vpm_clrkap(&capp);
        /* Set the DX_CAP structure as needed for call analysis.
        * Allow 3 rings before no answer.
        */
        capp.ca_nbrdna = 3;
        /* Perform the outbound dial with call analysis enabled. */
        if (vpm_dial(chdev,"5551212",&capp,DX_CALLP|EV_SYNC) == -1) {
            /* perform error routine */
        }
    }
    .
    .
    /* Examine last call progress termination on the device */
}
```



```

switch (VPMX_CPTerm(chdev)) {
case CR_CNCT: /* Call Connected, get some additional info */
    .
    break;
    .
    .
case AT_FAILURE: /* Error */
}
}

```

■ Errors

chdev handle , AT_FAILURE .



Related to Call Analysis :

- vpm_dial()
- DX_CAP structure(Chapter 4. Voice Data Structures and Device Parameters)
- “Call Analysis” (Voice Feature for Windows 2000)

returns the tone identifier

VPMX_CRTNID()

Name:	long	VPMX_ CRTNID (chdev)
Inputs:	int chdev	SCT channel device handle
Returns:	Call Analysis	tone :
	AT_FAILURE :	
Includes:	srllib.h	
	smartsdk.h	
Category:	Extended Attribute	

■ Description

VPMX_CRTNID() channel device Call Analysis tone
tone id . DSP board Perfect Call CallAnalysis
. Perfect Call CallAnaysis Voice Features Guide
For Windows 2000 .

parameter .

Parameter	Description
chdev:	vpm_open() channel open valid channel device handle

- 가 .
- | | |
|---------------|----------------------------|
| TID_DIAL_LCL | • Local dial tone |
| TID_DIAL_INTL | • International dial tone |
| TID_DIAL_XTRA | • Special dial tone |
| TID_BUSY1 | • First signal busy |
| TID_BUSY2 | • Second signal busy |
| TID_RINGBK1 | • Ringback |
| TID_FAX1 | • First fax or modem tone |
| TID_FAX2 | • Second fax or modem tone |



■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    DX_CAP cap_s;
    int ddd, car;
    char *chnam, *dialstrg;
    chnam = "vpmB1C1";
    dialstrg = "L1234";
    /*
    * Open channel
    */
    if ((ddd = vpm_open( chnam, NULL )) == -1 ) {
        /* handle error */
    }
    /*
    * Delete any previous tones
    */
    if ( vpm_deltone(ddd) < 0 ) {
        /* handle error */
    }
    /*
    * Now enable Enhanced call progress with above changed settings.
    */
    if (vpm_initcallp( ddd )) {
        /* handle error */
    }
    /*
    * Set off Hook
    */
    if ((vpm_sethook( ddd, DX_OFFHOOK, EV_SYNC )) == -1) {
        /* handle error */
    }
}
```



```

/*
 * Dial
 */
printf("Dialing %s\n", dialstrg );
car = vpm_dial(ddd,dialstrg,(DX_CAP *)&cap_s,DX_CALLP|EV_SYNC);
if (car == -1) {
    /* handle error */
}
switch( car ) {
case CR_NODIALTONE:
    switch( VPMX_DTNFAIL(ddd) ) {
        case ' L' :
            printf(" Unable to get Local dial tone\n");
            break;
        case ' I' :
            printf(" Unable to get International dial tone\n");
            break;
        case ' X' :
            printf(" Unable to get special eXtra dial tone\n");
            break;
    }
    break;
case CR_BUSY:
    printf(" %s engaged - %s detected\n", dialstrg,
(VPMX_CRTNID(ddd) == TID_BUSY1 ? "Busy 1" : "Busy 2") );
    break;
case CR_CNCT:
    printf(" Successful connection to %s\n", dialstrg );
    break;
default:
    break;
}
/*
 * Set on Hook
 */
if ((vpm_sethook( ddd, DX_ONHOOK, EV_SYNC )) == -1) {

```



```

        /* handle error */
    }
    vpm_close( ddd );
}

```

■ Errors

chdev handle , AT_FAILURE .

returns device type VPMX_DEVTYPE()

Name:	long	VPMX_	DEVTYPE	(chdev)
Inputs:	int dev		SCT	device handle
Returns:	device type	:		
	AT_FAILURE	:		
Includes:	srllib.h			
	smartsdk.h			
Category:	Extended Attribute			

■ **Description**

VPMX_DEVTYPE() Board channel device type .

	parameter	.
Parameter	Description	

dev:	vpm_open()	open	channel	board
	device handle			

- 가 .
- DT_DXBD • Board device
 - DT_DXCH • Channel



■ **Example**

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    int bddev;
    long devtype;
```



```

/* Open the board device */
if ((bddev = vpm_open("vpmB1",NULL)) == -1) {
    /* Process error */

}
if((devtype = VPMX_DEVTTYPE(bddev)) == AT_FAILURE) {
    /* Process error */

}
if(devtype == DT_DXBD) {
    printf("Device is a Board\n");
}
/* Continue processing */
.
.
}

```

Errors

chdev handle , AT_FAILURE .

returns character for dial tone

VPMX_DTNFAIL()

Name: long VPMX_ DTNFAIL(chdev)
Inputs: int chdev SCT channel device handle
Returns: dial tone code :
AT_FAILURE :
Includes: srllib.h
smartsdk.h
Category: Extended Attribute

■ Description

VPMX_DTNFAIL() PerfectCall Call Analysis dial tone
code . PerfectCall Call Anaysis .

Parameter	Description
-----------	-------------

chdev: vpm_open() open channel
device handle

- 가 .
- Local dial tone
 - International dial tone
 - Special dial tone



■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
```



```

DX_CAP cap_s;
int ddd, car;
char *chnam, *dialstrg;
chnam = "vpmB1C1";
dialstrg = "L1234";
/*
* Open channel
*/
if ((ddd = vpm_open( chnam, NULL )) == -1 ) {
    /* handle error */
}
/*
* Delete any previous tones
*/
if ( vpm_deltone(ddd) < 0 ) {
    /* handle error */
}
/*
* Now enable Enhanced call progress with above changed settings.
*/
if (vpm_initcallp( ddd )) {
    /* handle error */
}
/*
* Set off Hook
*/
if ((vpm_sethook( ddd, DX_OFFHOOK, EV_SYNC )) == -1) {
    /* handle error */
}
/*
* Dial
*/
printf("Dialing %s\n", dialstrg );
car = vpm_dial(ddd,dialstrg,(DX_CAP *)&cap_s,DX_CALLP|EV_SYNC);
if (car == -1) {
    /* handle error */
}

```



```

}
switch( car ) {
case CR_NODIALTONE:
    switch( VPMX_DTNFAIL(ddd) ) {
        case 'L' :
            printf(" Unable to get Local dial tone\n");
            break;
        case 'I' :
            printf(" Unable to get International dial tone\n");
            break;
        case 'X' :
            printf(" Unable to get special eXtra dial tone\n");
            break;
    }
    break;
case CR_BUSY:
    printf(" %s engaged - %s detected\n", dialstrg,
        VPMX_CRTNID(ddd) == TID_BUSY1 ? "Busy 1" : "Busy 2" );
    break;
case CR_CNCT:
    printf(" Successful connection to %s\n", dialstrg );
    break;
default:
    break;
}
/*
 * Set on Hook
 */
if ((vpm_sethook( ddd, DX_ONHOOK, EV_SYNC )) == -1) {
    /* handle error */
}
vpm_close( ddd );
}

```

Errors

chdev handle , AT_FAILURE .

