

DPM Software Programmer' s Guide for Windows 2000

Copyright © 1999 - 2000 Seoul Commtech Corporation

Table of Contents

1 . DPM SOFTWARE REFERENCE OVERVIEW	5
1. 1 VOICE PRODUCT TERMINOLOGY	5
1.2 ORGANIZATION OF THIS DPM REFERENCE GUIDE	6
1.3 DPM DRIVER	7
1.4 DPM LIBRARY	8
1.4.1 SINGLE THREADED ASYNCHRONOUS PROGRAMMING MODEL	8
1.4.2 MULTITHREADED SYNCHRONOUS PROGRAMMING MODEL	9
1.4.3 EXTENDED ASYNCHRONOUS PROGRAMMING MODEL	9
2. USING THE DPM REFERENCE LIBRARY.....	10
2.1 DPM LIBRARY	10
2.2 DPM PROGRAMMING REQUIREMENTS.....	11
2.2.1 OPENING AND USING DEVICES	11
2.2.2 OPENING AND USING VOICE FILES.....	12
2.2.3 BUSY AND IDLE STATES	12
2.2.4 I/O TERMINATION.....	13
2.2.5 ERROR HANDLING.....	13
2.2.6 DPM LIBRARY INCLUDE FILES	13
2.2.7 COMPILING APPLICATION.....	13
3. DPM FUNCTION REFERENCE.....	14
3.1 DPM FUNCTION REFERENCE OVERVIEW	14
3.2 R2MF OVERVIEW	16
3.2.1 R2MF SIGNALING	16
3.2.2 R2MF FLOW.....	18
3.2.2.1 R2MF FLOW (WITHOUT ANI).....	19
3.2.2.2 R2MF FLOW (WITH ANI).....	20
3.3 ROUTING.....	21
3.4 E1 / R2 TRACE	21
3.5 DPM NETWORK LIBRARY FUNCTION DESCRIPTION.....	24
<i>dpm_sendvt() sends the selected events.....</i>	<i>24</i>
<i>DPMT_BDSGBIT() returns the current state of the transmit and receive.....</i>	<i>28</i>
<i>DPMT_DNLDVER() returns the firmware version.....</i>	<i>31</i>
<i>DPMT_TSSGBIT() retrieves the current state</i>	<i>33</i>
<i>dpm_getparm() gets the current value.....</i>	<i>35</i>

<i>dpm_getevt()</i>	<i>blocks and returns control to the program.....</i>	<i>38</i>
<i>dpm_getevtmask()</i>	<i>retrieves the current event bitmask(s).....</i>	<i>41</i>
<i>dpm_setparm()</i>	<i>changes the value of a device parameter.....</i>	<i>43</i>
<i>dpm_setevtmask()</i>	<i>enables and disables notification for events.....</i>	<i>45</i>
<i>dpm_open()</i>	<i>opens a Digital Network Interface device.....</i>	<i>48</i>
<i>dpm_getxmitslot()</i>	<i>returns the SCbus time slot.....</i>	<i>52</i>
<i>dpm_listen()</i>	<i>connects the digital listen channel.....</i>	<i>54</i>
<i>dpm_unlisten()</i>	<i>disconnects the receive.....</i>	<i>56</i>
<i>dpm_dial()</i>	<i>allows the application to pulse dial.....</i>	<i>58</i>
<i>dpm_seizure()</i>	<i>seizure a E1 line.....</i>	<i>66</i>
<i>dpm_answer()</i>	<i>equivalent to conventional “set hook off” function.....</i>	<i>68</i>
<i>dpm_playR2()</i>	<i>send R2 signal.....</i>	<i>71</i>
<i>dpm_release()</i>	<i>Release a E1 channel.....</i>	<i>75</i>
<i>dpm_block()</i>	<i>block a E1 channel.....</i>	<i>78</i>
4. LOGICAL CALL FUNCTION REFERENCE.....		81
4.1 LOGICAL CALL REFERENCE OVERVIEW		81
4.3 LOGICAL CALL STATES.....		81
4.3.1 BASIC CALL MODEL –		82
4.3.1 BASIC CALL MODEL –		88
4.4 ROUTING.....		93
4.5 EVENT HANDLING.....		94
4.6 EVENT DEFINITIONS.....		94
4.7 ERROR HANDLING.....		97
4.8 LOGICAL CALL LIBRARY FUNCTION DESCRIPTION.....		97
<i>lc_AnswerCall()</i>	<i>equivalent to conventional “set hook off” function.....</i>	<i>98</i>
<i>lc_DropCall()</i>	<i>disconnects a call.....</i>	<i>101</i>
<i>lc_MakeCall()</i>	<i>enables the application to make an outgoing call.....</i>	<i>104</i>
<i>lc_ReleaseCall()</i>	<i>releases all internal resources.....</i>	<i>109</i>
<i>lc_WaitCall()</i>	<i>sets up conditions for processing inbound calls.....</i>	<i>111</i>
<i>lc_AcceptCall()</i>	<i>optional response to an inbound call.....</i>	<i>114</i>
<i>lc_GetANI()</i>	<i>returns ANI information.....</i>	<i>117</i>
<i>lc_GetDNIS()</i>	<i>gets the DNIS information.....</i>	<i>119</i>
<i>lc_GetVer()</i>	<i>gets the version number.....</i>	<i>122</i>
<i>lc_SetBilling()</i>	<i>sets billing information for the call.....</i>	<i>124</i>
<i>lc_SetCallingNum()</i>	<i>sets the default calling party number.....</i>	<i>127</i>
<i>lc_SetChanState()</i>	<i>changes the maintenance state.....</i>	<i>129</i>

<i>lc_Close()</i>	<i>closes a previously opened device.....</i>	<i>131</i>
<i>lc_CRN2LineDev()</i>	<i>matches a CRN to its line device ID.....</i>	<i>133</i>
<i>lc_ErrorValue()</i>	<i>gets an error value/failure reason code.....</i>	<i>135</i>
<i>lc_GetCallState()</i>	<i>acquires the state of the call.....</i>	<i>137</i>
<i>lc_GetCRN()</i>	<i>gets the CRN.....</i>	<i>140</i>
<i>lc_GetLineDev()</i>	<i>gets a line device.....</i>	<i>142</i>
<i>lc_GetMetaEvent()</i>	<i>maps the current SRL event into a metaevent.....</i>	<i>145</i>
<i>lc_GetMetaEventEx()</i>	<i>maps an event handle's SRL event into a metaevent.....</i>	<i>152</i>
<i>lc_GetNetworkH()</i>	<i>returns the network device handle</i>	<i>154</i>
<i>lc_GetUsrAttr()</i>	<i>retrieves the attribute.....</i>	<i>156</i>
<i>lc_GetXmitSlot()</i>	<i>returns the network SCbus time slot number.....</i>	<i>159</i>
<i>lc_Listen()</i>	<i>connects a channel to a network SCbus time slot.....</i>	<i>162</i>
<i>lc_Open()</i>	<i>opens a LogicalCall device</i>	<i>165</i>
<i>lc_ResetLineDev()</i>	<i>disconnects any active calls.....</i>	<i>170</i>
<i>lc_ResultMsg()</i>	<i>retrieves an ASCII string describing a result code</i>	<i>173</i>
<i>lc_ResultValue()</i>	<i>retrieves the cause of an event.....</i>	<i>175</i>
<i>lc_SetEvtMsk()</i>	<i>sets the event mask.....</i>	<i>177</i>
<i>lc_SetUsrAttr()</i>	<i>sets an attribute defined by the user.....</i>	<i>180</i>
<i>lc_UnListen()</i>	<i>disconnects a receive channel from the network SCbus time slot.....</i>	<i>182</i>
<i>lc_Attach()</i>	<i>attaches a voice resource</i>	<i>184</i>
<i>lc_Detach()</i>	<i>logically detach a voice resource</i>	<i>187</i>
<i>lc_GetVoiceH()</i>	<i>returns the voice device handle</i>	<i>190</i>
<i>lc_GetCallInfo()</i>	<i>gets information for the call.....</i>	<i>192</i>
<i>lc_StartTrace()</i>	<i>starts logging debug information.....</i>	<i>195</i>
<i>lc_StopTrace()</i>	<i>stops the trace.....</i>	<i>197</i>
5. DPM DATA STRUCTURE AND DEVICE PARAMETERS.....		199
APPENDIX A.....		203
DPM DEVICE ENTRIES AND RETURNS.....		203
EVENT MANAGEMENT FUNCTIONS.....		203
STANDARD ATTRIBUTE FUNCTIONS.....		206
APPENDIX B		207
PUBLICATIONS.....		207

1 . DPM Software Reference Overview

1. 1 Voice Product Terminology

Product Naming	Guide	
CT-V04A : 4 가 Slot 가 ,	1	가 .
CT-V08A : 8 가 Slot 가 ,	2	가 .
CT-V16A : 16 가 Slot 가 ,	2	가 .
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 DPM Slot Dot Board	4 FAX .	,
CT-V30A : 30 가 E-1 signal	가	.
Firmware Load File : Voice Board	download	firmware file
SCbus : SCSA(Singnal Computing System Architecture) voice TDM Bus		Board resource .

1.2 Organization of This DPM Reference Guide

DPM Software Programmer's Guide For Windows 2000 Windows 2000
DPM Logical Call Software Interface Driver Library
instruction .

Chapter 1. DPM Software Reference Overview DPM Logical Call Software
Overview . Component 가 SCT
Board , DPM Driver Library .

Chapter 2. Using the DPM Reference Library DPM Library LibDPM.lib Logical
Call Library LibLC.lib . function
category Overview Library , Programming

Chapter 3. DPM Function Reference dpm Library Reference

Chapter 4. Logical Call Function Reference Logical Call Library
Reference .

Chapter 5. Data Structure and Device Parameters .

- dpm Library Logical Call library data structure table

Appendix A DPM Device Entry , DPM Device Standard
Runtime Library return Definition .

Appendix B DPM Library error .

Appendix C SCT Publication List .

DPM Software

- DPM Driver
- DPMLibrary of C functions
- Standard Runtime Library

DPM Driver DPM Library

DPM Software

- DPM 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

1.3 DPM Driver

DPM Driver DPM hardware , DPM hardware

DPM Hardware CT-V30A

CT-V30A DPM Processing feature

- Record Play
- Play Speed Volume
- Call
- Call Analysis
- R2MF
- DTMF
- Pulse Dial
- Global Tone Generation Detection

Record Play, Play Speeddhk Volume , DTMF
voice programmer' s Guide for Windows 2000

Global Tone Detection Voice Feature Guide

DPM 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 DPM Library

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

- LibDPM.lib – Main DPM Library
- LibLC.lib – Logical Call Library
- LibSrl.lib – Standard Runtime Library
 “C” function library .
- DPM Board .
- Single Thread Synchronous Multithread Asynchronous Programming Model
 Application .
- Device .
- Device event
- Device information

LibDPM.lib LibLC.lib 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
 DPM channel . task가
 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 block
channel thread가
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 DPM Reference Library

DPM library	Programming	description
• DPM library	category (section 2.1)	
• DPM Library	Programming	(section 2.2)
2.1 DPM Library		
DPM library	DPM device driver	interface
category		
• Alarm Functions	E-1 Alarm handling	control
• Extended Attribute Functions		가 data
• Parameter Request Functions	device parameter	
• Parameter Setting Functions	device parameter	set
• Resource Management Functions	device	open close
• Scbus Routing Functions	device	SCbus timeslot
• Time Slot Signaling Functions	Timeslot	signaling
• Basic Functions	signaling system	
• Optional Call Handling and Features Functions		가 call
	handling billing()	number ID
• System Controls and Tools Functions	call,parameter,	call control
	library management capability	
• CAS Interface Specific Functions	CAS interface	signaling system
• ISDN Interface Specific Functions	ISDNinterface	signaling system
	Category	category
DPM Function Reference	,	header
가 category가		

2.2 DPM Programming Requirements

DPM library library

- Device open (section 2.2.1)
- voice file 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)
- DPM 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 = vpm_fileopen(filename, mode);
```

file action file descriptor ,
open action

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

```
int chdev;
chdev = dpm_open(NetworkDeviceName, NULL);
```

dpm_open() Network . Record Play
voice device open , routing 가 .
Routing voice device VPM dpm_open()

```
nr_scroute(chdev, SC_DPM, chdev, SC_DSP, SC_FULLDUP);
```


LogicalCall dpm_open() lc_Open()
 Network device name voice device name
 routing , 가 line device , network,
 voice . voice device voice .

LINEDEV ldev;

lc_Open(&ldev, NetworkDeviceName, VoiceDeviceName, usrattr)

 channel DPM library function , channel
 SCT channel device handle chdev .
 channel name channel open open action
 handle chdev ldev .
 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 dpm_open() dpm_close() .

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

2.2.2 Opening and Using Voice Files

voice programmer's Guide for Windows 2000 Opening and Using Voice Files
 section .

2.2.3 Busy and Idle States

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

state dependency .

2.2.4 I/O Termination

voice programmer's Guide for Windows 2000 I/O Termination .

2.2.5 Error Handling

SCT voice library .

0 .

2.2.6 DPM Library Include Files

DPM library application code 가 .

#include <smartsdk.h>

Error Handling application code 가 .

#include <gcerr.h>

Library header file install .

<install drive:>\<install directory> \ inc

2.2.7 Compiling Application

Windows 2000 voice library application

library가 link .

Multi thread library default directory .

<install drive:>\<install directory> \ lib

library link .

DPM board	DPM Function
DPM Function	Alarm Functions, Diagnostic Functions, Extended Attribute Functions, Parameter Request Functions, Resource Management Functions, Scbus Routing Functions, Time Slot Signaling Functions, Basic Functions, Optional Call Handlings and Features Functions, System Controls and Tools Functions, CAS Interface Specific Functions, ISDN Interface Specific Functions

```

DPM( Digital Processing Module) Application
initiating dpm_xxxx() , lc_xxxx()
call setup R2 signaling
application layer ,
, R2 signaling Application control dpm_xxxx()
, lc_xxxx()
application DPM configuration file( dpm.cfg ) service
program start Application dpm_xxxx()
0 lc_xxxx() 1

```

- `lc_xxxx()`
:
parameter 44:

lc_ or dpm_ use flag
\$44 lc(logical call library) use flag (0=not use, 1=use) : 1
:
- `dpm_xxxx()`
:
parameter 44:

lc_ or dpm_ use flag
\$44 lc(logical call library) use flag (0=not use, 1=use) : 0