VPMX_FWVER()

Name:	char * VPMX_ FWVER(bddev)		
Inputs:	int bddev	SCT	board device handle
Returns:	VPM Board F/W version : “ Unknown Device” string pointer :		
Includes:	srllib.h smartsdk.h		
Category:	Extended Attribute		

■ Description

```
VPMX_FWVER()      VPM Board      F/W version      string
pointer           .
parameter         .
```

Parameter	Description
bddev:	vpm_open() device handle

■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>

main()
{
    int bddev;
    char fwver[100];
```



```

    .
    /* Open the board device */
    if ((bddev = vpm_open("vpmB1",NULL)) == -1) {
        /* Process error */
    }
    .
    .
    /* Display Firmware version number */
    strcpy(fwver,VPMX_FWVER(bddev))
    printf("Firmware version %s\n",fwver);
    .
    .
}

```

Errors

```

chdev          handle          ,          AT_FAILURE          .

```


returns the current hook state

VPMX_HOOKST()

Name: long VPMX_HOOKST(chdev)
Inputs: int chdev SCT channel device handle
Returns: channel hook state :
AT_FAILURE :
Includes: srllib.h
smartsdk.h
Category: Extended Attribute

■ **Description**

VPMX_HOOKST() hook state .
parameter .

Parameter	Description
-----------	-------------

chdev:	vpm_open() open channel device handle
---------------	---

가 .
DX_OFFHOOK • off-hook
DX_ONHOOK • on-hook



■ **Example**

```
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    int chdev;
    long hookst;
```



```

/* Open the channel device */
if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
    /* Process error */

}
.
.
/* Examine Hook state of the channel. Perform application specific action */
if((hookst = VPMX_HOOKST(chdev)) == AT_FAILURE) {
    /* Process error */

}
if(hookst == DX_OFFHOOK) {
    /* Channel is Off-hook */

}
.
.
}

```

Errors

chdev handle , AT_FAILURE .

returns a bitmapped representation of activity

VPMX_LINEST()

Name: long VPMX_HOOKST(chdev)
Inputs: int chdev SCT channel device handle
Returns: channel line state :
AT_FAILURE :
Includes: srllib.h
smartsdk.h
Category: Extended Attribute

■ **Description**

VPMX_LINEST() channel line activity bitmap
.
parameter .

Parameter	Description
-----------	-------------

chdev:	vpm_open() open channel device handle
---------------	---

가 .

RLS_SILENCE	• Silence on the line
RLS_DTMF	• present
RLS_LCSENSE	• not present
RLS_RING	• Ring not present
RLS_HOOK	• Channel is on-hook
RLS_RINGBK	• Audible ringback detected



■ **Example**

```
#include <srllib.h>
#include <smartsdk.h>
```



```

#include <windows.h>
main()
{
    int chdev;
    long linest;
    /* Open the channel device */
    if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
        /* Process error */
    }
    /* Examine line status bitmap of the channel. Perform application-specific
    * action
    */
    if((linest = VPMX_LINEST(chdev)) == AT_FAILURE) {
        /* Process error */
    }
    if(linest & RLS_LCSENSE) {
        /* No loop current */
    }
    .
    .
}

```

■ Errors

chdev handle , AT_FAILURE .



- vpm_sethook()
- DX_CST (Chapter 4. Voice Data Structure and Device Parameters)

returns duration of the longer silence VPMX_LONGLOW()

Name:	long	VPMX_ LONGLOW(chdev)
Inputs:	int chdev	SCT channel device handle
Returns:	duration	:
	AT_FAILURE	:
Includes:	srllib.h	
	smartsdk.h	
Category:	Extended Attribute	

■ **Description**

VPMX_LONGLOW() channel Call Analysis
signal 10ms . VPMX_SIZEHI()
VPMX_SHORTLOW cadence
. Voice Features Guide For Windows 2000 .
parameter .

Parameter	Description
-----------	-------------

chdev:	vpm_open() open channel device handle
---------------	---



■ **Example**

```
/* Call Analysis with user-specified parameters */
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    int cares, chdev;
```



```

DX_CAP capp;
.
.

/* open the channel using vpm_open( ). Obtain channel device descriptor in
 * chdev
 */
if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
    /* process error */
}
/* take the phone off-hook */
if (vpm_sethook(chdev,DX_OFFHOOK,EV_SYNC) == -1) {
    /* process error */
}
/* Set the DX_CAP structure as needed for call analysis. Perform the
 * outbound dial with call analysis enabled
 */
if ((cares = vpm_dial(chdev,"5551212",&capp,DX_CALLP|EV_SYNC)) == -1) {
    /* perform error routine */
}
switch (cares) {
    case CR_CNCT: /* Call Connected, get some additional info */
        printf("\nDuration of short low - %ld ms",
                VPMX_SHORTLOW(chdev)*10);
        printf("\nDuration of long low - %ld ms",
                VPMX_LONGLOW(chdev)*10);
        printf("\nDuration of answer - %ld ms",
                VPMX_ANSRSIZ(chdev)*10);
        break;
    case CR_BUSY:
.
.
}
}

```

Errors

```

chdev          handle          ,          AT_FAILURE          .

```




- vpm_dial()
- VPMX_CPTerm()
- DX_CAP structure (Chapter 4. Voice Data Structure and Device Parameters)
- “Call Analysis” (Voice Features Guide For Windows 2000)
- “Cadence Detection” (Voice Features Guide For Windows 2000)

returns the physical address VPMX_PHYADDR()

Name:	long	VPMX_ PHYADDR(chdev)
Inputs:	int chdev	SCT channel device handle
Returns:	physical address	:
	AT_FAILURE	:
Includes:	srllib.h	
	smartsdk.h	
Category:	Extended Attribute	

■ **Description**

VPMX_PHYADDR() Board Physical address .

parameter .

Parameter	Description
chdev:	vpm_open() open channel device handle



■ **Example**

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    int bddev;
    long phyaddr;
    /* Open the board device */
    if ((bddev = vpm_open("vpmB1",NULL)) == -1) {
        /* Process error */
    }
```



```

    }
    if((phyaddr = VPMX_PHYADDR(bddev)) == AT_FAILURE) {
        /* Process error */
    }
    printf("Board is at address %X\n",phyaddr);
    .
    .
}

```

Errors

chdev handle , AT_FAILURE .

returns duration of shorted silence VPMX_SHORTLOW()

Name:	long	VPMX_ SHORTLOW(chdev)
Inputs:	int chdev	SCT channel device handle
Returns:	duration	:
	AT_FAILURE	:
Includes:	srllib.h	
	smartsdk.h	
Category:	Extended Attribute	

■ **Description**

VPMX_SHORTLOW() channle Call Analysis
signal 10ms . VPMX_SIZEHI()
VPMX_SHORTLOW cadence
. Voice Features Guide For Windows 2000 .

 parameter .

Parameter	Description
-----------	-------------

chdev:	vpm_open() open channel device handle
---------------	---



■ **Example**

```
/* Call Analysis with user-specified parameters */
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    int cares, chdev;
```



```

DX_CAP capp;
.
/* open the channel using vpm_open( ). Obtain channel device descriptor
 * in chdev
 */
if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
    /* process error */
}
/* take the phone off-hook */
if (vpm_sethook(chdev,DX_OFFHOOK,EV_SYNC) == -1) {
    /* process error */
}
/* Set the DX_CAP structure as needed for call analysis. Perform the
 * outbound dial with call analysis enabled
 */
if ((cares = vpm_dial(chdev,"5551212",&capp,DX_CALLP|EV_SYNC)) == -1) {
    /* perform error routine */
}
switch (cares) {
case CR_CNCT: /* Call Connected, get some additional info */
    printf("\nDuration of short low - %ld ms",
           VPMX_SHORTLOW(chdev)*10);
    printf("\nDuration of long low - %ld ms",
           VPMX_LONGLOW(chdev)*10);
    printf("\nDuration of answer - %ld ms",VPMX_ANSRSIZ(chdev)*10);
    break;
case CR_BUSY:
.
.
}
}

```

■ Errors

chdev handle , AT_FAILURE .



- vpm_dial()

- VPMX_LONGLOW()
- VPMX_SIZEHI()
- VPMX_CPTERM()
- DX_CAP structure
- “Call Analysis” (Voice Features Guide For Windows 2000)
- “Cadence Detection” (Voice Features Guide For Windows 2000)

returns duration of initial non-silence

VPMX_SIZEHI()

Name:long VPMX_SIZEHI(chdev)

Inputs:int chdevSCT channel device handle

Returns:10ms unit non-silence duration :
AT_FAILURE :

Includes:srllib.h
smartsdk.h

Category:Extended Attribute

■ Description

VPMX_SIZEHI()channelCall Analysisnon-silence10ms.VPMX_SHORTLOW,VPMX_LONGLOW()cadence

Voice Features Guide For Windows 2000

parameter.

Parameter	Description
chdev:	vpm_open() open channel device handle



■ Example

```
/* Call Analysis with user-specified parameters */
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    int cares, chdev;
```



```

DX_CAP capp;
.
.
/* open the channel using vpm_open( ). Obtain channel device descriptor
* in chdev
*/
if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
    /* process error */
}
/* take the phone off-hook */
if (vpm_sethook(chdev,DX_OFFHOOK,EV_SYNC) == -1) {
    /* process error */
}
/* Set the DX_CAP structure as needed for call analysis. Perform the
* outbound dial with call analysis enabled
*/
if ((cares = vpm_dial(chdev,"5551212",&capp,DX_CALLP|EV_SYNC)) == -1) {
    /* perform error routine */
}
switch (cares) {
case CR_CNCT: /* Call Connected, get some additional info */
    printf("\nDuration of short low - %ld ms",
           VPMX_SHORTLOW(chdev)*10);
    printf("\nDuration of long low - %ld ms",
           VPMX_LONGLOW(chdev)*10);
    printf("\nDuration of non-silence - %ld ms",
           VPMX_SIZEHI(chdev)*10);
    break;
case CR_BUSY:
.
.
}
}

```

Errors

chdev handle , AT_FAILURE .