```
SDK
copy .
<install drive:>\<install directory> \SctCTiSDK \dpm.cfg
```


3.2 R2MF Overview

R2MF signaling is used for signaling between the central office (CO) and customer premises equipment (CPE) for lines. It is used for international signaling. Call set-up, R2MF signal, interregister signal, and Central office (CO) customer premises equipment (CPE) signals are used. CO가 outgoing register, CPE가 incoming register. Outgoing register forward interregister signal, backward interregister signal. incoming register forward interregister signal, backward interregister signal.

3.2.1 R2MF Signaling

R2MF signaling is used for forward, backward group 1 ~ 15 digit. outgoing register forward group (Group I, II), incoming register backward group (Group A, B). Table.1

Table 1. Forward Signal (Group I , II)

NO	Group I		Group II	
1	I-1	Digit 1	II-1	Subscriber Without Priority
2	I-2	Digit 2	II-2	Subscriber With Priority
3	I-3	Digit 3	II-3	MAINTENANCE
4	I-4	Digit 4	II-4	PAY STATION(COIN BOX)
5	I-5	Digit 5	II-5	OPERATOR
6	I-6	Digit 6	II-6	DATA TRANSMISSION
7	I-7	Digit 7	II-7	Reserved
8	I-8	Digit 8	II-8	Reserved
9	I-9	Digit 9	II-9	Reserved
10	I-10	Digit 0	II-10	Reserved
11	I-11	NOT USED	II-11	Reserved
12	I-12	REQUEST NOT ACCEPTED	II-12	Reserved
13	I-13	ACCESS TO TEST EQUIPMENT	II-13	Reserved
14	I-14	Reserved	II-14	Reserved
15	I-15	END OF PULSING	II-15	Reserved

Table 2. Backward Signal (Graoup A , B)

NO	Group A		Group B	
1	A-1	Request Next (N + 1) digit	B-1	Subscriber Line Free. Last Party
2	A-2	Request Previous (N - 1) digit	B-2	Changed Number
3	A-3	Address Complete. Change Group B	B-3	Subscriber Line Busy
4	A-4	congest	B-4	congest
5	A-5	Request Calling Number	B-5	Unallocated Number
6	A-6	Address Complete. Call Setup	B-6	Subscriber Line Free. Charge
7	A-7	Request N-2 Digit	B-7	Subscriber Line Free. No Charge
8	A-8	Request N-3 Digit	B-8	Subscriber Line Out Of Order
9	A-9	Request First Digit	B-9	Reserved
10	A-10	Reserved	B-10	Reserved
11	A-11	Reserved	B-11	Reserved
12	A-12	Reserved	B-12	Reserved
13	A-13	Reserved	B-13	Reserved
14	A-14	Reserved	B-14	Reserved
15	A-15	Reserved	B-15	Reserved

calling party called party A-3 A-5 , signal
 acknowledge signal Group II signal . , called party
 calling party Group II signal , acknowledge signal
 Group B signal . , called party A-5
 calling party category digit ANI digits ,
 called party Group II signal , ANI digits
 Group I digit .

Table. 3 3.2.2 section .

Table 3. Signal Group

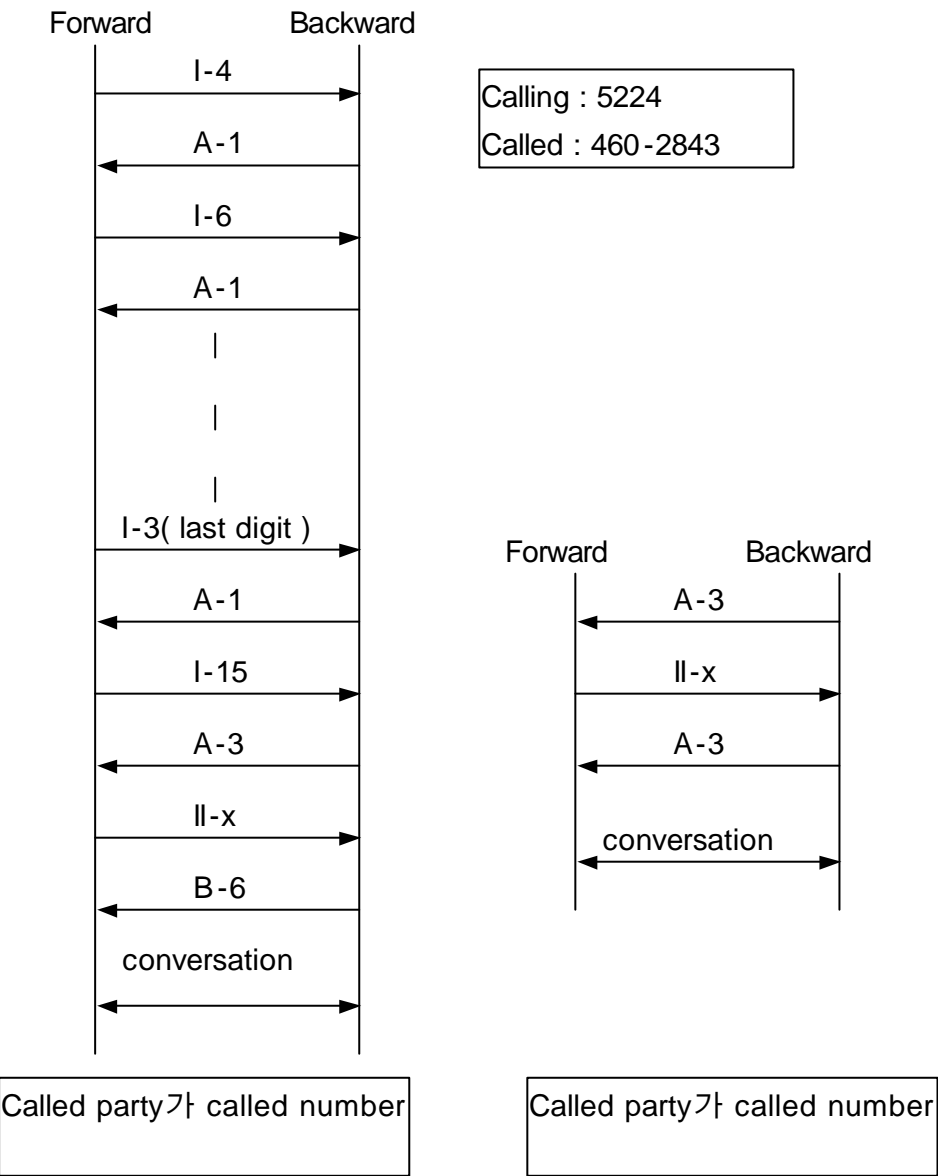
Signal	
Group I	Group I signal call set-up , address .
Group A	Group A signal call set-up address

	Group I signal acknowledge signals	
	A-3	Address Complete signal Group A B change
	A-5	Calling party's category Group A B change
Group II	Group II signal A-3 A-5 signals acknowledge signal calling party category	
Group B	Group B signal Group II signal acknowledge , signal subscriber Group B signal , called party A-3 signal	

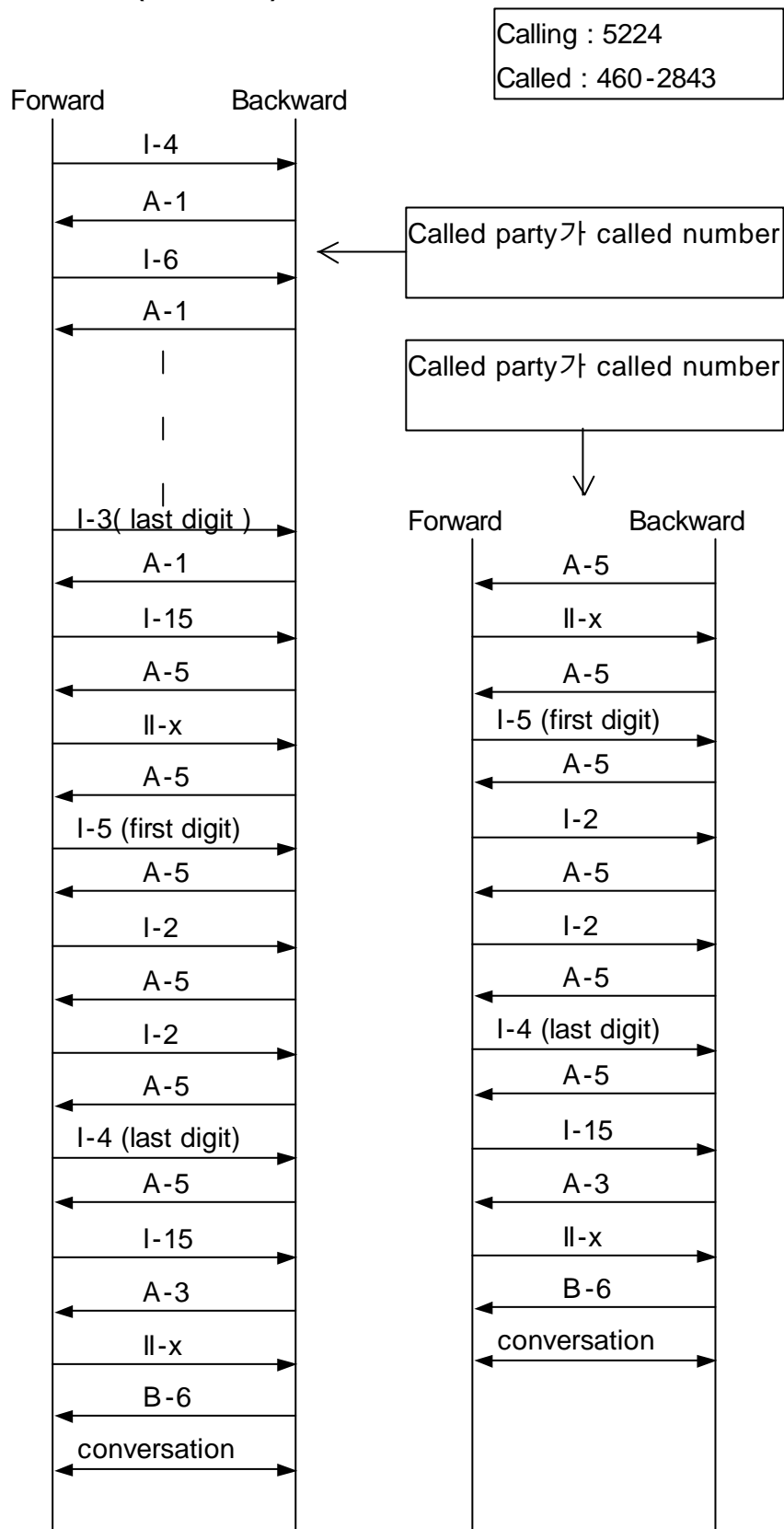
3.2.2 R2MF Flow

R2MF flow Without ANI With ANI 가 , called
party가 called number . 3.2.2.1
section 3.2.2.2 section .

3.2.2.1 R2MF Flow (Without ANI)



3.2.2.2 R2MF Flow (With ANI)



3.3 Routing

SCbus SCSA(signal Computing System Architecure) voice telephone resource TDM(Rime Division Multiplexed) bus .

SCbus voice , analog , digital voice telephone network .

Data 1024 time slots . , (analog , voice, digital network ...)

SCbus time slots .

Digital Network Interface Device time slot SCbus time slot 가 . transmit SCbus time slot . transmit application .

SCbus SCbus Routing Function Reference for Windows 2000 .

3.4 E1 / R2 Trace

Logical Call E1 signaling bit, Alarm R2MF signaling monitoring .

application application

data .

R2MF signaling Trace R2EV_R2TRACE , signaling bit

Trace R2EV_E1TRACE, E1 Alarm Trace

R2EV_E1ERRC . sr_getevtttype() , data

sr_getevtdatap() .

device name open .

dpm_setparm () ,

DPMCH_R2_TRACE value 1 .

- R2EV_R2TRACE
- E1 call setup R2MF signaling
- sr_getevtdatap() .
- DX_CST Direction, Group Number, Digit

, DX_CST cst_event Direction (Tx
0, Rx 1) , cst_data 8bit Group number,
8bit Digit .

● R2EV_E1TRACE

time slot signaling bit(Tx, Rx) time slot
signaling bit(Tx, Rx)
sr_getevtdatap() DX_CST
cst_event Duration ,
cst_data 8bit time slot signaling bit(Tx, Rx) ,
8bit time slot signaling bit(Tx, Rx)
Tx, Rx DPMT_TSSGBIT () .

● R2EV_E1ERRC

Alarm Application . E-1 trunk
가
sr_getevtdatap() DX_CST cst_event
cst_event 가 가 Alarm code .

Table 4. Alarm Code

Alarm code	Description
DPME1_LOS	Loss of Signal On
DPME1_AIS	AIS(Alarm Indication Signal) On
DPME1_RYEL	Yello Alarm(Remote Side가 Multi Frame Sync. Error) On
DPME1_ALM	ALM(Remote Side가 Frame Sync. Error) On
DPME1_FSERR	Frame Sync.(Frame Sync.) ON
DPME1_MFSERR	Multi Frame Sync.(Multi Frame Sync.) On
DPME1_SLIP	Slip On
DPME1_CLKLOS	Clock Lost On
DPME1_LOSOFF	Loss of Signal recovered
DPME1_AISOFF	AIS(Alarm Indication Signal) recovered
DPME1_RYELOFF	Yello Alarm(Remote Side가 Multi Frame Sync. Error) recovered
DPME1_ALMOFF	ALM(Remote Side가 Frame Sync. Error) recovered

DPME1_FSERROFF	Frame Sync.(Frame Sync.) recovered
DPME1_MFSERROFF	Multi Frame Sync.(Multi Frame Sync.) recovered
DPME1_SLIPOFF	Slip recovered
DPME1_CLKLOSOFF	Clock Lost recovered

cst_event 가 가 Error code maskable . ,
enable disable dpm_setevtmask()
R2EV_E1ERRC , masking value
. masking value ' | ' (or) 가 .

Table 5. Alarm Masking Value

Masking value	Description
DPMEC_LOS	Loss of Signal
DPMEC_AIS	AIS(Alarm Indication Signal)
DPMEC_RYEL	Yello Alarm
DPMEC_ALM	ALM
DPMEC_FSERR	Frame Sync
DPMEC_MFSERR	Multi Frame Sync
DPMEC_SLIP	Slip
DPMEC_CLKLOS	Clock Lost

3.5 DPM Network Library Function Description

dpm_sendevt()		sends the selected events	
<hr/>			
Name	int dpm_sendevt(devh, evttype, *evtdatap, evtlen, flags)		
Inputs	int devh:	device handle	
	unsigned long evttype:	Event type	
	void *evtdatap :	evttype	datablock
	short evtlen :	datablock	
	Unsigned short flags :	event bitmap	process
Returns	0 -1		
Includes	SmartSDK.h		
Category	Alarm		
Mode	Synchronous		
<hr/>			
■ Description			
device	open	process	.
Flags	,	process,	process
	.	process	sr_waitevt()
dpm_getevt()가			
.			
■ Termination Events			
.			
■ Dialogic			
dpm_sendevt()	evtdatap	DX_CST	.
■			
	Parameter	.	
<hr/>			


```

else
    printf("The handle for dpmiB1T1 is %d\n", dev);

printf("ALL (1), OTHERS (2), SELF (3) : ");
scanf("%d", &sendmode);
switch(sendmode) {
case 1:
    CST.cst_event = DE_R2TONE;
    CST.cst_data = 0x0101; // Group Number 1, R2 Number 1
    printf( "rc = %d\n",dpm_sendevt( dev, R2EV_SIG EVT, (void *)&CST,
    strlen(CST), EVFL_SENDOOTHERS));
    printf("case 1:Error - %s", ATDV_ERRMSGP(dev));
    break;
case 2:
    CST.cst_event = DE_R2TONE;
    CST.cst_data = 0x0101; // Group Number 1, R2 Number 1
    printf( "rc = %d\n",dpm_sendevt( dev, R2EV_SIG EVT, (void *)&CST,
    strlen(CST), EVFL_SENDSSELF));
    printf("case 1:Error - %s", ATDV_ERRMSGP(dev));
    break;
case 3:
    CST.cst_event = DE_R2TONE;
    CST.cst_data = 0x0101; // Group Number 1, R2 Number 1
    printf( "rc = %d\n",dpm_sendevt( dev, R2EV_SIG EVT, (void *)&CST,
    strlen(CST), EVFL_SENDALL));
    printf("case 1:Error - %s", ATDV_ERRMSGP(dev));
    break;
}
while (1) {
    sr_waitevt(-1);
    printf("event\n");
    switch( sr_getevtttype(0)
    {
case R2EV_SIG EVT:
    pCst = (DX_CST *)sr_getevtdatap(0);
    switch(pCst ->cst_event )

```



```

{
case DE_R2TONE:
R2Num = pCst -> cst_data & 0x00FF;
nGroup = pCst -> cst_data >> 8;
printf(" R2EV_SIG EVT : DE_R2TONE : group num(%d), R2 num(%d)" ,
        nGroup, R2Num);
        break;
case DE_R2EOS:
        printf(" R2EV_SIG EVT : DE_R2EOS" );
        break;
case DE_DTRING:
        printf(" R2EV_SIG EVT : DE_DTRING" );
        ,
        ,
        ,
        }
} // end of switch
} // end of while
dpm_close( dev );
}

```


DPMT_BDSGBIT() returns the current state of the transmit and receive

Name	char * DPMT_BDSGBIT (devh)
Inputs	int devh: Digital : digital device time slot signaling bit state 가
Returns	buffer : AT_FAILUREP
Includes	SmartSDK.h
Category	Extended Attribute
Mode	Synchronous

■ **Description**

digital board time slot signaling bit . 30
TX, RX signaling bit . 30byte buffer
, masking .

- DTSG_RCV A - "A" receive signaling bit
- DTSG_RCV B - "B" receive signaling bit
- DTSG_RCV C - "C" receive signaling bit (E-1 only)
- DTSG_RCV D - "D" receive signaling bit (E-1 only)
- DTSG_XMT A - "A" transmit signaling bit
- DTSG_XMT B - "B" transmit signaling bit
- DTSG_XMT C - "C" transmit signaling bit (E-1 only)
- DTSG_XMT D - "D" transmit signaling bit (E-1 only)

■ **Termination Events**

.

■ **Dialogic**

.

■

Parameter .


```

devh          dpm_open()          return    handle
              ( e.g. : devh = dpm_open(" dpmB1", 0 ) )

```

■ Example

```

#include <windows.h>
#include <stdio.h>
#include " SmartSDK.h"

main()
{
    int devh;                /* Board device handle */
    char *sigbits;           /* Pointer to signaling bits array */
    int i;                   /* Loop counter */
    int arcv, brcv, axmt, bxmt; /* Bit mask values */
    /*
    * Open board 1 device
    */
    if ( ( devh = dpm_open( "dpmB1", 0 ) ) == -1 ) {
        printf( "Cannot open board dpmiB1. errno = %d", errno );
        exit( 1 );
    }
    /*
    * Get current transmit and receive signaling bits of all time slots
    */
    if ( ( sigbits = DPMT_BDSGBIT ( devh ) ) == AT_FAILUREP ) {
        printf( "Error message = %s.", ATDV_ERRMSGP( devh ) );
        exit( 1 );
    }
    /*
    * Display it
    */
    for ( i = 0; i < 30; i++ ) {
        arcv = ( sigbits[ i ] & DTSG_RCVA ) ? 1 : 0;
        brcv = ( sigbits[ i ] & DTSG_RCVB ) ? 1 : 0;
    }
}

```



```

axmt = ( sigbits[ i ] & DTSG_XMTA ) ? 1 : 0;
bxmt = ( sigbits[ i ] & DTSG_XMTB ) ? 1 : 0;
printf( "tslot #%d arcv = %d, brcv = %d, axmt = %d, bxmt = %d\n",
i + 1, arcv, brcv, axmt, bxmt );
}
.
.
.
}

```


DPMT_DNLDVER()		returns the firmware version
<hr/>		
Name	char * DPMT_DNLDVER (devh)	
Inputs	int devh:	Digital
	CRN crn :	Call reference number
	char *numberstr :	
	int timeout :	time-out (: second)
	unsigned long mode :	async or sync
Returns	:	
	: AT_FAILUREP	
Includes	SmartSDK.h	
Category	Extended Attribute	
Mode	Synchronous	
<hr/>		

■ Description

■ Termination Events

■ Dialogic

Parameter			
devh	dpm_open()	return	handle

■ Example

```
#include <windows.h>
#include <stdio.h>
#include " SmartSDk.h"
```



```

void main()
{
    int bdev;
    char * version;
    if ((bdev = dpm_open("dpmB1", 0)) == -1) {
        printf("Error in dpm_open\n");
    }
    /* Get the version number */
    version = DPMT_DNLDVER (bdev);
    if (version == AT_FAILUREP) {
        printf("ERROR in getting version #\n");
    }
    printf(" firmware version %s", version);
}

```


DPMT_TSSGBIT()		retrieves the current state
<hr/>		
Name	long DPMT_TSSGBIT (devh)	
Inputs	int devh:	time slot
Returns	: time slot signaling bits : AT_FAILURE	
Includes	SmartSDK.h	
Category	Extended Attribute	
Mode	Synchronous	
<hr/>		

■ **Description**

time slot TX, RX signaling bit 가 .
masking .

- DTSG_RCV A - "A" receive signaling bit
- DTSG_RCV B - "B" receive signaling bit
- DTSG_RCV C - "C" receive signaling bit (E-1 only)
- DTSG_RCV D - "D" receive signaling bit (E-1 only)
- DTSG_XMT A - "A" transmit signaling bit
- DTSG_XMT B - "B" transmit signaling bit
- DTSG_XMT C - "C" transmit signaling bit (E-1 only)
- DTSG_XMT D - "D" transmit signaling bit (E-1 only)

■ **Termination Events**

.

■ **Dialogic**

.

■	Parameter	.
---	-----------	---

devh	dpm_open()	return	handle
-------------	------------	--------	--------

■ Example

```
#include <windows.h>
#include <srllib.h>
#include "SmartSDK.h"
main()
{
    int devh; /* Time slot device handle */
    long tsbits; /* Time slot signaling bits */
    int arcv, brcv, axmt, bxmt; /* Bit mask values */
    /*
    * Open board 1 time slot 1 device
    */
    if ( ( devh = dpm_open( "dpmB1C1", 0 ) ) == -1 ) {
        printf( "Cannot open time slot dpmB1C1. errno = %d", errno );
        exit( 1 );
    }
    /*
    * Get time slot signaling bits
    */
    tsbits = DPMT_TSSGBIT( devh );
    if ( tsbits == AT_FAILURE ) {
        printf( "Error message = %s.", ATDV_ERRMSGP( devh ) );
        exit( 1 );
    }
    /*
    * Display it
    */
    arcv = ( tsbits & DTSG_RCVA ) ? 1 : 0;
    brcv = ( tsbits & DTSG_RCVB ) ? 1 : 0;
    axmt = ( tsbits & DTSG_XMTA ) ? 1 : 0;
    bxmt = ( tsbits & DTSG_XMTB ) ? 1 : 0;
    printf( "tslot 1 arcv = %d, brcv = %d, axmt = %d, bxmt = %d\n",
        arcv, brcv, axmt, bxmt );
    .
    .
}
```


gets the current value

Name	int dpm_getparm(devh,param,valuep)
-------------	------------------------------------

Inputs int devh :

unsigned long param :

void *valuep :	가
----------------	---

parameter	value
-----------	-------

Returns 0
 -1

Includes SmartSDK.h

Category	Parameter Request
----------	-------------------

Mode	Synchronous
-------------	-------------

■ Description

dpm_setparm() setting device parameter value 가 .

■ Termination Events

•

■ Dialogic

Table 7.

■ Parameter

devh dpm_open() open time slot handle

param 가 device parameter .

valuep	가	parameter	value
--------	---	-----------	-------

pointer

Table 7. dpm_getparm()

Define	Value	
DPMCH_SIGNAL_TYPE	1 DTMF	Inbound signal type

	2 R2MF 0 Decadi	
DPMCH_DIAL_TYPE	1 DTMF 2 DPD 3 R2MF	dial type .
DPMCH_R2_TRACE	1	R2MF signal Trace
DPMCH_REQ_CALLINGNO	0 Without ANI 1 With ANI (default,) 2 가 (India)	ANI digit signaling Enable Disable .
DPMCH_DDI_MINDGT	Digit number	minimum digits
DPMCH_MAX_DIGIT	Digit number	maximum digits
DPMCH_CALLING_NO	Digits	
DPMCH_CLG_CATEGORY	0 - subscriber without priority 1 - subscriber with priority	subscriber category

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"
```

```
main()
{
    int devh; /* time slot handle */
    int value; /* Parameter value */
    int valuep; /* Parameter value */
    /*
    * Open board 1 device
    */
    if ( ( devh = dpm_open( "dpmB1C1", 0 ) ) == -1 ) {
        printf( "Cannot open board dpmB1C1. errno = %d", errno );
```



```

exit( 1 );
}

/*
 * Set current dial type parameter value
 */
value = 3;
if ( dpm_setparm( devh, DPMCH_DIAL_TYPE, ( void *)&value ) == -1 ) {
printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
exit( 1 );
}

/*
 * Get current dial type parameter value
 */
if ( dpm_getparm( devh, DPMCH_DIAL_TYPE, ( void *)&valuep ) == -1 ) {
printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
exit( 1 );
}
.
}

```


dpm_getevt()

blocks and returns control to the program

Name	int dpm_getevt(devh, eblkp, timeout)		
Inputs	int devh :	time slot	
	DX_EBLK *ebkp :	Event Block Structure	pointer
	int timeout :	time-out	(: second)
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	Parameter Request		
Mode	Synchronous		

■ Description

dpm_setevtmask() set event가 block 가,
event가 . timeout .
multi-threaded aplication .

■ Termination Events

■ Dialogic

Dialogic DNA SDK 3.3 EV_EBLK .

■	Parameter	.
---	-----------	---

devh	dpm_open()	return	handle
eblkp	event	event data가	pointer
typedef struct dx_eblk {			
		unsigned short ev_event;	/* Event that occurred */
		unsigned short ev_data;	/* Event-specific data */


```

/*
 * Wait for events on this timeslot
 */
dt_getevt ( devh, &eblk, -1 ); /* Wait for ever */

unsigned short nGroup = 0x00, nR2Num = 0x00;
switch( ebld.ev_event )
{
case DE_R2TONE:
    nR2Num = ebld.ev_data & 0x00FF;      // R2 Number
    nGroup = ebld.ev_data >> 8;         // Group
    printf ("Receive DE_R2TONE event; Group:%d, R2 Number:%d",
            nGroup, nR2Num);

    break;

case DE_R2EOS:
    printf("Receive DE_R2EOS event");
    break;

case DE_DTRING:
    printf("Receive DE_R2EOS event");
    break;

case DE_DTHOOKON:
    printf("Receive DE_R2EOS event");
    break;

case DE_ANSWERED:
    printf("Receive DE_R2EOS event");
    break;
} // end of switch.

.
.
}

```


dpm_getevtmsk()	retrieves the current event bitmask(s)
-------------------------	---

dpm_getevtmsk()	retrieves the current event bitmask(s)
-------------------------	---

Name	int dpm_getevtmask(devh,event,bitmaskp)		
Inputs	int devh :	time slot	
	int event :	bitmask	가
	unsigned short *bitmaskp :	bitmask	
Returns	0 -1		
Includes	SmartSDK.h		
Category	Parameter Request		
Mode	Synchronous		

■ Description

Digital Network Interface time slot bitmask

■ Termination Events

■ Dialogic

■ Parameter

int devh	dpm_open()	return	handle
int event	가ㅓ event		
unsigned short *bitmaskp	bitmask		

■ Example


```

#include <windows.h>
#include <stdio.h>
#include "SmartSDk.h"

DX_EBLK eblk;
main()
{
    int devh; /* Board device handle */
    unsigned short bitmaskp; /* Bitmask variable */
    unsigned short sigmsk = DM_R2TONE | DM_R2EOS | DM_DTHOOKON |
    DM_DTRING | DM_ANSWERED;

    short sig, indx;
    /*
    * Open Timeslot 1 device
    */
    if ( ( devh = dpm_open( "dpmB1C1", 0 ) ) == -1 ) {
        printf( "Cannot open timeslot dpmiB1T1. errno = %d", errno );
        exit( 1 );
    }
    if (dpm_setevtmsk(devh, R2EV_SIGEVT, sigmsk, DTA_SETMSK) == -1) {
        // Error
        dpm_close(devh);
        exit(1);
    }

    if ( dpm_getevtmsk( devh, R2EV_SIGEVT, &bitmaskp ) == -1 ) {
        printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
        exit( 1 );
    }

    .
    .
}

```


dpm_setparm()

changes the value of a device parameter

Name	int dpm_setparm(devh, param, valuep)		
Inputs	int devh:		
	unsigned long param :		
	Void *valuep :	dev i ce	
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	Parameter Setting		
Mode	Synchronous		

■ Description

Digital Network Interface device .
value Table 7. .

■ Termination Events

.

■ Dialogic

.

■ Parameter

int devh	dpm_open()	return	handle
unsigned long param	value	parameter	
	(dpm_getparm()	Table 7.)
void *valuep	device parameter		
가 Digital Network Interface device time slot			
close	.		

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"

main()
{
    int devh; /* time slot handle */
    int value; /* Parameter value */
    int valuep; /* Parameter value */
    /*
    * Open board 1 device
    */
    if ( ( devh = dpm_open( "dpmB1", 0 ) ) == -1 ) {
        printf( "Cannot open board dpmB1. errno = %d", errno );
        exit( 1 );
    }

    /*
    * Set current dial type parameter value
    */
    value = 3;
    if ( dpm_setparm( devh, DPMCH_DIAL_TYPE, ( void * )&value ) == -1 ) {
        printf( "Error message = %s.",ATDV_ERRMSGP( devh ) );
        exit( 1 );
    }
    .
    .
    .
}
```


dpm_setevtmask() enables and disables notification for events

Name	dpm_setevtmask(devh,event,bitmask,action)			
Inputs	int devh :	time slot		
	int event	enable	disable	event
	unsigned short bitmask :	event	bitmask	
	int action :	masking	(, 가)	
Returns	0			
	-1			
Includes	SmartSDK.h			
Category	Parameter Setting			
Mode	Synchronous			

■ **Description**

device handle enable disable . Bitmask
action add, set, subtract .
bitmask bitwise operation (‘ | ’ or) 가 .

■ **Termination Events**

.

■ **Dialogic**

.

■ Parameter .

int devh	dpm_open()	return	handle
int event	enabled/disabled	event	
unsigned short bitmask	event	bitmask	
int action	masking	(, 가)	

Table 10. dpm_setevtmsk() action

Action	
DTA_SETMSK	bitmask diable bitmask enable .
DTA_ADDMSK	bitmask 가 bitmask enable .
DTA_SUBMSK	bitmask bitmask disable .

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"

main()
{
    int devh; /* Board device handle */
    unsigned short bitmaskp; /* Bitmask variable */
    unsigned short sigmsk = DM_R2TONE | DM_R2EOS | DM_DTHOOKON |
    DM_DTRING | DM_ANSWERED;

    short sig, indx;
    /*
    * Open Timeslot 1 device
    */
    if ( ( devh = dpm_open( "dpmB1C1", 0 ) ) == -1 ) {
        printf( "Cannot open timeslot dpmiB1T1. errno = %d", errno );
        exit( 1 );
    }
    if (dpm_setevtmsk(devh, R2EV_SIGEVT, sigmsk, DTA_SETMSK) == -1) {
        // Error
        dpm_close(devh);
        exit(1);
    }
}
```


}

.