- vpm_dial()
- VPMX_LONGLOW()
- VPMX_SHORTLOW()
- VPMX_CPTERM()
- DX_CAP structure
- “Call Analysis” (Voice Features Guide For Windows 2000)
- “Cadence Detection” (Voice Features Guide For Windows 2000)

returns the current state

VPMX_STATE()

Name:	long	VPMX_STATE(chdev)
Inputs:	int chdev	SCT channel device handle
Returns:	channel	:
	AT_FAILURE	:
Includes:	srllib.h	
	smartsdk.h	
Category:	Extended Attribute	

■ Description

VPMX_STATE() channel .

	parameter	.
Parameter	Description	

chdev:	vpm_open()	open	channel
	device handle		



- 가 .
- | | |
|----------|-----------------------|
| CS_DIAL | • Dial state |
| CS_CALL | • Call state |
| CS_GTDIG | • Get Digit State |
| CS_HOOK | • Hook state |
| CS_IDLE | • Idle state |
| CS_RECD | • Record state |
| CS_STOPD | • Stopped state |
| CS_TONE | • Playintg tone state |

NOTE : 가 fax board ,
가 .

- | | |
|------------|---|
| CS_SENDFAX | • |
| CS_RECVFAX | • |

NOTE : I/O 가 device 가 idle .



■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <smartsdk.h>
m#include <windows.h>
main()
{
    int chdev;
    long chstate;
    /* Open the channel device */
    if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
        /* Process error */
    }
    .
    .
    /* Examine state of the channel. Perform application specific action based
    * on state of the channel
    */
    if((chstate = VPMX_STATE(chdev)) == AT_FAILURE) {
        /* Process error */
    }
    printf("current state of channel %s = %ld\n", VPMX_NAMEP(chdev), chstate);
    .
    .
}
```

■ Errors

chdev handle , AT_FAILURE .

returns a bitmap VPMX_TERMMSK()

Name:	long	VPMX_TERMMSK(chdev)
Inputs:	int chdev	SCT channel device handle
Returns:	channel	termination bitmap : AT_FAILURE :
Includes:	srllib.h	
	smartsdk.h	
Category:	Extended Attribute	

■ **Description**

VPMX_TERMMSK ()	channle	termination
bitmap	bitmap	I/O function
	parameter	reset
Parameter	Description	

chdev:	vpm_open()	open	channel
	device handle		



가	.
TM_NORMTERM	• Normal Termination (for vpm_dial() , vpm_sethook())
TM_MAXDTMF	• Maximum DTMF count
TM_MAXSIL	• Maximum period of silence
TM_MAXNOSIL	• Maximum period of non-silence
TM_LCOFF	• Loop current off
TM_IDDTIME	• Inter-digit delay
TM_MAXTIME	• Maximum function time
TM_DIGIT	• Specific digit received
TM_USRSTOP	• Function stopped byu user
TM_EOD	• End of Data reached on playback
TM_TONE	• Tone-on/off event

TM_ERROR

• I/O Device Error



, bit termination

bitmap

■ Example

```
#include <stdio.h>
#include <fcntl.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    int chdev;
    long term;
    DX_IOTT iott;
    DV_TPT tpt[4];
    /* Open the channel device */
    if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
        /* Process error */
    }
    /* Record a voice file. Terminate on receiving a digit, silence, loop
    * current drop, max time, or reaching a byte count of 50000 bytes.
    */
    /* set up DX_IOTT */
    iott.io_type = IO_DEV|IO_EOT;
    iott.io_bufp = 0;
    iott.io_offset = 0;
    iott.io_length = 50000;
    if((iott.io_fhandle = vpm_fileopen("file.vox", O_RDWR)) == -1) {
        /* process error */
    }
    /* set up DV_TPTs for the required terminating conditions */
    vpm_clrtp(tpt,4);
    tpt[0].tp_type = IO_CONT;
```



```

tpt[0].tp_termno = DX_MAXDTMF; /* Maximum digits */
tpt[0].tp_length = 1; /* terminate on the first digit */
tpt[0].tp_flags = TF_MAXDTMF; /* Use the default flags */
tpt[1].tp_type = IO_CONT;
tpt[1].tp_termno = DX_MAXTIME; /* Maximum time */
tpt[1].tp_length = 100; /* terminate after 10 secs */
tpt[1].tp_flags = TF_MAXTIME; /* Use the default flags */
tpt[2].tp_type = IO_CONT;
tpt[2].tp_termno = DX_MAXSIL; /* Maximum Silence */
tpt[2].tp_length = 30; /* terminate on 3 sec silence */
tpt[2].tp_flags = TF_MAXSIL; /* Use the default flags */
tpt[3].tp_type = IO_EOT; /* last entry in the table */
tpt[3].tp_termno = DX_LCOFF; /* terminate on loop current drop */
tpt[3].tp_length = 10; /* terminate on 1 sec silence */
tpt[3].tp_flags = TF_LCOFF; /* Use the default flags */
/* Now record to the file */
if (vpm_rec(chdev,&iott,tpt,EV_SYNC) == -1) {
    /* process error */
}
/* Examine bitmap to determine if digits caused termination */
if((term = VPMX_TERMMSK(chdev)) == AT_FAILURE) {
    /* Process error */
}
if(term & TM_MAXDTMF) {
    printf("Terminated on digits\n");
    .
    .
}
}

```

■ Errors

chdev handle , AT_FAILURE .



Setting Termination Conditions :

- DV_TPT(Appendix A)

Retrieving Termination Events-asynchronously:

- Event Management Functions(Standard Runtime Library Programmer' s Guide for Windows 2000)

returns the user-defined tone id

VPMX_TONEID()

Name: long VPMX_TONEID(chdev)
Inputs: int chdev SCT channel device handle
Returns: user-defined tone id :
AT_FAILURE :
Includes: srllib.h
smartsdk.h
Category: Extended Attribute

■ **Description**

VPMX_TONEID() user-defined tone id .
VPMX_TERMMSK() 가 user-sedcified tone I/O function
DX_TONE , tone id .

parameter .
Parameter Description

chdev:	vpm_open() device handle	open	channel
---------------	------------------------------	------	---------



■ **Example**

```
#include <stdio.h>
#include <errno.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
#define TID_1 101
main()
{
    TN_GEN tngen;
    DV_TPT tpt[ 5 ];
```



```

int chdev;
/*
 * Open the D/xxx Channel Device and Enable a Handler
 */
if ( ( chdev = vpm_open( "vpmB1C1", NULL ) ) == -1 ) {
    perror( "vpmB1C1" );
    exit( 1 );
}
/*
 * Describe a Simple Dual Tone Frequency Tone of 950-
 * 1050 Hz and 475-525 Hz using leading edge detection.
 */
if ( vpm_blddt( TID_1, 1000, 50, 500, 25, TN_LEADING ) == -1 ) {
    printf( "Unable to build a Dual Tone Template\n" );
}
/*
 * Add the Tone to the Channel
 */
if ( vpm_addtone( chdev, NULL, 0 ) == -1 ) {
    printf( "Unable to Add the Tone %d\n", TID_1 );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( chdev ), ATDV_ERRMSGP( chdev ) );
    vpm_close( chdev );
    exit( 1 );
}
/*
 * Build a Tone Generation Template.
 * This template has Frequency1 = 1140,
 * Frequency2 = 1020, amplitude at -10dB for
 * both frequencies and duration of 100 * 10 msecs.
 */
vpm_bldtngen( &tngen, 1140, 1020, -10, -10, 100 );
/*
 * Set up the Terminating Conditions
 */
tpt[0].tp_type = IO_CONT;

```



```

tpt[0].tp_termno = DX_TONE;
tpt[0].tp_length = TID_1;
tpt[0].tp_flags = TF_TONE;
tpt[0].tp_data = DX_TONEON;
tpt[1].tp_type = IO_CONT;
tpt[1].tp_termno = DX_TONE;
tpt[1].tp_length = TID_1;
tpt[1].tp_flags = TF_TONE;
tpt[1].tp_data = DX_TONEOFF;
tpt[2].tp_type = IO_EOT;
tpt[2].tp_termno = DX_MAXTIME;
tpt[2].tp_length = 6000;
tpt[2].tp_flags = TF_MAXTIME;
if (vpm_playtone( chdev, &tngen, tpt, EV_SYNC ) == -1 ){
    printf( "Unable to Play the Tone\n" );
    printf( "Lasterror = %d Err Msg = %s\n",
        ATDV_LASTERR( chdev ), ATDV_ERRMSG( chdev ) );
    vpm_close( chdev );
    exit( 1 );
}
if ( VPMX_TERMMSK( chdev ) & TM_TONE ) {
    printf( "Terminated by Tone Id = %d\n", VPMX_TONEID( chdev ) );
}
/*
* Continue Processing
* .
* .
* .
*/
/*
* Close the opened D/xxx Channel Device
*/
if ( vpm_close( chdev ) != 0 ) {
    perror( "close" );
}
/* Terminate the Program */

```



```
        exit( 0 );  
    }
```

■ Errors

chdev handle , AT_FAILURE .

returns number of bytes

VPMX_TRCOUNT()

Name:	long	VPMX_ TONEID(chdev)
Inputs:	int chdev	SCT channel device handle
Returns:	play,record	:
	AT_FAILURE	:
Includes:	srllib.h	
	smartsdk.h	
Category:	Extended Attribute	

■ **Description**

VPMX_TRCOUNT()

play/record

byte

.

	parameter	.
Parameter	Description	

chdev:	vpm_open()	open	channel
	device handle		



■ **Example**

```
#include <stdio.h>
#include <fcntl.h>
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
main()
{
    int chdev;
    long trcount;
    DX_IOTT iott;
    DV_TPT tpt[2];
    /* Open the channel device */
}
```



```

if ((chdev = vpm_open("vpmB1C1",NULL)) == -1) {
    /* Process error */
}
/* Record a voice file. Terminate on receiving a digit, max time,
 * or reaching a byte count of 50000 bytes.
 */
.
.
/* set up DX_IOTT */
iott.io_type = IO_DEV|IO_EOT;
iott.io_bufp = 0;
iott.io_offset = 0L;
iott.io_length = 50000L;
if((iott.io_fhandle = vpm_fileopen("file.vox", O_RDWR)) == -1) {
    /* process error */
}
/* set up DV_TPTs for the required terminating conditions */
vpm_clrtp(tpt,2);
tpt[0].tp_type = IO_CONT;
tpt[0].tp_termno = DX_MAXDTMF; /* Maximum digits */
tpt[0].tp_length = 1; /* terminate on the first digit */
tpt[0].tp_flags = TF_MAXDTMF; /* Use the default flags */
tpt[1].tp_type = IO_EOT;
tpt[1].tp_termno = DX_MAXTIME; /* Maximum time */
tpt[1].tp_length = 100; /* terminate after 10 secs */
tpt[1].tp_flags = TF_MAXTIME; /* Use the default flags */
/* Now record to the file */
if (vpm_rec(chdev,&iott,tpt,EV_SYNC) == -1) {
    /* process error */
}
/* Examine transfer count */
if((trcount = VPMX_TRCOUNT(chdev)) == AT_FAILURE) {
    /* Process error */
}

printf("%ld bytes recorded\n", trcount);
.

```



```
    .  
}  
}
```

■ Errors

```
chdev      handle      ,      AT_FAILURE      .
```


4. Voice Data Structures and Device Parameters

voice library data structures voice board parameter

Voice Library Data Structures Tables (Section 4.1)
vpm_setparm() and **vpm_getparm()** Parameter
 (Section 4.2)

4.1. Voice Library Data Structures

section Voice Library functions data
 structure

VPM software I/O termination device' s status
 가 structure , structure

<i>DV_DIGIT</i>	User Digit Buffer Structure
<i>DX_CST</i>	Call Status Transition Structure
<i>DX_CAP</i>	Call Analysis Parameter Structure
<i>DX_EBLK</i>	Event Block Structure
<i>DX_IOTT</i>	I/O Transfer Table Structure
<i>DX_SVCB</i>	Speed/Volume Adjustment Condition Block
<i>DX_SVMT</i>	Speed/Volume Modification Block
<i>DX_TPB</i>	Test Parameter Block Structure
<i>DX_UIO</i>	User-definable I/O Structure
<i>DX_XPB</i>	I/O Transfer Parameter Block
<i>TN_GEN</i>	Tone Generation Template structure

NOTE: DV_TPT termination parameter structure Standard Runtime Library
Appendix A

4.1.1. DV_DIGIT - user digit buffer

vpm_getdig() 가 digit VPM_DLL
 digit digit DV_DIGIT structure digit buffer
 array

structure typedef .

```
typedef struct DV_DIGIT {
    char dg_value[DG_MAXDIGS +1]; /* ASCII values of digits */
    char dg_type[DG_MAXDIGS +1]; /* Type of digits */
} DV_DIGIT;
```

	dg_value	digit	ASCII value	NULL-termination	
dg_type	dg_value		digit type	,	NULL-
termination		DG_END	type	.	digit type

DG_DTMF	DTMF digit
DG_LPD	Loop Pulse digit
DG_USER1	User defined tone
DG_USER2	User defined tone
DG_USER3	User defined tone
DG_USER4	User defined tone
DG_USER5	User defined tone

DG_MAXDIGS **vpm_getdig()**

digit 31 . DG_MAXDIGS

dxlib.h .

4.1.2. DX_CAP - change default call analysis parameters

DX_CAP structure Frequency Detection, Cadence Detection, Loop Current,
Positive Voice Detection . structure

vpm_dial() default call Analysis . structure

VPM channel dialing

vpm_clrkap() DX_CAP field clear

NOTE: Call Analysis DX_CAP structure

the *Voice Features Guide for Windows 2000* .

DX_CAP structure	field	clear	vpm_clrca () function
DX_CAP field	0	field	field
reset	.	vpm_dial () function	default value
setting	.		

structure typedef

* DX_CAP

*

* Call Analysis parameters

* [NOTE: All user-accessible structures must be defined so as to be

* unaffected by structure packing.]

*/

typedef struct DX_CAP {

unsigned short ca_nbrdna; /* # of rings before no answer. */

unsigned short ca_stdely; /* Delay after dialing before
analysis. */

unsigned short ca_cnosig; /* Duration of no signal time out
delay. */

unsigned short ca_lcdly; /* Delay after dial before lc drop
connect */

unsigned short ca_lcdly1; /* Delay after lc drop con. before
msg. */

unsigned short ca_hedge; /* Edge of answer to send connect
message. */

unsigned short ca_cnosil; /* Initial continuous noise timeout
delay. */

unsigned short ca_lo1tola; /* % acceptable pos. dev of short low
sig. */

unsigned short ca_lo1tolb; /* % acceptable neg. dev of short low
sig. */

unsigned short ca_lo2tola; /* % acceptable pos. dev of long low
sig. */

unsigned short ca_lo2tolb; /* % acceptable neg. dev of long low

	sig. */
unsigned short ca_hi1tola;	/* % acceptable pos. dev of high signal. */
unsigned short ca_hi1tolb;	/* % acceptable neg. dev of high signal. */
unsigned short ca_lo1bmax;	/* Maximum interval for shrt low for busy. */
unsigned short ca_lo2bmax;	/* Maximum interval for long low for busy. */
unsigned short ca_hi1bmax;	/* Maximum interval for 1st high for busy */
unsigned short ca_nsbuzy;	/* Num. of highs after nbrdna busy check. */
unsigned short ca_logltch;	/* Silence deglitch duration. */
unsigned short ca_higlth;	/* Non-silence deglitch duration. */
unsigned short ca_lo1rmax;	/* Max. short low dur. of double ring. */
unsigned short ca_lo2rmin;	/* Min. long low dur. of double ring. */
unsigned short ca_intflg;	/* Operator intercept mode. */
unsigned short ca_intfltr;	/* Minimum signal to qualify freq. detect. */
unsigned short rfu1;	/* reserved for future use */
unsigned short rfu2;	/* reserved for future use */
unsigned short rfu3;	/* reserved for future use */
unsigned short rfu4;	/* reserved for future use */
unsigned short ca_hisiz;	/* Used to determine which lowmax to use. */
unsigned short ca_alowmax;	/* Max. low before con. if high >hisize. */
unsigned short ca_blowmax;	/* Max. low before con. if high <hisize. */
unsigned short ca_nbrbeg;	/* Number of rings before analysis begins. */
unsigned short ca_hi1ceil;	/* Maximum 2nd high dur. for a retrain. */

unsigned short ca_lo1ceil;	/* Maximum 1st low dur. for a retrain. */
unsigned short ca_lowerfrq;	/* Lower allowable frequency in hz. */
unsigned short ca_upperfrq;	/* Upper allowable frequency in hz. */
unsigned short ca_timefrq;	/* Total duration of good signal required. */
unsigned short ca_rejctfrq;	/* Allowable % of bad signal. */
unsigned short ca_maxansr;	/* Maximum duration of answer. */
unsigned short ca_ansrdgl;	/* Silence deglitching value for answer. */
unsigned short ca_mxtimefrq;	/* max time for 1st freq to remain in bounds */
unsigned short ca_lower2frq;	/* lower bound for second frequency */
unsigned short ca_upper2frq;	/* upper bound for second frequency */
unsigned short ca_time2frq;	/* min time for 2nd freq to remains in bounds */
unsigned short ca_mxtime2frq;	/* max time for 2nd freq to remain in bounds */
unsigned short ca_lower3frq;	/* lower bound for third frequency */
unsigned short ca_upper3frq;	/* upper bound for third frequency */
unsigned short ca_time3frq;	/* min time for 3rd freq to remains in bounds */
unsigned short ca_mxtime3frq;	/* max time for 3rd freq to remain in bounds */
unsigned short ca_dtn_pres;	/* Length of a valid dial tone (def=1sec) */
unsigned short ca_dtn_npres;	/* Max time to wait for dial tone (def=3sec)*/
unsigned short ca_dtn_deboff;	/* The dialtone off debouncer (def=100ms) */
unsigned short ca_pamd_failtime;	/* Wait for AMD/PVD after cadence break(default=4sec)*/
unsigned short ca_pamd_minring;	/* min allowable ring duration (def=1.9sec)*/
byte ca_pamd_spdval;	/* Set to 2 selects quick decision (def=1) */