.

.

dpm_open() opens a Digital Network Interface device

dpm_open() opens a Digital Network Interface device

dpm_open() opens a Digital Network Interface device

■ Description

Description	Digital network interface device	open
--------------------	----------------------------------	------

■ Termination Events

■ Te .

■ Dialogic

■ **Dia**

■ Parameter



Parameter	Description
char *name	time slot (null-terminated string)
int oflags	open attribute flags()

■ Example

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"
```



```

main()
{
int devh; /* Board device handle */
/*
* Open board 1 device
*/
if ( ( devh = dpm_open( "dpmB1", 0 ) ) == -1 ) {
printf( "Cannot open board dpmB1C1. errno = %d", errno );
exit( 1 );
}
}

```


dpm_close() **closes Digital Network Interface devices**

Name	int dpm_close(devh)
Inputs	int devh : , time slot
Returns	0 -1
Includes	SmartSDK.h
Category	Resource Management
Mode	Synchronous

■ **Description**

dpm_open() open Digital network interface device close

■ **Termination Events**

.

■ **Dialogic**

.

■

Parameter .

Devh	dpm_open()	return	handle
-------------	------------	--------	--------

■ **Example**

```
#include <windows.h>
#include <stdio.h>
#include " SmartSDK.h"

main()
{
int devh; /* Board device handle */
/*
```



```

* Open board 1 device
*/
if ( ( devh = dpm_open( "dpmB1", 0 ) ) == -1 ) {
printf( "Cannot open board dpmB1. errno = %d", errno );
exit( 1 );
}
/*
* Continue processing
*/
.
.
.
/*
* Done processing - close device.
*/
if ( dpm_close( devh ) == -1 ) {
printf( "Cannot close board dpmB1. errno = %d", errno );
}
}

```


dpm_getxmitslot()		returns the SCbus time slot
<hr/>		
Name	int dpm_getxmitslot(devh, sc_tsinfop)	
Inputs	int devh :	time slot
	SC_TSINFO *sc_tsinfop	SCbus time slot information structure
Returns	0	
	-1	
Includes	SmartSDK.h	
Category	SCbus Routing	
Mode	Synchronous	
<hr/>		

■
 Description

network device TX network SCbus time slot number
 return . SCbus time slot information SC_TSINFO return
 device transmit channel network SCbus time slot number .

■
 Termination Events

.

■
 Dialogic

.

■	Parameter	.
---	-----------	---

Devh	dpm_open()	return	handle
SC_TSINFO *sc_tsinfop	SCbus time slot information structure		
typedef struct {			
	unsigned long sc_numts;		
	long *sc_tsarrayp;		
	} SC_TSINFO		

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"

main( )
{
    int devh; /* Time slot device handle */
    SC_TSINFO sc_tsinfo; /* Time slot information structure */
    long scts; /* SCbus time slot */
    /* Open board 1 time slot 1 for Digital network interface device */
    if ((devh = dpm_open("dpmB1C1", 0)) == -1) {
        printf("Cannot open time slot dpmB1C1. errno = %d", errno);
        exit(1);
    }
    /* Fill in the SCbus time slot information */
    sc_tsinfo.sc_numts = 1;
    sc_tsinfo.sc_tsarrayp = &scts;
    /* Get SCbus time slot connected to transmit of time slot (digital
    channel) 1 on board 1 */
    if (dpm_getxmitslot(devh, &sc_tsinfo) == -1) {
        printf("Error message = %s", ATDV_ERRMSGP(devh));
        exit(1);
    }
    printf("transmitting on SCbus time slot %d", scts);
}
```


dpm_listen() **connects the digital listen channel**

Name	int dpm_listen(devh, sc_tsinfop)		
Inputs	int devh :	time slot	
	SC_TSINFO *sc_tsinfop	SCbus time slot information structure	
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	SCbus Routing		
Mode	Synchronous		

■ **Description**

RX . SC_TSINFO time slot network device

■ **Termination Events**

.

■ **Dialogic**

.

■

Parameter .

Devh	dpm_open()	return	handle
SC_TSINFO *sc_tsinfop	SCbus time slot information structure		

```
typedef struct {
    unsigned long sc_numts;
    long *sc_tsarrayp;
} SC_TSINFO;
```


■ Example

```
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"

main( )
{
    int voxh; /* Voice channel device handle */
    int dpmih; /* Digital channel (time slot) device handle */
    SC_TSINFO sc_tsinfo; /* Time slot information structure */
    long scts; /* SCbus time slot */
    /* Open board 1 channel 1 device */
    if ((voxh = vpm_open("dpmB1C1", 0)) == -1) {
        printf("Cannot open channel dpmB1C1. errno = %d", errno);
        exit(1);
    }
    /* Fill in the SCbus time slot information */
    sc_tsinfo.sc_numts = 1;
    sc_tsinfo.sc_tsarrayp = &scts;
    /* Get SCbus time slot connected to transmit of channel 1 on board 1 */
    if (vpm_getxmitslot(voxh, &sc_tsinfo) == -1) {
        printf("Error message = %s", ATDV_ERRMSGP(voxh));
        exit(1);
    }
    /* Open board 1 time slot 1 on Digital network interface device */
    if ((dpmh = dpm_open("dpmB1C1", 0)) == -1) {
        printf("Cannot open time slot dpmB1C1. errno = %d", errno);
        exit(1);
    }
    /* Connect the receive of digital channel (time slot) 1 on board 1 to
    SCbus transmit time slot of voice channel 1 */
    if (dpm_listen(dpmh, &sc_tsinfo) == -1) {
        printf("Error message = %s", ATDV_ERRMSGP(dpmih));
        exit(1);
    }
}
```


dpm_unlisten()		disconnects the receive	
Name	int dpm_unlisten(devh)		
Inputs	int devh :	time slot	
Returns	0 -1		
Includes	SmartSDK.h		
Category	SCbus Routing		
Mode	Synchronous		

■ Description

dpm_listen() network device RX Scbus
time slot disconnect .

■ Termination Events

.

■ Dialogic

.

■	Parameter	.
---	-----------	---

Devh dpm_open() return handle

■ Example

```
#include <windows.h>
#include <stdio.h>
#include " SmartSDK.h"

main( )
{
int devh; /* Digital channel (time slot) device handle */
```



```

int voxh; /* Voice channel device handle */
int dpmih; /* Digital channel (time slot) device handle */
SC_TSINFO sc_tsinfo; /* Time slot information structure */
long scts; /* SCbus time slot */
/* Open board 1 channel 1 device */
if ((voxh = vpm_open("dpmB1C1", 0)) == -1) {
printf("Cannot open channel dpmB1C1. errno = %d", errno);
exit(1);
}
/* Fill in the SCbus time slot information */
sc_tsinfo.sc_numts = 1;
sc_tsinfo.sc_tsarrayp = &scts;
/* Get SCbus time slot connected to transmit of channel 1 on board 1 */
if (vpm_getxmitslot(voxh, &sc_tsinfo) == -1) {
printf("Error message = %s", ATDV_ERRMSGP(voxh));
exit(1);
}
/* Open board 1 time slot 1 on Digital network interface device */
if ((dpmh = dpm_open("dpmB1C1", 0)) == -1) {
printf("Cannot open time slot dpmB1C1. errno = %d", errno);
exit(1);
}
/* Connect the receive of digital channel (time slot) 1 on board 1 to
SCbus transmit time slot of voice channel 1*/
if (dpm_listen(dpmh, &sc_tsinfo) == -1) {
printf("Error message = %s", ATDV_ERRMSGP(dpmih));
exit(1);
}
/* Disconnect receive of board 1, time slot 1 from all SCbus time
slots */
if (dpm_unlisten(devh) == -1) {
printf("Error message = %s", ATDV_ERRMSGP(devh));
exit(1);
}
}

```


dpm_dial()

allows the application to pulse dial

Name	int dpm_dial(devh,digstr,tmo)
Inputs	int devh : time slot char *digstr : dialing number (null-terminated string) unsigned int tmo : timeout
Returns	0 -1
Includes	SmartSDK.h
Category	Time Slot Signaling
Mode	Synchronous

■ Description

CT-V30A Pulse DTMF call setup 가 .
Pulse DTMF call setup 가 setting .
dpm_dial() dial number outbound pulse dialing .
dpm_setparm() pluse dial type , dpm_seizure()
(Table 5.)
outbound pulse dialing vpm_dial() 가 . digit string
pulse dialing 'P' 가 . DTMF outbound call setup digit
string 'T' 가 digit string . vpm_dial()
'voice programmer' s guide for Window NT' vpm_dial()
.
Pulse DTMF **Inbound call setup** dpm_setparm() pulse
DTMF (Table 5.), **vpm_getdig()**
. vpm_getdig() 'voice programmer' s guide for Window
NT ' vpm_getdig() .
vpm_getdigit() DV_TPT Structure 가
, pulse DTMF digit **dpm_answer()**
Inbound call setup .

■ Termination Events

■ Dialogic



Parameter

Devh	dpm_open()	return	handle
char *digstr	pointer to an ASCIIZ string of digits. 32digit		
unsigned int tmo	timeout value (: second) dialing		

■ Example

/**** Outbound pulse dial ******/**

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

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

```
#include "SmartSDK.h"
```

```
void main()
```

```
{
```

```
    int nDev;
```

```
    int Value;
```

```
    if ( (nDev = dpm_open(" dpmB1C1", 0 ) < 0 )
```

```
    {
```

```
        // error
```

```
        printf(" dpm_open( dpmB1C1, 0 ) : Error\n");
```

```
        return;
```

```
    }
```

```
    Value = 2; // Pulse Dial type
```

```
    if( dpm_setparam( nDev, DPMCH_DIAL_TYPE, &Value) < 0 )
```

```
    {
```

```
        // error
```

```
        printf(" dpm_setparam( ) : Error\n");
```



```

        return;
    }
    if( nr_scroute( nDev, SC_DPM, nDev, SC_DSP, SC_FULLDUP) < 0 )
    {
        //Error
        printf(" nr_scroute() : Error\n");
        return;
    }

    if( dpm_seizure(nDev, EV_SYNC) < 0 )
    {
        // error
        printf(" dpm_seizure(%d, EV_SYNC) : Error", nDev);
        return;
    }
    if( dpm_dial(nDev, "4001", 60) < 0 )
    {
        // error
        printf(" dpm_dial(%d, 4001, 60) : Error\n", nDev);
        return;
    }

    .
    .
    .

}

/***** Outbound DTMF dial *****/
#include <windows.h>
#include <stdio.h>
#include " SmartSDK.h"

void main()
{
    int nDev;

```



```

int Value, cares;
DX_CAP capp; // Call Analysis parameter

if ( (nDev = dpm_open(" dpmB1C1", 0) < 0 )
{
    // error
    printf(" dpm_open( dpmB1C1, 0 ) : Error");
    return;
}

Value = 1; // DTMF Dial type
if( dpm_setparm( nDev, DPMCH_DIAL_TYPE, &Value) < 0 )
{
    // error
    printf(" dpm_setparm( ) : Error");
    return;
}

if( nr_scroute( nDev, SC_DPM, nDev, SC_DSP, SC_FULLDUP) < 0 )
{
    //Error
    printf(" nr_scroute() : Error\n");
    return;
}

if( dpm_seizure(nDev, EV_SYNC) < 0 )
{
    // error
    printf(" dpm_seizure(%d, EV_SYNC) : Error", nDev);
    return;
}

/* 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,"4001",&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_CEPT: /* Operator Intercept detected */
    break;
case CR_BUSY:
    .
    .
}
.
.
}

```

/**** Inbound Pulse or DTMF signal *****/**

```

#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"

```

```

void GetDigitAndAnswer( nDev);

```

```

void main()
{

```

```

    int nDev;
    int Value;

```



```
DX_EBLK eblk;
```

```
if ( (nDev = dpm_open(" dpmB1C1", 0) < 0 )  
{  
    // error  
    printf(" dpm_open( dpmB1C1, 0 ) : Error");  
    return;  
}
```

```
Value = 2; // Pulse signal type, DTMF signal type : 1  
if( dpm_setparm( nDev, DPMCH_SIGNAL_TYPE, &Value) < 0 )  
{  
    // error  
    printf(" dpm_setparm( ) : Error");  
    return;  
}
```

```
if( nr_scroute( nDev, SC_DPM, nDev, SC_DSP, SC_FULLDUP) < 0 )  
{  
    //Error  
    printf(" nr_scroute() : Error\n");  
    return;  
}
```

```
if( dpm_getevt(nDev, &eblk, -1 ) < 0 ) //      가  
{  
    //Error  
    printf(" dpm_getevt ( ) : Error\n");  
    return;  
}
```

```
switch( eblk.ev_event )  
{  
case DE_DTRING:  
    GetDigitAndAnswer();
```



```

        break;
    case DE_DTHOOKON:
        dpm_release(nDev, EV_SYNC);
        break;
    case DE_ANSWERED:
        // Connected
        break;
    }
    .
    .
    .
}

void GetDigitAndAnswer( nDev)
{
    DV_TPT tpt[2];
    DV_DIGIT dtbuf;

    if( vpm_clrdigbuf(nDev) < 0 )
    {
        // Error
        printf(" vpm_clrdigbuf() : Error\n");
        return;
    }

    tpt[0].tp_type = IO_CONT;
    tpt[0].tp_termno = DX_MAXDTMF;
    tpt[0].tp_length = 8;                // allow 8 digit
    tpt[0].tp_flags = TF_MAXDTMF;

    tpt[1].tp_ttp_type = IO_EOT;
    tpt[1].tp_ttp_termno = DX_IDDTIME;
    tpt[1].tp_ltp_length = 100;
    tpt[1].tp_ftp_flags = TF_IDDTIME;
    if( vpm_getdig(nDev, &tpt[0], &dtbuf, EV_SYNC) < 0 )
    {

```



```

        // Error
        printf(" vpm_getdig() : Error\n");
        return;
    }

    if( dpm_answer(nDev, EV_SYNC) < 0 )
    {
        // Error
        printf(" dpm_answer() : Error\n");
        return;
    }

    printf(" Connected\n");
}

```


seizure a E1 line


```

int  nDev;
if( (nDev = dpm_open(" dpmB1C1", 0 )) < 0 )
{
    // error
    return;
}
if( dpm_setparm( nDev, DPMCH_SIGNAL_TYPE, 3 ) < 0 )
{
    // Error
    return;
}
if( dpm_seizure( nDev, EV_SYNC ) < 0 )
{
    //Error
    return;
}

// R2 signaling
.
.
..
}

```


dpm_answer()
function

equivalent to conventional “set hook off”

Name	int dpm_answer(devh, mode)
Inputs	int devh : time slot unsigned short mode : async or sync
Returns	0 -1
Includes	SmartSDK.h
Category	Time Slot Signaling
Mode	Synchronous, Asynchronous

■ Description

Inbound call incomming call . “set hook off”

■ Termination Event

R2EV_ANSWER –

■ Dialogic

Dialogic DNA SDK 3.3 가 .

dt_setsigmod() dt_settssig()

	Parameter		
dev h	dpm_open()	return	handle
mode	EV_SYNC :		
	EV_ASYNC :		

■ Example

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

```
#include "SmartSDK.h"
```



```

// Inbound Call          R2 signaling          DE_R2EOS          .
.
.
int nDev, nEvent;
DX_CST * pCst;
unsigned short nR2Num;
unsigned short nGroup;

while ( 1 )
{
    sr_waitevt( -1 );
    nDev = sr_getevtdev( );
    nEvent = sr_getevttype( );

    switch ( nEvent )
    {
    case R2EV_ SIGEVT:
        pCst = (DX_CST *)sr_getevtdatap( );
        switch( pCst->cst_event )
        {
        case DE_R2TONE:
            nR2Num = pCst->cst_data & 0x00FF; // R2 digit
            nGroup = pCst->cst_data >> 8; // Group number
            // R2 signal
            .
            .
            break;
        case DE_R2EOS:
            printf(" receive R2EV_ SIGEVT : DE_R2TONE\n");
            if( dpm_answer( nDev, EV_ASYNC ) < 0 )
            {
                // error
            }
            break;
        case DE_DTRING:

```



```
        printf(" receive R2EV_SIGEV_T : DE_DTRING \n");
break;
case DE_DTHOOKON:
        printf(" receive R2EV_SIGEV_T : DE_DTHOOKON \n");
break;
case DE_ANSWERED:
        printf(" receive R2EV_SIGEV_T : DE_ANSWERED \n");
break;
}
```

```
break;
case R2EV_ANSWER:
break;
.
.
.
}
```

```
}
```


dpm_playR2()

send R2 signal

Name	int dpm_playR2(devh, nGroup, nDigit)	
Inputs	int devh :	time slot
	int nGroup :	Group number (I, II, A, B)
	int nDigit :	digit (1 ~ 15)
Returns	0	
	-1	
Includes	SmartSDK.h	
Category	Time Slot Signaling	
Mode	Synchronous	

■ Description

R2 signal network .