```

byte ca_pamd_qtemp;          /* The Qualification template to use
                               for PAMD */
unsigned short ca_noanswer;   /* time before no answer after first
                               ring (default=30sec) */
unsigned short ca_maxintering; /* Max inter ring delay before connect
                               (8 sec) */
unsigned short ca_ringhigh;   //(default 200),
unsigned short ca_ringlow1;   //(default 400),
unsigned short ca_ringlow2;   //(default 0)
unsigned short ca_bussyhigh;  //(default 50),
unsigned short ca_bussylow;   //(default 50),
unsigned short ca_basic_dev;  //(default 20 %) parameter .

} DX_CAP;

```

DX_CAP Parameter Descriptions

ca_nbrdna	No Answer	Ring	: <i>no answer</i>
		single	double ring

Length: 1. Default: 4. Units: rings.

ca_stdely	dialing	Cadence Detection, Frequency Detection,
	Positive Voice Detection	analysis

Length: 2. Default: 25. Units: 10 ms.

ca_cnosig	Cadence Detection
	가 <i>no ringback</i> .

Length: 2. Default: 4000. Units: 10 ms.

ca_lcdly

ca_lcdly1

ca_hedge
 ca_cnasil
 ca_lo1tola
 ca_lo1tolb
 ca_lo2tola
 ca_lo2tolb
 ca_hi1tola
 ca_hi1tolb
 ca_lo1bmax
 ca_lo2bmax
 ca_hi1bmax
 ca_nsbusy
 ca_logltch
 ca_higtch
 ca_lo1rmax
 ca_lo2rmin
 ca_intflg

parameter Positive Voice Detection (PVD), Positive Answering
 Machine Detection (PAMD) 가 가
 Frequency Detection mode .

DX_OPTEN: intercept frequency Detection
 가 Cadence Detection
 Loop Current Detection connect
 .

DX_OPTDIS: Frequency Detection PVD 가
 .

DX_OPTNOCON: valid frequency detect *intercept*
 return Frequency Detection 가
 .

DX_PVDENABLE: Enable PVD.

DX_PVDOPTEN: Enable PVD and DX_OPTEN.

DX_PVDOPTNOCON: Enable PVD and DX_OPTNOCON.

DX_PAMDENABLE: Enable PAMD.

DX_PAMDOPTEN: Enable PAMD and DX_OPTEN.

Length: 1. Default: DX_OPTEN.

ca_intfltr

ca_hisiz

ca_alowmax

ca_blowmax

ca_nbrbeg

ca_hi1ceil

ca_lo1ceil

ca_lowerfrq

ca_upperfrq

ca_timefrq

ca_rejctfrq

ca_maxansr

ansrsize time Ansrsiz 가
maxansr application connect .

Length: 2. Default: 1000. Units: 10 ms.

ca_ansrdgl

word silence .
parameter .

Length: 2. Default: -1. Units: 10 ms.

ca_pvdmxper

ca_pvdszwnd

ca_pvddly

ca_mvertimefrq

ca_lower2frq

ca_upper2frq

ca_time2frq

ca_mvertime2frq
ca_lower 3frq
ca_upper3frq
ca_time3frq
ca_mvertime3frq
ca_dtn_pres

Dial Tone

Default: 100. Units: 10 ms.

ca_dtn_npres Dial Tone dial tone

Default: 300. Units: 10 ms.

ca_dtn_deboff
ca_pamd_failtime
ca_pamd_minring
ca_pamd_spdval
ca_noanswer

ringback

call No Answer

Default: 3000. Units: 10 ms.

ca_maxintering call ringback signal
, , connect

Default: 800. Units: 10 ms.

ca_ringhigh ring cadence check ring ontime cadence
default 200 Unit :10ms
ca_ringlow1 ring cadence check ring offtime cadence
default 400 Unit :10ms
ca_ringlow2 ring cadence check dual ring 2 offtime cadence
default 0 Unit :10ms
ca_busyhigh busy cadence check busy tone ontime cadence
default 50 Unit :10ms

ca_busylow	busy cadence check	busy tone	offtime cadence
	default 50 Unit :10ms		
ca_basic_dev;	Basic Call Anaysis	t one	deviation
	(default 20 %)		

4.1.3. DX_CST - call status transition structure

TDX_CST termination TDX_SETHOOK event DX_CST
call status . structure , Event Management
sr_getevtdatap() .

DX_CST typedef .

```
typedef struct DX_CST {
    unsigned short cst_event;
    unsigned short cst_data;
} DX_CST;
```

cst_event event .

DE_DIGITS	received a digit
DE_LCOFF	loop current off event
DE_LCON	loop current on event
DE_LCREV	loop current reversal event
DE_RINGS	rings received event
DE_RNGOFF	caller hang up (incoming call is dropped before being accepted) event
DE_SILOFF	silence off event
DE_SILON	silence on event
DE_TONEOFF	tone off event
DE_TONEON	tone on event
DE_WINK	received a wink
DX_OFFHOOK	offhook event
DX_ONHOOK	onhook event

NOTE: TDX_SETHOOK termination event 가 DX_ONHOOK
DX_OFFHOOK 가 .

cst_data cst_event data .

CST event type CST event data

DE_DIGITS	low byte	ASCII digit ()	high byte	digit type ()
DE_LCOFF	loop on		10 ms units	
DE_LCON	loop off		10 ms units	
DE_RINGS	0			
DE_SILOFF	silence 가	10 ms units		
DE_SILON	nonsilencerk	10 ms units		
DE_TONEOFF	user-specified tone ID			
DE_TONEON	user-specified tone ID			
DE_WINK	N/A			
DX_OFFHOOK	N/A			
DX_ONHOOK	N/A			

4.1.4. DX_EBLK- call status event block structure

structure Call Status Transition event vpm_getevt()
.

NOTE: vpm_getevt() event 가 block
CST event vpm_setevtmsk()
.

structure typedef .

```
typedef struct DX_EBLK {
    unsigned short ev_event; /* Event that occurred */
    unsigned short ev_data; /* Event specific data */
    unsigned char ev_rfu[12]; /* Reserved for future use*/
}DX_EBLK;
```

where:

ev_event	device	event
ev_event		
DE_DIGITS		digit received
DE_LCOFF		loop current off
DE_LCON		loop current on
DE_LCREV		loop current reversal
DE_RINGS		rings received
DE_SILOFF		non-silence detected
DE_SILON		silence detected
DE_TONEOFF		tone off event occurred
DE_TONEON		tone on event occurred
DE_WINK		wink has occurred

ev_data	ev_event	event	data	Table 6
ev_event field		event	data	10
ms units				

Table 6. Values Returned in ev_data

Event	Data Returned in ev_data			
DE_DIGITS	low byte	ASCII digit ()	high byte	digit type ()
DE_LCOFF	loop on		10 ms units	
DE_LCON	loop off		10 ms units	
DE_RINGS	0			
DE_SILOFF	silence 가	10 ms units		
DE_SILON	nonsilencerk	10 ms units		
DE_TONEOFF	user-specified tone ID			
DE_TONEON	user-specified tone ID			
DE_WINK	N/A			

4.1.5. DX_IOTT - I/O transfer table

The DX_IOTT structure play read voice data source
destination . structure vpm_play() vpm_rec()

the structure typedef .

```
typedef struct vpm_iott {
    unsigned short io_type; /* Transfer type */
    unsigned short rfu; /* Reserved */
    int io_fhandle; /* File descriptor */
    char * io_bufp; /* Pointer to base memory */
    unsigned long io_offset; /* File/Buffer offset */
    long int io_length; /* Length of data */
    DX_IOTT *io_nextp; /* Ptr to next DX_IOTT if IO_LINK set */
    DX_IOTT *io_prevp; /* (Optional) Ptr to previous DX_IOTT */
}DX_IOTT;
```

where:

io_type data 가 file memory .
 field DX_IOTT structure 가 linke
 , DX_IOTT .
 define OR making .

data type .

IO_DEV dat 가 file play .
 IO_MEM data 가 memory play .
 structure linkage .

IO_CONT *DX_IOTT* structure .
 IO_LINK DX_IOTT structure link .
 IO_EOT DX_IOTT structure 가 .

IO_CONT, IO_LINK, IO_EOT IO_CONT

IO_DEV 가 io_type io_fhandle file descriptor
 . IO_DEV 가 io_type io_fhandle 0

IO_MEM io_type io_bufp field base memory address

가 .

io_offset 가 .

IO_DEV 가 io_type offset

IO_MEM io_type io_bufp base buffer
address offset .

io_length recording byte playback file
byte 가 . data 가 play -1 .
vpm_play() -1 play , vpm_rec()
가 record .

IO_LINK 가 io_type link list io_nextp

DX_IOTT structure pointer 가 .

io_prevp DX_IOTT structure pointer . vpm_rec() vpm_play()가
field . DX_IOTT structure

io_prevp NULL .

DX_IOTT structure single data file,memory block custom
device . Voice data 가 custom device ,
device standard Windows 2000 device interface 가 . device
open(), close(), read(),write(),lseek()
I/O source destination DX_IOTT
structure . DX_IOTT entry
io_type field IO_EOT 가 .
play source DX_IOTT
structure linked list .

Playback Array Example

```
#include <srllib.h>
#include <smartsdk.h>
#include <windows.h>
DX_IOTT iott[3];
/* first iott: voice data in a file with descriptor fd1*/
```



```

iott[0].io_fhandle = fd1;
iott[0].io_offset = 0;
iott[0].io_length = -1;
iott[0].io_type = IO_DEV;
/* second iott: voice data in a file with descriptor fd2 */
iott[1].io_fhandle = fd2;
iott[1].io_offset = 0;
iott[1].io_length = -1;
iott[1].io_type = IO_DEV;
/* third iott: voice data in a file with descriptor fd3 */
iott[2].io_fhandle = fd3;
iott[2].io_offset = 0;
iott[2].io_length = -1;
iott[2].io_type = IO_DEV|IO_EOT;
.
.
/* play all three voice files: pass &iott[0] as argument to vpm_play( )
.
.

/* form a linked list of iott[0] and iott[2] */
iott[0].io_nextp=&iott[2];
iott[0].io_type|=IO_LINK
/* pass &iott[0] as argument to vpm_play( ). This time only files 1 and 3
* will be played.
*/
.

```

4.1.6. DX_SVMT - speed/volume modification table structure

DX_SVMT structure	level	speed	volume	21	entry
. structure	Speed/Volume Modification Table				

DX_SVMT typedef

```

typedef struct DX_SVMT {
    char decrease[10];    /* Ten Downward Steps */
    char origin;          /* Regular Speed or Volume */
}

```



```

char increase[10];      /* Ten Upward Steps */
} DX_SVMT;

```

Table 7 DX_SVMT entry speed volume

NOTE: DX_SVMT structure21 entry 가 , entry 가
speed volume speed
volume speed volume
entry -128

Table 7. DX_SVMT Entries

Field	Description
decrease[10]	<p>speed volume 10 Array. size table entry -128</p> <p>Valid Values:</p> <p>Speed - 0 original Percentage</p> <p>Volume - 0 original Step (dB)</p>
origin	<p>standard play speed volume speed volumeControl original setting starting point</p> <p>Valid Values:</p> <p>speed volume 0</p>
increase[10]	<p>speed volume 10 Array. size table entry -128</p> <p>Valid Values:</p>

Speed - 0 original 가
Percentage .Valid Value 1 50
Volume - 0 original step(dB) .Valid
Value 1 10

4.1.7. DX_SVCB - speed/volume adjustment condition block

This structure vpm_setsvcond()

Speed/Volume Modification Table
adjustment type (increase/decrease, absolute value, toggle)
adjustment conditions (digit, play)

DX_SVCB structure typedef

```
typedef struct DX_SVCB {
    unsigned short type; /* Bit Mask */
    short adjsize; /* Adjustment Size */
    unsigned char digit; /* ASCII digit value that causes the action */
    unsigned char digtype; /* Digit Type (e.g., 0 = DTMF) */
} DX_SVCB;
```

- NOTES:** 1. DX_SVCB adjustment condition block clear
vpm_clrsvcond() function
2. DX_SVCB adjustment condition block 가
DX_SVCB adjustment condition block reset
remove **vpm_clrsvcond()**
clear

Table DX_SVCB structure entry

Table 8. DX_SVCB Entries

Defines (for type field)	Description (type field)	Description (for adjsize field)
Speed or Volume		

Choose one:		
SV_SPEEDTBL	Modify speed table.	N/A.
SV_VOLUMETBL	Modify volume table.	N/A.
Adjustment Type		
Choose one:		
SV_ABSPOS	Speed Volume Modification Tables original position adjsize field speed volume adjustment position	Volume Modification Tables speed -10 +10 speed .

Volume Example

the DTMF digit "1" DX_SVCB structure one step
volume .

```
svcb[0].type = SV_VOLUMETBL | SV_RELCURPOS;
svcb[0].adjsize = - 1;
svcb[0].digit = ' 1' ;
svcb[0].digtype = DG_DTMF;
```

DTMF digit "2" 가 Speed Modification Table position 5
DX_SVCB structure speed .

```
svcb[0].type = SV_SPEEDTBL | SV_ABSPOS;
svcb[0].adjsize = 5;
svcb[0].digit = ' 2' ;
svcb[0].digtype = DG_DTMF;
```

4.1.8. DX_UIO - user-definable I/O structure

vpm_setuio() structure nonstandard storage devices
accessgkmsrjtdp eogks I/O pointer .


```

/*
 * Structure for user-defined I/O functions
 */
typedef struct DX_UIO {

    int (*u_read) ( );
    int (*u_write) ( );
    int (*u_seek) ( );
} DX_UIO;

```

u_read	field	read	byte	error	-1
read()	function		pointer	.	

u_write	field	write	byte	error	-1
write()	function		pointer	.	

u_seek	field	read	write	start address	offset
lseek()	function		pointer	.	

4.1.9. TN_GEN - tone generation template structure

tone generation template	play	single	dual frequency tone
frequency, amplitude	.		
Structure set up	vpm_bldtngen()	.	

tone play	vpm_playtone()	.
-----------	------------------------	---

TN_GEN data structure .