R2 signaling Group Forward(Group I,II) Group Backward(Group A, B) . 4 group Application Group II

Group B .

, Group Number change Group I(R2_GROUP_1) Group II(R2_GROUP_2) change R2_GROUP_2 R2_GROUP_1 .

Forward (R2_GROUP_1) Backward (R2_GROUP_A) .

calling party Forward Group(Group I) digit 가

category digit (II-x) , Application category digit

Forward Group(Group I-x) digit .

Backward R2_GROUP_A R2_GROUP_B change R2_GROUP_A

.

■ Termination Events

.

■ Dialogic

Dialogic DNA SDK 3.3 가 .

	dt_setsigmod()	dt_settssig()	
Parameter			
dev h	dpm_open()	return	handle . time slot
nGroup	Group Number.		
	Forward : R2_GROUP_1		
	Backward : R2_GROUP_A		
nDigit	digit (1 ~ 15)		
	R2_NO_15 : END OF PULSING		

■ Example

```
#include "SmartSDK.h"
.
.
typedef PORT_INFO_tag
{
    char    sDialDigit[20]; int nldxDial;
    char    sANldigit[20]; int nldxANI;
    short   nChangedGroup;
} PORT_INFO;

PORT_INFO g_PortInfo; // Global variable
/*
* g_PortInfo.sDialDigit    "1234" string          가
* g_PortInfo.nChangedGroup, g_PortInfo.nldxDial, g_PortInfo.nldxANI      0

*
*/
.
.
int nDev, nEvent;
DX_CST * pCst;
unsigned short nR2Num;
unsigned short nGroup;
```



```

while ( 1 )
{
    sr_waitevt( -1 );
    nDev = sr_getevtdev( );
    nEvent = sr_getevttype( );

    switch ( nEvent )
    {
    case R2EV_ SIGEVT:
        pCst = (DX_CST *)sr_getevtdatap( );
        switch( pCst->cst_event )
        {
        case DE_R2TONE:
            nR2Num = pCst->cst_data & 0x00FF; // R2 digit
            nGroup = pCst->cst_data >> 8; // Group number
            // R2 signal
            ProcessR2Signal(nDev, nGroup, nR2Num);
            .
            .
        .
        .
        .
    }
}

void ProcessR2Signal(int nDev, unsigned short nGroup, unsigned short nR2Num)
{
    unsigned short nNextDigit;

    if( nGroup == R2_GROUP_A ) // received backward signal
    {
        switch( nR2Num )
        {
        case R2_NO_1:
            if( sDialDigit[ g_PortInfo.nIdxDial ] == ' \0' ) //dial digit
            {

```



```

        // send end of pulsing
        if( dpm_playR2( nDev, R2_GROUP_1, R2_NO_15 ) < 0 )
        { // Error
        }
    }
    else
    {
        nNextDigit = 0x30 - sDialDigit[ g_PortInfo.nIdxDial++];
        if( nNextDigit == 0 )
            nNextDigit = R2_NO_10;
        // send next digit
        if( dpm_playR2( nDev, R2_GROUP_1, nNextDigit ) < 0 )
        { // Error
        }
    }
    break;
    .
    .
    .
    .
    }
}

.
.
.
}

```


Release a E1 channel

■ Description

■ Termination Event

■ Dialogic

■ Parameter .

EV_ASYNC:

■ Example

```
#include "SmartSDK.h"
```

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

```
.
```

```
.
```

```
int nDev, nEvent;
```

```
DX_CST * pCst;
```

```
unsigned short nR2Num;
```

```
unsigned short nGroup;
```

```
while ( 1 )
```

```
{
```

```
    sr_waitevt( -1 );
```

```
    nDev = sr_getevtdev( );
```

```
    nEvent = sr_getevttype( );
```

```
    switch ( nEvent )
```

```
    {
```

```
        case R2EV_ SIGEVT:
```

```
            pCst = (DX_CST *)sr_getevtdatap( );
```

```
            switch( pCst->cst_event )
```

```
            {
```

```
                case DE_R2TONE:
```

```
                    nR2Num = pCst->cst_data & 0x00FF; // R2 digit
```

```
                    nGroup = pCst->cst_data >> 8; // Group number
```

```
                    // R2 signal
```

```
                    .
```

```
                    .
```

```
                break;
```

```
                case DE_R2EOS:
```

```
                    printf(" receive R2EV_ SIGEVT : DE_R2TONE\n");
```

```
                    if( dpm_answer( nDev, EV_ASYNC ) < 0 )
```

```
                    {
```

```
                        // error
```

```
                    }
```



```

        break;
    case DE_DTRING:
        printf(" receive R2EV_SIG EVT : DE_DTRING \n");
        break;
    case DE_DTHOOKON:
        printf(" receive R2EV_SIG EVT : DE_DTHOOKON \n");
        if( dpm_release( nDev, EV_ASYNC ) < 0 )
        { // Error
        }
        break;
    .
    .
    .
}
break;
case R2EV_RELEASE:
    printf(" receive R2EV_RELEASE\n");
break;
.
.
.
}

```


block a E1 channel


```

{
    int    ndev;

    // device open
    ndev = dpm_open(" dpmB1C1" , 0);
    if( ndev < 0 )
    {
        printf(" dpm_open() : Error!\n");
        return;
    }

    // Block
    if( dpm_block( ndev, SCT_DPM_BLOCK) < 0 )
    {
        printf(" dpm_block(): Error!\n");
        return;
    }

    // Unblock
    if( dpm_block( ndev, SCT_DPM_UNBLOCK) < 0 )
    {
        printf(" dpm_block(): Error!\n");
        return;
    }
    // device close
    if( dpm_close( ndev) < 0 )
    {
        printf(" dpm_close(): Error!\n");
        return;
    }
}

```


4. Logical Call Function Reference

4.1 Logical Call Reference Overview

Logical Call API multiple signaling system common API .
 , network transactions (call establishment, call maintenance, call clearing ,
 call signaling control , application) . ,
 OS API .
 hardware signaling call line device
 identifier(LDID) call Reference Number(CRN) .
 line device identifier(LDID) Call Reference Number (CRN) lc_open()

Dialogic DNA SDK 3.3 CRN Dialogic gc_WaitCall()
 GCEV_OFFERED , gc_MakeCall() 가
 lc_Open() 가

NOTE : Logical Call library dpm.cfg .
 service restart .

```
:
parameter 44:
-----
lc_ or dpm_ use flag
$44 lc(logical call library) use flag (0=not use, 1=use) : 1
:
```

4.3 Logical Call States

Logical Call event function termination event call state가
 . Call State .
 • Function Call Return.
 • Termination events. (가 .)
 • Unsolicited events.

4.3.1 Basic Call Model –

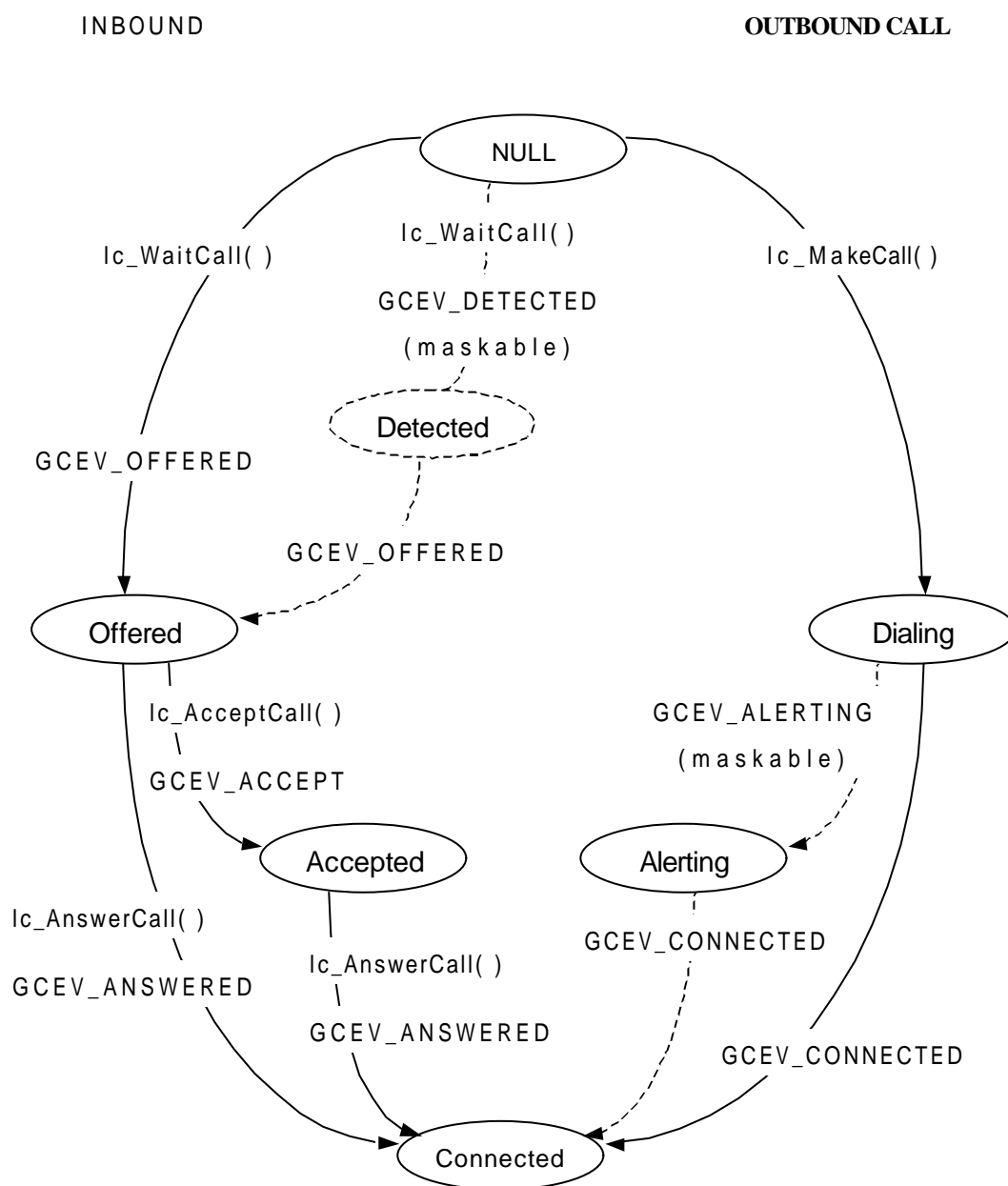


Figure 1. Basic Asynchronous Call Establishment state Diagram

Table 11. Call State

state	
Accepted (GCST_ACCEPTED)	Calling party 가 . Inbound call connect accept 가 .
Alerting (GCST_ALERTING)	Dialing 가 . Application Answer . (GCEV_ALERTING) application , masking .
Connected (GCST_CONNECTED)	Inbond call outbound call connect , 가 가 .
Detected (GCST_DETECTED)	Incomming call application offer . line device outbound call initiate .
Dialing (GCST_DIALING)	call establishment가 . Dialing 가 , network 가 .
Disconnected (GCST_DISCONNECTED)	Network call disconnect . Application call drop CRN resource 가 .
Idle (GCST_IDLE)	call drop 가 . Application CRN resource 가 .
Null (GCST_NULL)	Device(time slot or line) call 가 .
Offered (GCST_OFFERED)	Network Application call offer 가 . Call receive , ANI DNIS .

Inbound Call

Application NULL State open line device lc_WaitCall()
 . NULL state line device가 lc_Open() open
 lc_release() 가 , lc_ResetLineDev() 가
 State .
 lc_WaitCall() 가 , inbound call unsolicited
 GCEV_DETECTED . GCEV_DETECTED lc_SetEvtmsk()

mask, disable
 . Call application offered unsolicited GCEV_OFFERED가
 . call State Offered state
 Connected State Accepted State . Connected State
 가 Ic_AnswerCall()
 GCEV_ANSWERED Call State Offered state
 Connected State
 , Application Offered inbound call Answer 가
 Ic_AcceptCall() call Connect
 Application 가 parameters , Routing,
 interval 가
 GCEV_ACCEPT 가 , state Accepted State
 .
 Offered State Accepted State ANI, DNIS
 Ic_GetANI(), Ic_GetDNIS()
 Offered State Call reject Ic_DropCall(), Ic_ReleaseCall()
 .
 E-1 CAS protocol Accepted State reject Ic_AcceptCall()
 Ic_DropCall()
 call error Application
 Ic_AnswerCall(), Ic_DropCall(), Ic_ReleaseCall()
 GCEV_DISCONNECTED 가 Ic_DropCall()
 .
 Call establishment GCEV_TASKFAIL call State
 Application NULL State
 Ic_DropCall(), Ic_ReleaseCall() 가 trunk error
 Application Ic_ResetLineDev() line device
 reset NULL State

Table 8. Inbound Call Set-Up ()

Function / Event	Action / Description
*Ic_SetEvtMsk()	Maskable Enable Disable .
Ic_WaitCall()	Open line device Call .
*GCEV_DETECTED	Incomming call Application offer
GCEV_OFFERED	Inbound Call , Offered State .

*Ic_GetANI()	Caller ID
*Ic_GetDNIS()	DNIS
*Ic_AcceptCall()	call answer
*GCEV_ACCEPT	Termination call Answer , Accepted State
Ic_AnswerCall()	Inbound call answer call connect 가
GCEV_ANSWERED	Termination inbound call Connect
* = optional maskable	

Outbound Call

Outbound Call initiate application open line device
dpm_setparm(linedev, DPMCH_DIAL_TYPE, 3) Ic_MakeCall()
. Ic_MakeCall() Call State Dialing State
Ic_MakeCall() 가 line device NULL
Dialing State dialing 가 , network
call State Connected state Alerting State
Called party 가 call GCEV_CONNECTED 가
. call connect , call State가 Connected State
. , 가
Called party가 call 가 GCEV_ALERTING
. called party가 answer network
Called party가 connection
GCEV_ALERTING call State Alerting State
E-1 CAS system GCEV_ALERTING remote end가 ringback
, ISDN remote end가 Alerting
.
Remote end가 call answer call connect . GCEV_CONNECTED
Call State Connected State
GCEV_CONNECTED call Connect
, GCEV_TASKFAIL GCEV_DISCONNECTED GCEV_CALLSTATUS
.
Application GCEV_TASKFAIL local 가
, GCEV_DISCONNECTED remote end가 answer
.

Call connect

lc_ResultValue() lc_ResultMsg()

Ic_MakeCall()

Table 9. Outbound Call Set-Up ()

Function / Event	Action / Description
dpm_setparm()	Digital Network Interface device .
*lc_SetEvtMsk()	Maskable Enable Disable .
lc_MakeCall()	Connect GCEV_CONNECTED , GCEV_DISCONNECTED GCEV_CALLSTATUS GCEV_TASKFAIL .
*GCEV_ALERTING	Called party call answer .
GCEV_CONNECTED	lc_MakeCall() , Call connect .
* = optional	maskable

Call Termination

Figure 2. Call (termination) (teardown)
Call State .

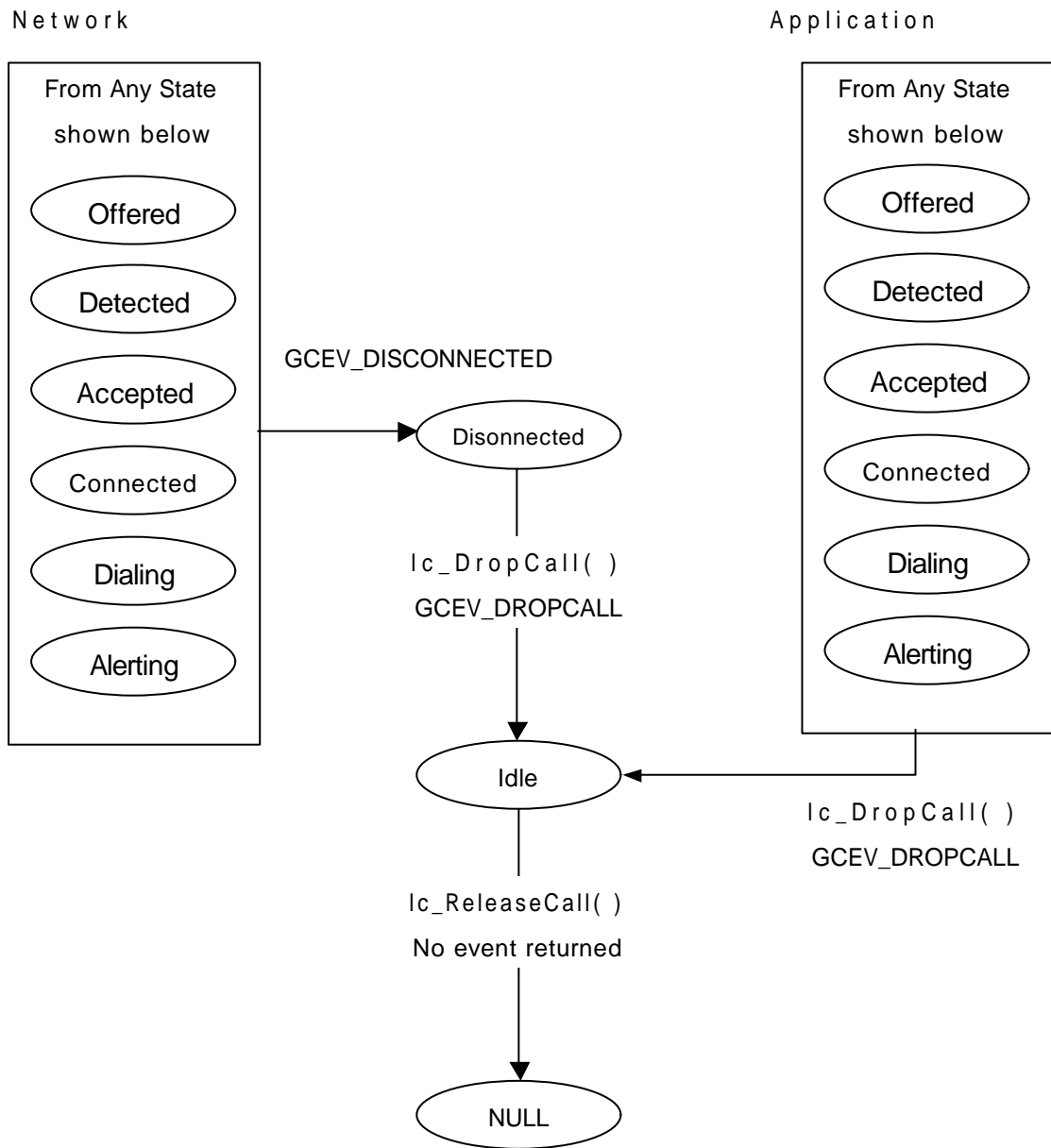


Figure 2. Asynchronous Call Tear-Down State Diagram

Application `Ic_DropCall()` call disconnect .
`Ic_DropCall()` call State idle state . Idle State가

call disconnect , Application call
resource lc_ReleaseCall()
call State Null State
Connect call disconnect lc_DropCall()
GCEV_DISCONNECTED 가
Application GCEV_DISCONNECTED Remote End
call disconnect call setup error가
call State disconnected state
Application lc_DropCall()
“set hook on” , call resource
lc_ReleaseCall()

Table 12. Call Termination ()

Function / Event	Action / Description
GCEV_DISCONNECTED	Network call termination Call State disconnected State
lc_DropCall()	Call disconnect
GCEV_DROPCALL	Termination call Disconnect Idle State
lc_ReleaseCall()	call resource call State NULL State , Network port call initiate
* = optional maskable	

4.3.1 Basic Call Model –

Function Call Return Unsolicited events Call State
가 , Unsolicited

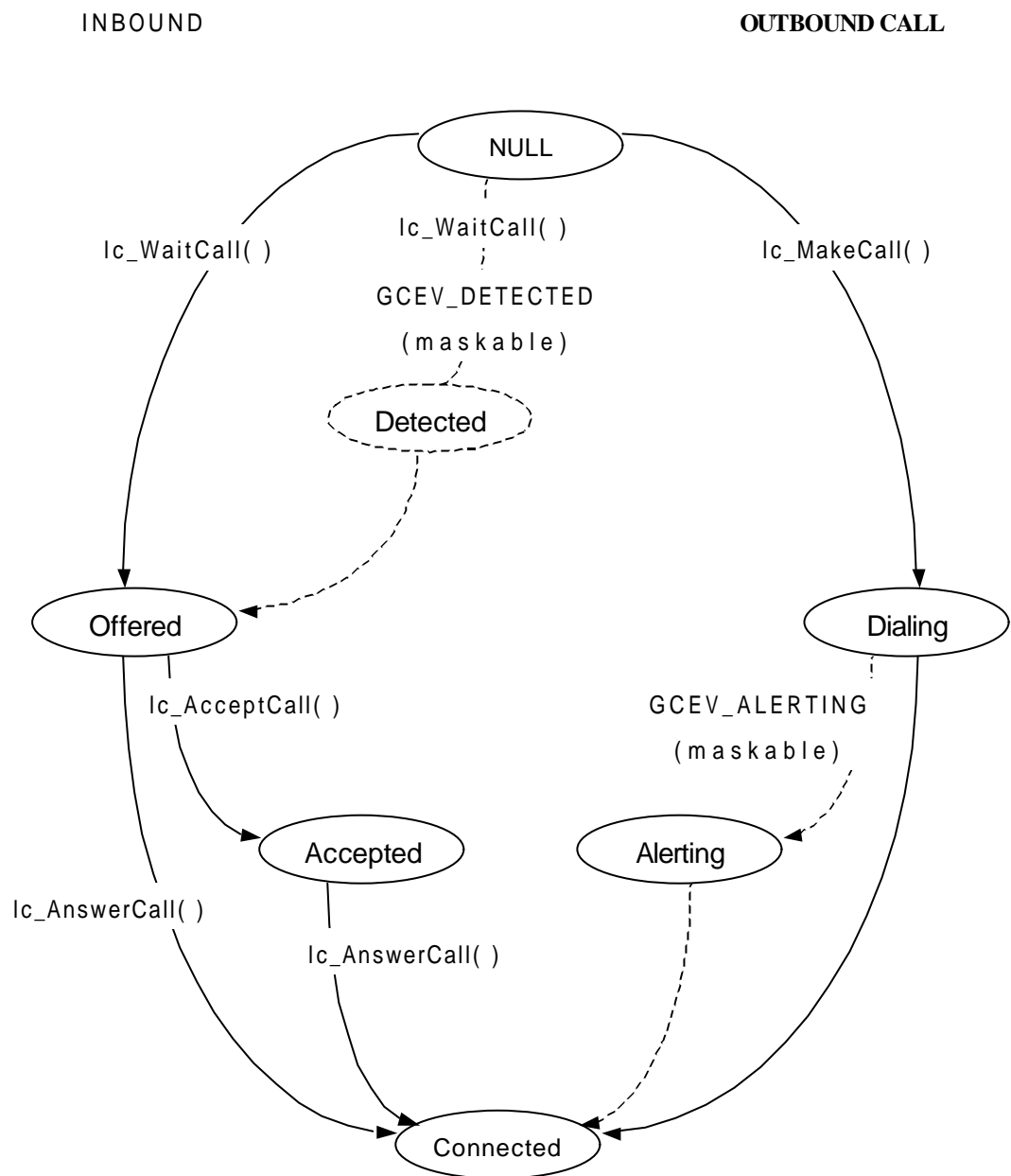


Figure 2. Basic Synchronous Call Establishment state Diagram

Inbound Call

가 Application NULL State Ic_WaitCall()
 Ic_WaitCall() timeout
 timeout , Application time out error
 , timeout 0 call line
 device .

lc_WaitCall()	Open line device Call .
*GCEV_DETECTED	Incomming call Application offer .
*lc_GetANI()	Caller ID .
*lc_GetDNIS()	DNIS .
*lc_AcceptCall()	call . answer .
lc_AnswerCall()	Inbound call answer . call connect 가 .
* = optional	maskable

Outbound Call

Outbound Call initiate 가 application
open line device dpm_setparm(linedev, DPMCH_DIAL_TYPE , 3)
lc_MakeCall() . lc_MakeCall() 가 call
Connect .
Called party 가 call GCEV_CONNECTED 가
call connect , call State가 Connected State
 , 가
Called party가 call 가 GCEV_ALERTING
called party가 answer network
Called party가 connection .
GCEV_ALERTING call State Alerting State .
E-1 CAS system GCEV_ALERTING remote end가 ringback
 , ISDN remote end가 Alerting
 .
Remote end가 call answer call connect . Connect
lc_MakeCall() 가 , Call State Connected State
 .
GCEV_TASKFAIL GCEV_DISCONNECTED
 . Application GCEV_TASKFAIL local 가
 , GCEV_DISCONNECTED remote end가 answer
 .

Table 16. Outbound Call Set-Up ()

Function / Event	Action / Description
dpm_setparm()	Digital Network Interface device

	.
*Ic_SetEvtMsk()	Maskable Enable Disable .
Ic_MakeCall()	CRN . Connect GCEV_CONNECTED GCEV_TASKFAIL
*GCEV_ALERTING	Called party call answer
* = optional	maskable

Call Termination

call (termination) (teardown)

Call State .