```

typedef struct {
    unsigned short tg_dflag;          /* Dual Tone - 1, Single Tone - 0 */
    unsigned short tg_freq1;          /* Frequency for Tone 1 (HZ) */
    unsigned short tg_freq2;          /* Frequency for Tone 2 (HZ) */
    short tg_ampl1;                   /* Amplitude for Tone 1 (dB) */
    short tg_ampl2;                   /* Amplitude for Tone 2 (dB) */
    short tg_dur;                     /* Duration of the Generated Tone */
                                     /* Units = 10ms */
}

```



```
} TN_GEN;
```

Table 9 field valid value .

Table 9. TN_GEN Values

TN_GEN Field	Description
tg_dflag	single dual tone . single tg_freq2 tg_ampl2 ignore . Choose one: TN_SINGLE single tone TN_DUAL dual tone
tg_freq1	Frequency in Hz for tone 1 (range 200 to 2000 Hz)
tg_freq2	Frequency in Hz for tone 2; (range 200 to 2000 Hz)
tg_ampl1	Amplitude in dB for tone 1; (range 0 to -40 dB)
tg_ampl2	Amplitude in dB for tone 2; (range 0 to -40 dB)
tg_dur	Duration of the tone in 10 ms units (-1 = infinite duration)

4.1.10. DX_XPB - I/O transfer parameter block

kl/O Transfer Parameter Block (DX_XPB) data structure file format data format,
sampling rate, resolution record play

```
typedef struct {
    USHORT wFileFormat; // file format
    USHORT wDataFormat; // audio data format
    ULONG nSamplesPerSec; // sampling rate
    ULONG nBitsPerSample; // bits per sample
} DX_XPB;
```


vpm_playwav() convenience function WAVE file format
 DX_XPB structure .

DX_XPB Field	Description
wFileFormat	audio file format . field Convenience functions vpm_recwav() , vpm_playwav() , vpm_recvox() , . vpm_playvox() ignore . Choose one: FILE_FORMAT_VOX SCT VOX file format FILE_FORMAT_WAVE Microsoft WAVE file format
wDataFormat	data format . Choose one: DATA_FORMAT ADPCM 4-bit OKI ADPCM (SCT registered) DATA_FORMAT_MULAW 8-bit mu-law PCM DATA_FORMAT_ALAW 8-bit a-law PCM
nSamplesPerSec	digitization rate . Choose one: DRT_8KHZ 8 KHz sampling rate.
nBitsPerSample	sample bit . field ADPCM 8

4.2. Voice Board Parameter Defines for vpm_getparm()

vpm_setparm() file <install drive:>\install directory>\SctCTiSDK \include\SmartSDK.h
 bitmask VPM parameter vpm_getparm() parameter,
 vpm_setparm() parameter .Channels
 boards parameter .board parameters, description Table10
 Channel params Table11 .

Table 10.Board Parameter

Define	Description
VPMBD_FLASHCHR	detect hook flash Flash Character.
VPMBD_FLASHTM	flash onhook .
VPMBD_MAXPDOFF	loop pulse digit 가 off loop .
VPMBD_MINIPD	loop pulse detection loop pulse digit
VPMBD_MINLCOFF	loop drop message 가 .
Define	Description
VPMBD_MINPDOFF	Minimum Loop Current Off - loop pulse detection break .
VPMBD_MINPDON	Minimum Pulse - loop pulse detection make .
VPMBD_MINTION	Minimum DTI On - event ring .
VPMBD_MINTIOFF	Minimum DTI Off - event ring .
VPMBD_OFFHDLY	Offhook Delay - detect DTMF digit 가 event 가 .
VPMBD_PAUSETM	Pause Time- dialing string comma Delay.
VPMBD_P_BK	Pulse Dial Break - Pulse dial offhook .
VPMBD_P_IDD	Pulse Interdigit Delay – pulse dialing digit .
VPMBD_P_MK	Pulse Dial Make – Pulse dial offhook .
VPMBD_R_EDGE	Ring Edge – ring edge

	:
	ET_RON : ring
	ET_ROFF : ring
VPMBD_R_IRD	Inter-ring Delay – ring .call
VPMBD_R_OFF	Ring-Off Interval – “not ring” ring
VPMBD_R_ON	Ring-on Interval – ring ring ring.
VPMBD_S_BNC	Call message 가 silence nonsilence debounce
VPMBD_TTDATA	Dialing DTMF
VPMBD_T_IDD	DTMF Interdigit delay(DTMF dialing digit)
VPMBD_T_MINOFFHKTM	offhook time(10ms)
VPMBD_SILENCE	Silence

Table 11.Channel Parameter

Define	Description
VPMCH_DFLAGS	DTMF detection edge select.
VPMCH_DTINITSET	Play DTMF digit
VPMCH_DTMFTLK	playback DTMF
VPMCH_DTMFDEB	DTMF debounce time delay – DTMF .recording DTMF
VPMCH_PLAYDRATE	Play Digitization Rate – parameter Channel play data digitization .Voice Data 6k 8k sampling rate play
VPMCH_RECRDRATE	Record Digitization Rate – parameter data rate .Voice data 6k 8k sampling rate digit
VPMCH_RINGCNT	ring event 가 return ring
VPMCH_CALLID	vpm_setparm() channel Caller

ID.

5. Voice Programming Conventions

VPM library 가

5.1. Always Check Return Code in Voice Programming

VPM
Voice Library zero

NOTE: I/O

Pointer Extended Attribute ASCIIZ
string "Unknown device" pointer

Pointer Extended Attribute AT_FAILURE

Non-attribute -1

Standard Attribute
ATDV_LASTERR() ATDV_ERRMSGP()
the *Standard Runtime Library Programmer's Guide for Windows 2000*

error 가 EDX_SYSTEM errno check
programming model TDX_ERROR event handler
install

5.2. Clearing Voice Structures

가 library 가 structure clear vpm_clrcap()
DX_CAP structure clear vpm_clrtp() DV_TPT structure clear

structure structure field clear
가

5.3. Using the Voice vpm_playf() and vpm_recf() Convenience Functions

vpm_playf() vpm_recf() Voice
Library . vpm_play() and vpm_rec() .

, vpm_playf() filename single file play
. operation vpm_play() file
가 DX_IOTT structure .
file DX_IOTT structure set up application file
가 vpm_playf() file play
.

vpm_recf() vpm_rec() single file convenience .

5.4. Using the Voice Asynchronous Programming Model

process multiple thread
I/O function operate
programming model the *Standard Runtime Library*
Programmer's Guide for Windows 2000 .

5.5. Using Multiple Processes in Voice Synchronous Applications

application processe
master control process , channel child
process child process .

channel open close .
channel parameter .
channel operation .
channel event monitoring .

NOTE: parent process child process application device
handle child process . Device child process open
.

Appendix A

Standard Runtime Library

Voice Device Entries and Returns

Standard Runtime Library Event Management function Standard Attribute
function DV_TPT Termination Parameter Table device
independent library . VPM SRL function data structure the *Standard Runtime
Library Programmer's Guide for Windows 2000* .

appendix Voice board entry , Standard Runtime Library (SRL)
.

Event Management Functions

The Event Management function Voice device
termination event , .
 vpm_dial()
 vpm_getdig()
 vpm_play()
 vpm_rec()
 vpm_playtone()
 vpm_sethook()
 vpm_wink()
 r2_playbsig()

Voice board 가 Event Management function
table list . Table 12 Table 13 event management
list .

Table 12. Voice Device Inputs for Event Management Functions

Event Management Function	Voice Device Input	Valid Value
sr_enbhdr() <i>Enable event handler</i>	evt_type	TDX_PLAY TDX_PLAYTONE TDX_RECORD TDX_GETDIG TDX_DIAL TDX_CALLP TDX_CST TDX_SETHOOK TDX_ERROR
sr_dishdr() <i>Disable event handler</i>	evt_type	

Table 13. Voice Device Returns from Event Management Functions

Event Management Function	Return Description	Returned Value
sr_getevtdv() Get SCT Device <i>handle</i>	device	Voice device handle
sr_getevttype() <i>Get event type</i>	event type	TDX_PLAY TDX_PLAYTONE TDX_RECORD TDX_GETDIG TDX_DIAL TDX_CALLP TDX_CST TDX_SETHOOK
sr_getevtlen()	event length	sizeof (DX_CST)

Get event data length

event data pointer event data DX_CST structure
pointer

Standard Attribute Functions

Standard Attribute function device name device error
general SCT device information . Standard
Attribute function Voice device information Table 14 list .
.

Table 14. Standard Attribute Functions

Standard Attribute Function	Information Returned for Voice Devices			
ATDV_ERRMSGP()	device			
		error		string
	Pointer (가	error	list
	Appendix B			가
	error			.
ATDV_LASTERR()	device			
		error .		가
	error			.
ATDV_NAMEP()	device name	Pointer (e.g., vpmBbCc).		
	device name			the
		System Release Software Installation Reference for Windows 2000		.
ATDV_SUBDEVS()	sub-devices	(sub-devices		
	the Standard Runtime Library Programmer's Guide for Windows 2000			.)

DV_TPT Structure

DV_TPT termination parameter table VPM device
VPM board DV_TPT valid section

DV_TPT structure I/O function termination condition
 structure I/O function

vpm_clrtp()
vpm_getdig()
vpm_play()
vpm_rec()
vpm_playtone()

DV_TPT structure 가 I/O function
 , I/O function
 . DV_TPT structure 가 link list array

structure format 가 .

```
typedef struct DV_TPT {  
    unsigned short tp_type;                      /* Flags describing this entry */  
    unsigned short tp_termno;                   /* Termination Parameter number */  
    unsigned short tp_length;                   /* Length of terminator */  
    unsigned short tp_flags;                   /* Parameter attribute flag */  
    unsigned short tp_data;                   /* Optional additional data */  
    unsigned short rfu;                   /* Reserved */  
    DV_TPT *tp_nextp;                   /* Pointer to next termination  
                                 * parameter if IO_LINK set  
                                 */  
}DV_TPT;
```

field section . field Table 15
field

tp_type

tp_type structure 가 link list , array , DV_TPT table DV_TPT entry . tp_type

IO_LINK tp_nextp 가 DV_TPT structure pointer 가 . IO_EOT DV_TPT . IO_CONT DV_TPT structure 가 .

tp_termno

tp_termno . Voice device .

- DX_MAXDTMF Maximum number of digits received
- DX_MAXSIL Maximum length of silence
- DX_MAXNOSIL Maximum length of non-silence
- DX_IDDTIME Maximum delay between digits
- DX_MAXTIME Maximum function time
- DX_DIGMASK Specific digit received
- DX_TONE Tone On/Off termination

I/O termination the Voice Features Guide for Windows 2000 .

Extended Attribute function . bitmap . bitmap "TM_" VPMX_TERMMSK() .

tp_length

tp_length size . tp_length 가 6000 . field .

Table 15. tp_length Settings

tp_length value	tp_length description				
time in 10 or 100ms units	6000	time	,		
# of digits	digit 가 가				DX_MAXDTMF
digit bit mask	bit mask			DX_DIGMASK	
	Digit	Define			
	Digit	Digit	Digit	Digit	Digit
		Define	Define	Define	Define
	0	DM_0			
	1	DM_1	6	DM_6	# DM_P
	2	DM_2	7	DM_7	a DM_A
	3	DM_3	8	DM_8	b DM_B
	4	DM_4	9	DM_9	c DM_C
	5	DM_5	*	DM_S	d DM_D
tp_flags					
tp_flag				bit mask	
termination flag					
TF_EDGE	termination condition	edge-sensitive			
TF_LEVEL	termination condition	level-sensitive			
TF_CLREND	history	function	clear		
TF_CLRBEG	history	function	clear		
TF_USE		terminator			
TF_SETINIT	DX_MAXSIL only - initial length of silence to terminate on				
TF_10MS	10 ms	time	unit		(default is 100

ms)

TF_FIRST DX_IDDTIME only - start looking for termination condition (interdigit delay) to be satisfied after first digit is received

default tp_flags . default

TF_MAXDTMF (TF_LEVEL|TF_USE)
TF_MAXSIL (TF_EDGE|TF_USE)
TF_MAXNOSIL (TF_EDGE|TF_USE)
TF_IDDTIME (TF_EDGE)
TF_MAXTIME (TF_EDGE)
TF_DIGMASK (TF_LEVEL)
 (TF_EDGE)
TF_TONE (TF_LEVEL|TF_USE|TF_CLREND)

NOTES:

DX_IDDTIME DX_MAXTIME TF_IDDTIME tp_termno
TF_IDDTIME TF_MAXTIME . flag OR
combination .

tp_flags field bitmap .

rfu	rfu	Units	ini	use	beg	end	level
7		bits			0		

bit list .

bit 0 (level): level-sensitive .

Level-sensitive 가 ,
가 .

가
terminator record history .

bit 가 edge-sensitive ,

table

edge-sensitive level-sensitive

terminationsedge-sensitive .

NOTE: level-sensitive termination condition

terminator history

Term. Condition	Level-sensitive	Edge-sensitive
DX_DIGTYPE	X	X
DX_MAXDTMF	X	X
DX_MAXSIL	X	X
DX_MAXNOSIL	X	X
DX_LCOFF	X	X
DX_DIGMASK	X	X
DX_IDDTIME	-	X
DX_MAXTIME	-	X

bit 1 (end): , terminator history

clear . bit DX_IDDTIME

digit

가 terminator .

bit 2 (beg): terminator history

clear . bit level bit override

history clear

terminator history

bit 3(use): bit 가 terminator

bit 가

terminator history clear .

terminator .

bit .

DX_MAXTIME

DX_IDDTIME

DX_DIGMASK

DX_PMOFF

DX_PMON

bit 4 (ini): bit DX_MAXSIL termination .

가 edge-sensitive bit 가

tp_data field silence 가 non-silence detect

silence

tp_data silence

tp_length .

가 level sensitive bit 0

tp_length .

bit 5 (units): time unit 10 ms 가 . Default

100ms .

NOTES: 10 ms timer resolution 0.62 version

가 D/4x firmware 가

.

tp_data

tp_data optional additional data . bit .

Table 16. tp_data Valid Values

Value in tp_termno	tp_data Entry
DX_MAXSIL	silence

tp_nexttp

tp_nexttp linked list DV_TPT structure pointer . tp_type
field IO_LINK .

table DV_TPT field 가 .

NOTE: * tp_flag default setting . default setting override
 tp_flag bit .

Table 17. DV_TPT Fields Settings Summary

NOTE: tp_flags column field
 . * bit default 가 . tp_flags field
default Appendix tp_flags list .

Appendix B

Error Defines

Errors - Voice Library

appendix Voice Library function error

.

SCbus function error code the error the

SCbus Routing Function Reference for Windows 2000 .

table ATDV_LASTERR() ATDV_ERRMSGP() function

error list .

Table 18. Voice Library Function Errors

Error Define	Error String	
EDX_AMPLGEN	Tone Generation Template	Invalid Amplitude Value
EDX_ASCII	Tone Template Description	Invalid ASCII Value
EDX_BADDEV	Device Descriptor error	
EDX_BADIOTT	DX_IOTT structure error	
EDX_BADPARM	Parameter error	
EDX_BADPROD	Board	Function
EDX_BADTPT	DX_TPT structure error	
EDX_BUSY	Device or channel Busy	.
EDX_CADENCE	Tone Template Description	Invalid Cadence Com- ponent Values
EDX_CHANNUM	Invalid Channel Number	
EDX_DIGTYPE	Tone Template Description	Invalid Dig_type Value
EDX_FLAGGEN	Tone Generation Template	Invalid tn_dflag field
EDX_FREQDET	Tone Template Description	Invalid Frequency Component Values
EDX_FREQGEN	Tone Generation Template	Invalid Frequency Component
EDX_FWERROR	Firmware Error	

EDX_IDLE	Device is Idle
EDX_INVSUBCMD	Invalid sub-command number
EDX_MAXTMPLT	Tone Templates Exist Max number the board user-defined tones[from r2_creatfsig()]
EDX_MSGSTATUS	Invalid Message Status Setting
EDX_NOERROR	No Error
EDX_NONZEROSIZE	Reset to Default was Requested but size was non-zero
EDX_SPDVOL	Must Specify either SV_SPEEDTBL or SV_VOLUMETBL
EDX_SVADJBLKS	Speed/Volume Adjustment Blocks Invalid Number
EDX_SVMTRANGE	An Entry in SV_SVMT was out of Range
EDX_SVMTSIZE	Invalid Table Size Specified
EDX_SYSTEM	Windows 2000 System Error; check the general variable errno for more information
EDX_TIMEOUT	I/O Function Timed Out
EDX_TONEID	Invalid Tone Template ID

Appendix C

DTMF

DTMF tone tone .

DTMF Tone Specifications

Code	Tone Pair Frequencies (Hz)	Default Length (ms)
1	697,1209	100
2	697,1336	100
3	697,1447	100
4	770,1209	100
5	770,1336	100
6	770,1447	100
7	852,1209	100
8	852,1336	100
9	852,1447	100
0	941,1336	100
*	941,1209	100
#	941,1447	100
a	697,1633	100
b	770,1633	100
c	852,1633	100
d	941,1633	100

Appendix D

Related Voice Publications

VPM hardware software product
VPM publication .

installing Voice software the *System Release Software Installation
Reference for Windows 2000* .

Standard Runtime Library the *Standard Runtime Library
Programmer's Guide for Windows 2000* .

Scbus the *SCbus Routing Guide* and the *SCbus Routing Function
Reference for Windows 2000* .