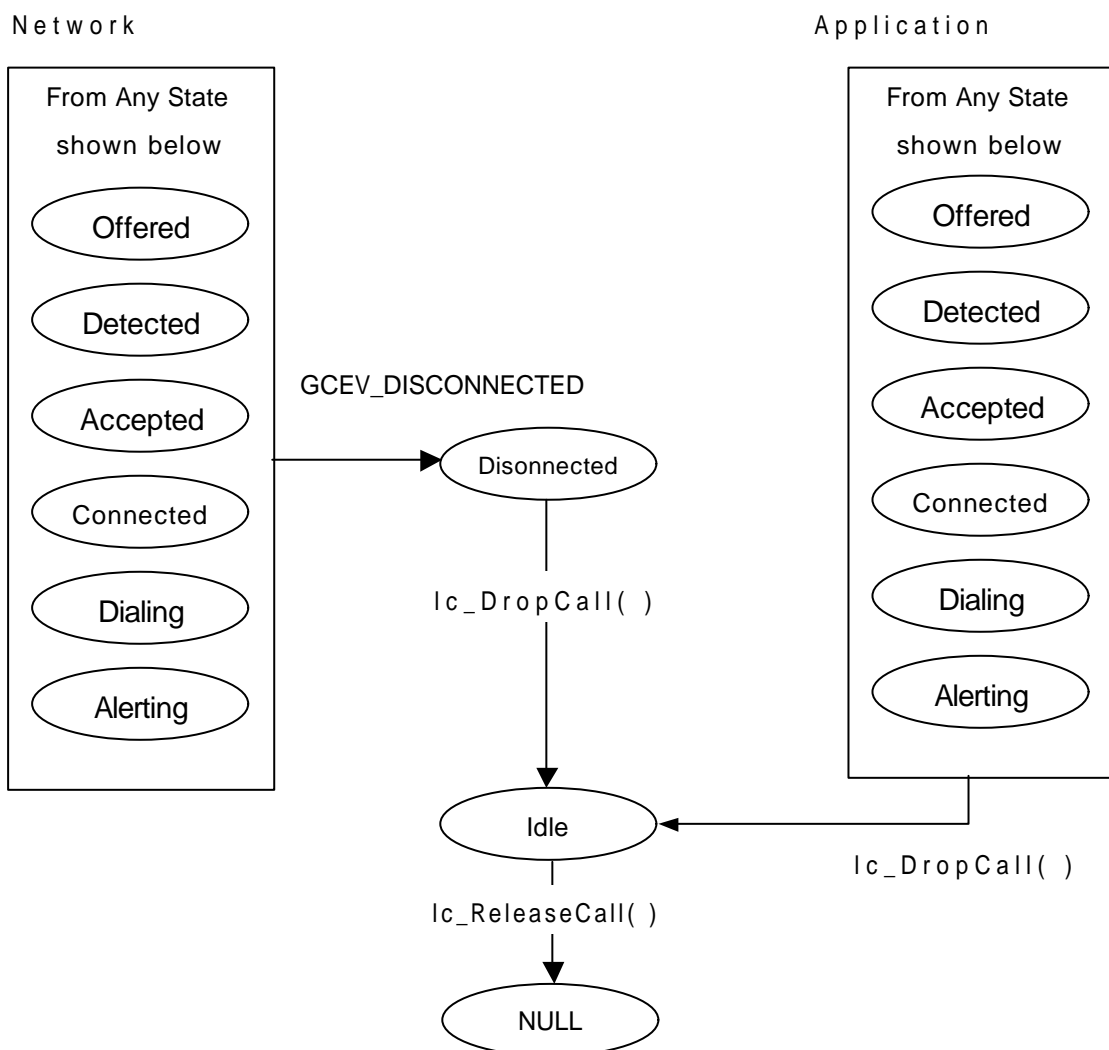


Figure 13. Asynchronous Call Tear-Down State Diagram

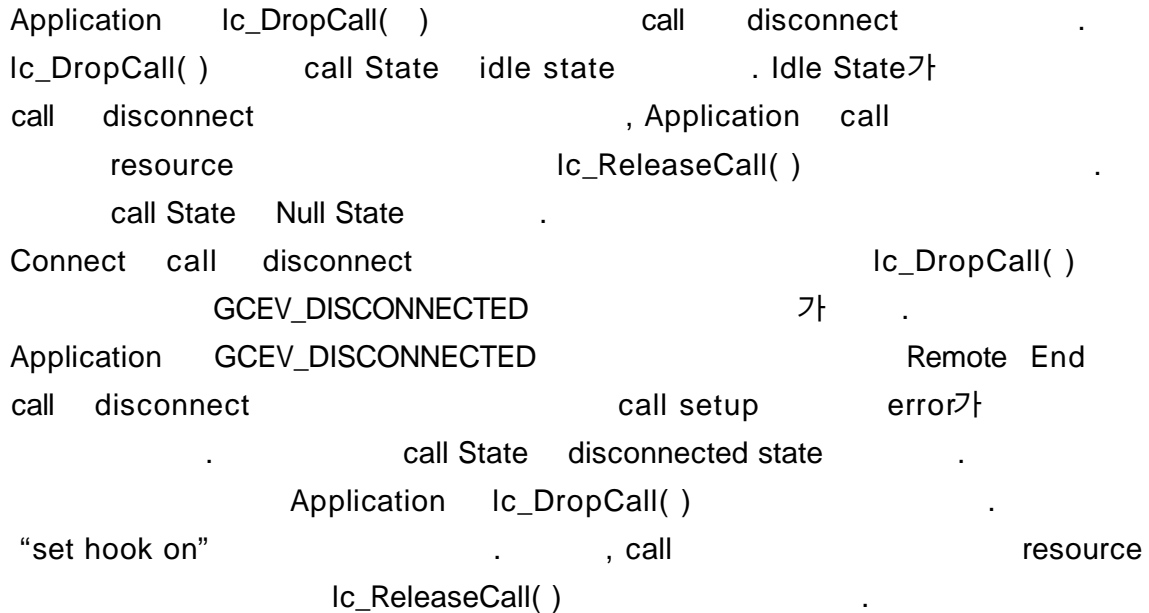


Table 17. Call Termination ()

Function / Event	Action / Description
GCEV_DISCONNECTED	Network call termination Call State disconnected State
lc_DropCall()	Call disconnect
lc_ReleaseCall()	call resource call State NULL State , Network port call initiate
* = optional maskable	

4.4 Routing

lc_Open() 가 line device ,
network voice . lc_GetNetworkH()
lc_GetVoiceH() 가 , lc_Open() 가
network device voice device Full Duplex routing .

4.5 Event Handling

LogicalCall protocol

network monitoring .

LogicalCall 가 .

1. unsolicited Network signal ,

2. termination -

Ic_SetEvtMsk() enable disable .

GCEV_ALERTING default enable

GCEV_DETECTED default enable

LogicalCall Ic_GetMetaEvent()

Ic_GetMetaEventEx() 가 . METAEVENT

CRN, line device 가 . Non-LogicalCall

device , type, data pointer length

METAEVENT .

linedev field가 line device

, crn filed가 CRN . crn field가

0 CRN 가 .

4.6 Event Definitions

section LogicalCall .

- Inbound call
- Outbound call
- Disconnect / Failure

Event

Terminates - , .

Reference - 가CRN LDID .

Unsolicited .

Table 14. Inbound Call

Event	Terminates	Reference	
GCEV_ACCEPT	lc_AcceptCall()	CRN	Remote end call answer
GCEV_ANSWERED	lc_AnswerCall()	CRN	Call Connect
GCEV_DETECTED	Unsolicited	CRN	Incomming call call application offer digit
GCEV_OFFERED	Unsolicited	CRN	Inbound call , call Offered State

Table 15. Outbound Call

Event	Terminates	Reference	
GCEV_ALERTING	Unsolicited (enabled by default)	CRN	Called party ring , CO ringback tone
GCEV_CONNECTED	lc_MakeCall()	CRN	Call connect

Table 18. Disconnected / Failed Call

Event	Terminates	Reference	
GCEV_DROPCALL	Lc_DropCall()	CRN	Call Disconnect Call State Idle State
GCEV_DISCONNECTED	Unsolicited	CRN	Remote end disconnect

GCEV_RESETLINEDEV	Lc_ResetLineDev()	LDID	Line device call disconnect .
-------------------	--------------------	------	-------------------------------

Table 19. Other GlobalCall

Event	Teriminates	Reference	
GCEV_BLOCKED	Unsolicited (enabled by default)	LDID	Line block , application call Block Ic_ResultValue() Ic_ResultMsg() 가 .
GCEV_UNBLOCKED	Unsolicited (enabled by default)	LDID	Line unblock , application call unblock Ic_ResultValue() Ic_ResultMsg() 가 .
GCEV_TASKFAIL	Unsolicited (enabled by default)	LDID	Logical Call error 가 Call state . Error가 Ic_ErrorValue() Ic_ResultValue() 가 .
GCEV_SETCHANSTATE	Ic_SetChanState()	LDID	Line device state가 Ic_SetChanState()

lc_AnswerCall() equivalent to conventional “set hook off” function

Name	int lc_AnswerCall(crn, rings, mode)	
Inputs	CRN crn :	Call reference number
	int rings :	return
	unsigned long mode :	async or sync
Returns	0	
	-1	
Includes	SmartSDK.h	
Category	Basic call control	
Mode	Synchronous, Asynchronous	

answering inbound call call
"set hook off"

■ Termination Events

- GCEV_ANSWERED –

■ Dialogic

Parameter

Crn : call reference number

Rings : return

- 0-14 rings : rings
- 15 rings : ring

Mode : EV_ASYNC –

EV_SYNC –

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "SmartSdk.h"
#include "gcerr.h"
/*
 * Assume the following has been done:
 * 1. Opened line devices for each time slot on dpmB1C1.
 * 2. Wait for a call using lc_WaitCall()
 * 3. An event has arrived and has been converted to a metaevent
 *    using lc_GetMetaEvent() or lc_GetMetaEventEx() (Windows 2000)
 * 4. The event is determined to be a GCEV_OFFERED event
 */
int answer_call(void)
{
    CRN crn; /* call reference number */
    int lc_error; /* LogicalCall Error */
    char *msg; /* pointer to error message string */
    /*
     * Do the following:
     * 1. Get the CRN from the metaevent
     * 2. Proceed to answer the call as shown below
     */
    crn = metaevent.crn;
    /*
     * Answer the incoming call
     */
    if (lc_AnswerCall(crn, 0, EV_ASYNC) < 0) {
        /* process error return as shown */
        lc_ErrorValue( &metaevent, &lc_error);
        lc_ResultMsg( (long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
            metaevent.evtdev, lc_error, msg);
        return(lc_error);
    }
}
```



```

/*
 * lc_AnswerCall() terminates with GCEV_ANSWERED event
 */
return (0);
}

```

■ Error

lc_ResultMsg() 가 0 code message 가 lc_ErrorValue() .

lc_DropCall()

disconnects a call

Name	int lc_DropCall(crn, cause, mode)		
Inputs	CRN crn :	Call reference number	
	int cause :	Call drop	
	unsigned long mode :	async or sync	
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	Basic call control		
Mode	Synchronous, Asynchronous		

■

connect call call setup call disconnect . ,
call state (Accepted, Alerting, Connected, Detected, or Offered) idle
state .

■ Termination Events

- GCEV_DROP_CALL - gc_DropCall() 가 .
- GCEV_TASKFAIL 가 . Error Handling section .

■ Cautions

lc_DropCall() CRN release . , lc_DropCall()
lc_ReleaseCall() . vpm_play()
vpm_rec() lc_DropCall() vpm_stopch()

■ Dialogic

■ Parameter .

Crn : call reference number

Cause :call drop

Mode : EV_ASYNC –
EV_SYNC –

Table 17. lc_DropCall() causes

Cause	
GC_UNASSIGNED_NUMBER	Request Number is unknown
GC_NORMAL_CLEARING	Call dropped under normal conditions
GC_USER_BUSY	End user is busy
GC_DEST_OUT_OF_ORDER	Destination is out of order
GC_NETWORK_CONGESTION	Call dropped due to traffic volume on network

■ **Example**

```
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"
/*
* Assume the following has been done:
* 1. Opened line devices for each time slot on dpmB1C1.
* 2. Wait for a call using lc_WaitCall()
* 3. The application has chosen to terminate the call
*OR
* the unsolicited event GCEV_DISCONNECTED has arrived
* Note: A call may be dropped from any state other than IDLE or NULL
*/
int drop_call(CRN crn)
{
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    if (lc_DropCall(crn, GC_NORMAL_CLEARING, EV_ASYNC) < 0 )
    {
```



```

/* process error return as shown */
lc_ErrorValue(&metaevent, &lc_error);
lc_ResultMsg( (long) lc_error, &msg);
printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
metaevent.evtdev, lc_error, msg);
return(lc_error);
}
/*
lc_DropCall() is terminated by the GCEV_DROP_CALL event.
* Application must then release the call using lc_ReleaseCall().
*/
return (0);
}

```

■ Error

lc_ErrorValue() 가 0 이면 성공, 그 외의 값은 에러를 나타냅니다. 에러 코드는 lc_ResultMsg()의 code 필드에, 에러 메시지는 message 필드에 저장됩니다.

lc_MakeCall() enables the application to make an outgoing call

Name	int lc_MakeCall(linedev, crnp, numberstr, timeout, mode)
Inputs	LINEDEV linedev : line device CRN crn : Call reference number char *numberstr : int timeout : time-out (: second) unsigned long mode : async or sync
Returns	0 -1
Includes	SmartSDK.h
Category	Basic call control
Mode	Synchronous, Asynchronous



outgoing call .

Call State NULL dialing State , dialing

State Alerting State . call Answer Call State

Dialing State Connected State .

Asynchronous GCEV_CALLSTATUS ,

timeout value remote answer .

lc_DropCall(), lc_ReleaseCall() lc_MakeCall()

가 . remote가 busy

GCEV_DISCONNECTED . 가 lc_DropCall()

lc_ReleaseCall() lc_MakeCall() 가 .

Remote answer GCEV_CONNECTED .

GCEV_CONNECTED GCEV_DISCONNECTED

GCEV_CALLSTATUS lc_ResultSet() lc_ResultSetMsg()

call connect .

GCEV_TASKFAIL lc_ResetLineDev() ,

`Ic_ErrorValue()` `Ic_ResultMsg()` Error가
 .
 Synchronous Connect 0 , connect
 1 . Connect `Ic_ErrorValue()`
`Ic_ResultMsg()` . Asynchronous
 가 `Ic_MakeCall()` `Ic_DropCall()`
`Ic_ReleaseCall()` .

Table 22. Call Conditions and Results

Condition	Event/Return Value	Result/Error Value
Call answered at remote end	Async : GCEV_CONNECTED Sync : 0	Connected and called party Answer
Error occurs prior to dialing	Async : GCEV_TASKFAIL Sync < 0	<code>Ic_ErrorValue()</code> , <code>Ic_ResultMsg()</code> .
Error occurs during dialing	Async : GCEV_DISCONNECTED Sync < 0	Time Out
Busy line	Async : GCEV_DISCONNECTED Sync < 0	Busy <code>Ic_ResultValue()</code> : GCRV_BUSY
Ring, no answer	Async : GCEV_CALLSTATUS Sync < 0	No Answer or Time Out <code>Ic_ResultValue()</code> : GCRV_NOANSWER

■ Termination Events

`GCEV_CONNECTED` , Connect
 .

1

```
int value = 3;
dpm_setparm(nDev, DPMCH_DIAL_TYPE, &Value)
```

```
typedef struct {
    GCLIB_MAKECALL_BLK *gclib;
    void *cclib;
} GC_MAKECALL_BLK, *GC_MAKECALL_BLKP;
```

Dialogic	GC MAKECALL BLK	NULL	assign	default
----------	-----------------	------	--------	---------

Dialogic	gc_MakeCall()	SYNC	connect
	가 -1	.	-1
CRN	0 gc_DropCall()	gc_ReleaseCall()	gc_MakeCall()
	, lc_MakeCall()	lc_DropCall()	lc_ReleasaeCall()
	lc_MakeCall()	.	

Linedev : line device handle.

Crnp :call reference number가 pointer.

Numberstr : call (null terminated string). Maximum length : 32 digit

Timeout : time interval call established , 가
timeout error .
0 .

Mode : EV_ASYNC –
 EV_SYNC –

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"
#define MAXCHAN 30 /* max. number of channels in system */
/*
 * Data structure which stores all information for each line
 */
struct linebag {
    LINEDEV ldev; /* LogicalCall line device handle */
    CRN crn; /* LogicalCall API call handle */
    int state; /* state of first layer state machine */
} port[MAXCHAN+1];
struct linebag *pline; /* pointer to access line device */
/*
 * Assume the following has been done:
 * 1. Opened line devices for each time slot on dpmB1C1.
 * 2. Each line device is stored in linebag structure "port"
 * 3. dpm_setparm(nde, DPMCH_DIAL_TYPE, &Value) : Value = 3;
 */
int make_call(int port_num)
{
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    /* Find info for this time slot, specified by 'port_num' */
    /* (assumes port_num is valid) */
    pline = port + port_num;
    /*
     * Make a call to the number 1234.
     */
    if ( lc_MakeCall(pline->ldev, &pline->crn, "1234", 0, EV_SYNC)
```



```

== 0) {
/* Call successfully connected; continue processing */
}
else {
/* process error return as shown */
lc_ErrorValue(&metaevent, &lc_error);
lc_ResultMsg( (long) lc_error, &msg);
printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
pline -> ldev, lc_error, msg);
return(lc_error);
}
/*
* Application may now wait for an event to indicate call
* completion.
*/
return (0);
}

```

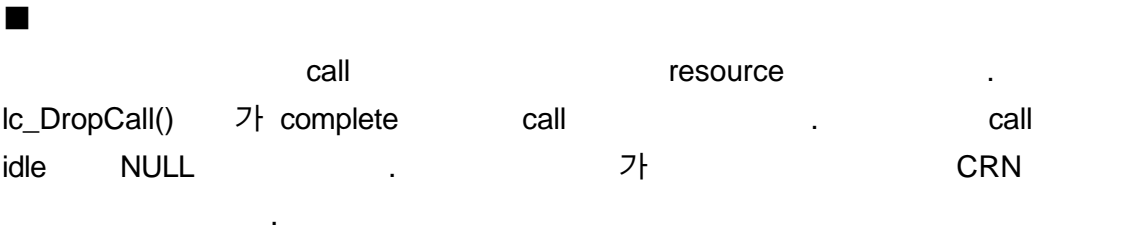
■ Error

lc_ErrorValue() 가 0 이면 성공, 0 이외의 값이면 실패.
 lc_ResultMsg() code message 가 .

lc_ReleaseCall()

releases all internal resources

Name	int lc_ReleaseCall(crn)	
Inputs	CRN crn :	Call reference number
Returns	0	
	-1	
Includes	SmartSDK.h	
Category	Basic call control	
Mode	Synchronous	



■ Termination Events

■ Dialogic

■

Parameter

Crn : call reference number

■ Example

```
#include <windows.h>
#include " SmartSDK.h"
#include " gcerr.h"
```



```

/*
* Assume the following has been done:
* 1. Opened line devices for each time slot on dpmB1C1.
* 2. Wait for a call using lc_WaitCall()
* 3. An event has arrived and has been converted to a metaevent
* using lc_GetMetaEvent() or lc_GetMetaEventEx() (Windows 2000)
* 4. The call has been dropped with lc_DropCall()
*/
int release_call(CRN crn)
{
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    /*
    * Release the system resources using lc_ReleaseCall().
    */
    if (lc_ReleaseCall(crn) < 0 ) {
        /* process error return as shown */
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg( (long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
            metaevent.evtdev, lc_error, msg);
        return(lc_error);
    }
    /*
    * Once lc_ReleaseCall() returns, system is now ready to generate
    * or accept another call on this line device.
    */
    return (0);
}

```


lc_WaitCall()	sets up conditions for processing inbound calls
-----------------------	--

lc_WaitCall()	sets up conditions for processing inbound calls
-----------------------	--

Name	int lc_WaitCall(linedev, crnp, timeout, mode)
Inputs	LINEDEV linedev : line device CRN *crnp : Call reference number int timeout : time-out (: second) unsigned long mode : async or sync
Returns	0 -1
Includes	SmartSDK.h
Category	Basic call control
Mode	Synchronous, Asynchronous


```

graph TD
    Inbound[inbound call] --> Unblock[unblock]
    Unblock --> Incoming[incomming call]
    Incoming --> GCEV[GCEV_OFFERED]
    GCEV --> CRN1[CRN]
    CRN1 --> GCEV2[GCEV_OFFERED]
    GCEV2 --> CRN2[CRN]
    CRN2 --> Meta[METAEVENT struct]
    Meta --> CrnField[crn field]
    CrnField --> Timeout[timeout value]
    Timeout --> Null[NULL]
    Null --> CRN3[CRN]
    CRN3 --> Assign[assign]
    Assign --> Null2[NULL]
    Null2 --> State[State]
    State --> Offered[OFFERED State]
    Offered --> Setup[setup]
    Setup --> TimeSlot[time slot]

```

■ Termination Events

GCEV_OFFERED – incoming call arrive

■ Dialogic

Dialogic	GC_WAITCALL_BLK	.	Dialogic	DNA3.3
lc_WaitCall()		.		NULL
	reserved	.		



Parameter

Linedev : line device handle

Crnp :CRN pointer.

Timeout : incomming call (: second)

Timeout expire -1 call state NULL

mode :EV_ASYNC –

EV_SYNC -

■ Example

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

```
#include " SmartSDK.h"
```

```
#include " gcerr.h"
```

```
#define MAXCHAN 30 /* max. number of channels in system */
```

```
/*
```

```
* Data structure which stores all information for each line
```

```
*/
```

```
struct linebag {
```

```
LINEDEV ldev; /* line device handle */
```

```
CRN crn; /* API call handle */
```

```
int state; /* state of first layer state machine */
```

```
} port[MAXCHAN+1];
```

```
struct linebag *pline; /* pointer to access line device */
```

```
/*
```

```
* Assume the following has been done:
```

```
* 1. Open line devices for each time slot on dpmB1C1.
```

```
* 2. Each Line Device ID is stored in linebag structure, 'port'.
```

```
*/
```

```
int wait_call(int port_num)
```

```
{
```



```

int lc_error; /* LogicalCall error code */
lchar *msg; /* points to the error message string */
/* Find info for this time slot, specified by 'port_num' */
pline = port + port_num;
/*
* Wait for a call, with 0 timeout.
*/
if (pline->state == GCST_NULL) {
    if (lc_WaitCall(pline->ldev, NULL, 0, EV_ASYNC) < 0) {
        /* process error return as shown */
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg((long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
            pline -> ldev, lc_error, msg);
        return(lc_error);
    }
}
/*
* GCEV_OFFERED event will indicate incoming call has arrived.
*/
return (0);
}

```


lc_AcceptCall()

optional response to an inbound call

Name	int lc_AcceptCall(crn, rings, mode)		
Inputs	CRN crn :	Call reference number	
	int rings :	return	ring
	unsigned long mode :	async or sync	
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	optional feature		
Mode	Synchronous, Asynchronous		

■

inbound call optional response , call

answer . GCEV_OFFERED 가 ,

 lc_WaitCall() 가 .

Application . incomming call

answer Application 가 .

 . GCEV_ACCEPT

가 , 0 . Call State OFFERED state

ACCEPT state .

■ Termination Events

GCEV_ACCEPT .

GCEV_TASKFAIL .

■ Dialogic

 .

■	Parameter	.
---	-----------	---

Crn : call reference number

Rings :return ring .

•0-14 rings : call ring accept .
•15 rings : answer ring , 15
ring Answer .

Mode : EV_ASYNC :

EV_SYNC :

■ Example

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

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

```
#include "SmartSDK.h"
```

```
#include "gcerr.h"
```

```
/*
```

```
* Assume the following has been done:
```

```
* 1. Opened line devices for each time slot on dpmB1C1.
```

```
* 2. Wait for a call using lc_WaitCall()
```

```
* 3. An event has arrived and has been converted to a metaevent
```

```
* using lc_GetMetaEvent() or lc_GetMetaEventEx() (Windows 2000)
```

```
* 4. The event is determined to be a GCEV_OFFERED event
```

```
*/
```

```
int accept_call(void)
```

```
{
```

```
    CRN crn; /* Call Reference Number */
```

```
    int lc_error; /* LogicalCall error code */
```

```
    char *msg; /* pointer to error message string */
```

```
    /*
```

```
    * Accept the incoming call.
```

```
    */
```

```
    crn = metaevent.crn;
```

```
    if (lc_AcceptCall(crn, 0, EV_ASYNC) != LC_SUCCESS) {
```

```
        lc_ErrorValue(&metaevent, &lc_error);
```

```
        lc_ResultMsg( (long) lc_error, &msg);
```

```
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
```



```
        metaevent.evtdev, lc_error, msg);
        return(lc_error);
    }
    /*
    * lc_AcceptCall() terminates with GCEV_ACCEPT event.
    * When GCEV_ACCEPT is received, the state changes to
    * Accepted and lc_AnswerCall() can be issued to complete
    * the connection.
    */
    return (0);
}
```


lc_GetANI()

returns ANI information

Name	int lc_GetANI(crn, ani_buf)			
Inputs	CRN crn :	Call reference number		
	char *ani_buf ANI :	digits		
Returns	0			
	-1			
Includes	SmartSDK.h			
Category	optional feature			
Mode	Synchronous			

■

call setup ANI return . ANI 가
가 error가 lc_GetANI() FAIL
. NULL OFFERED 가 ANI 가 .

■ Termination Events

.

■ Dialogic

.

■	Parameter	.
<hr/>		
Crn	:Call Reference Number	
Ani_buf	:ANI가 load	buffer address.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include " SmartSDK.h"
```



```

#include "gcerr.h"
/*
* Assume the following has been done:
* 1. Opened line devices for each time slot on dpmB1C1.
* 2. Wait for a call using lc_WaitCall()
* 3. An event has arrived and has been converted to a metaevent
* using lc_GetMetaEvent() or lc_GetMetaEventEx() (Windows 2000)
* 4. The event is determined to be a GCEV_OFFERED event
*/
int get_ani(void)
{
    CRN crn; /* call reference number */
    char ani_buf[LC_ADDRSIZE]; /* Buffer for ANI digits */
    liint lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    /*
    * Get the calling party number
    */
    crn = metaevent.crn;
    if (lc_GetANI(crn, ani_buf) < 0) {
        /* process error return as shown */
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg( (long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
            metaevent.evtdev, lc_error, msg);
        return(lc_error);
    }
    /* Application can answer, accept, or terminate the call at this
    * point, based on the ANI information. */
    return (0);
}

```


lc_GetDNIS()

gets the DNIS information

Name	int lc_GetDNIS(crn, dnis_buf)	
Inputs	CRN crn :	Call reference number
	Char *dnis_buf :	DNIS info
Returns	0	
	-1	
Includes	SmartSDK.h	
Category	optional feature	
Mode	Synchronous	



CRN DNIS . DDI digit ASCII '\0'

■ Termination Events

■ Dialogic



Parameter

crn: Call Reference Number

dnis_buf: DNIS가 address

■ Example

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

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

```
#include "SmartSDK.h"
```



```

#include "gcerr.h"
/*
* Assume the following has been done:
* 1. 'maxddi' has been setup depending on needs
* of application/protocol.
* 2. Line devices have been opened for each time slot on dpmB1C1.
* 3. Wait for a call using lc_WaitCall()
* 4. An event has arrived and has been converted to a metaevent
* using lc_GetMetaEvent() or lc_GetMetaEventEx() (Windows 2000)
* 5. The event is determined to be a GCEV_OFFERED event
*/
int get_dnis(void)
{
    CRN crn; /* call reference number */
    int maxddi = 10; /* maximum allowable DDI digits */
    char dnis_buf[LC_ADDRSIZE]; /* DNIS digit Buffer */
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    /* 1st get the crn */
    if (lc_GetCRN(&crn, &metaevent) < 0) {
        /* handle the lc_GetCRN error */
    }
    /*
    * Get called party number and check that there were not too
    * many digits collected.
    */
    if (lc_GetDNIS(crn, dnis_buf) < 0) {
        /* process error return as shown */
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg( (long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
            metaevent.evtdev, lc_error, msg);
        return(lc_error);
    }
    if (strlen(dnis_buf) <= maxddi) {
        /*

```



```

    Process called party number as needed by the application.
    */
    } else {
    /*
    * Drop the call if number of DDI digits exceeds maximum limit
    */
    if (lc_DropCall(crn, GC_NORMAL_CLEARING, EV_ASYNC) < 0) {
    /* process error return from lc_DropCall() */
    }
    }
    /*
    * Application can answer, accept, or terminate the call at this
    * point, based on the DNIS information.
    */
    return (0);
}

```


Ic_GetVer()

gets the version number

Name	int Ic_GetVer(linedev, releasenum, intnum)		
Inputs	LINEDEV linedev :	line device handle	
	Unsigned int *releasenum	production release number가	pointer
	Unsigned int *intump	release number가	pointer
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	optional feature		
Mode	Synchronous		



LC Library 가 .

■ Termination Events

.

■ Dialogic

Dialogic component number 가
software component 가 . component

GCGV_LIB - LogicalCall library

ICGV_LIB - ICAPI library

ANGV_LIB - ANAPI library

PDGV_LIB - PDKRT library



Parameter .

linedev : LogicalCall line devjce handle. parameter가 0

LogicalCall API version number .

Releasenum	: production release number	가	pointer
Intnum	: release number	가	pointer

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"

int print_version(LINEDEV ldev, long component)
{
    unsigned int releasenum; /* Production release number */
    unsigned int intnum; /* Internal release number */
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    /*
     * Get the version number of the library associate with the line
     * device.
     */
    if (lc_GetVer(ldev, &releasenum, &intnum) == 0) {
        printf("Production release number = 0x%lx\n", releasenum);
        printf("Internal release number = 0x%lx\n", intnum);
    }
    else {
        /* process error return as shown */
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg((long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
            ldev, lc_error, msg);
        return(lc_error);
    }
    return (0);
}
```


lc_SetBilling()

sets billing information for the call

Name	int lc_SetBilling(crn, rate_type, ratep)		
Inputs	CRN crn :	Call reference number	
	Int rete_type	billing data	type
	GC_RATE_U *ratep	call charge rate	pointer
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	optional feature		
Mode	Synchronous		



CRN call billing .

■ Termination Events

.

■ Dialogic

Dialogic DNA SDK 3.3

가 synchronous, Asynchronous

mode 가 . lc_SetBilling()

.



Parameter

.

crn : Call Reference Number

rate_type : billing data type

GCR_CHARGE : 가 (default).

GCR_NOCHARGE : 가 .

ratep : GC_RATE_U pointer. call .

■ Example

```
/*
* Assume the following has been done:
* 1. device has been opened
* 2. lc_WaitCall() has been issued to wait for a call.
* 3. lc_GetMetaEvent() or lc_GetMetaEventEx() (Windows 2000) has been
* called to convert the event into metaevent.
* 4. GCEV_OFFERED has been detected.
* 5. a call has been established.
*/
#include <windows.h>
#include "SmartSDK.h"
#include "gcerr.h"
/*
* For this example, let's assume that mode = SYNC and
* the rate_type = GCR_CHARGE. The ratep stores the billing information.
* rate_type can be one of the following:
*
* Note: This is only available for some protocols.
This function call is used any time after the connection
* is established.
*/
int set_billing(CRN crn, int rate_type, GC_RATE_U *ratep)
{
    LINEDEV ldev; /* Line device */
    int lc_err; /* LogicalCall Error Code */
    char *msg; /* Error Message */
    if(lc_CRN2LineDev(crn, &ldev) < 0 ) {
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg((long) lc_error, &msg);
        printf ("Error: lc_CRN2LineDev ErrorValue: %d - %s\n",
            lc_error, msg);
        return(lc_error);
    }
}
```



```
if(lc_SetBilling(crn, rate_type, ratep) < 0) {  
    lc_ErrorValue(&metaevent, &lc_error);  
    lc_ResultMsg((long) lc_error, &msg);  
    printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",  
        ldev, lc_err, msg);  
    return(lc_err);  
}  
return(0);  
}
```


lc_SetCallingNum()

sets the default calling party number

Name	int lc_SetCallingNum(linedev, calling_num)		
Inputs	LINEDEV linedev :	line device handle	
	Char *calling_num	calling party phone number	
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	optional feature		
Mode	Synchronous		

■

	line device	calling party number	default
.	Calling party number	null terminated string	.
Number	called party	ANI number	.

■ **Termination Events**

.

■ **Dialogic**

.

■

	Parameter	.
--	-----------	---

linedev : LogicalCall line device handle.

calling_num : calling party phone number.(null terminated string)

■ **Example**

```
#include <windows.h>
#include " SmartSDK.h"
#include " gcerr.h"
```



```

int set_calling_num(LINEDEV ldev)
{
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    /* Set up the calling party number on the line device */
    if (lc_SetCallingNum(ldev, "2019933000") < 0) {
        /* process error return as shown */
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg((long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
            ldev, lc_error, msg);
        return(lc_error);
    }
    /*
    * Application can proceed to make a call, and the calling party
    * number will be as set above. It may be changed later, if
    * necessary.
    */
    return (0);
}

```


lc_SetChanState()

changes the maintenance state

Name	int lc_SetChanState(linedev, chanstate, mode)
Inputs	LINEDEV linedev : line device handle unsigned long mode : async or sync
Returns	0 -1
Includes	SmartSDK.h
Category	optional feature
Mode	Synchronous, Asynchronous

line device maintenance state , Inbound call
Outbound call 가 Inbound call 가 , 가
. Default In service .

■ **Termination Events**

.

■ **Dialogic**

.

■ Parameter .

line : LogicalCall line device handle.

chanstate : line service .

- GCLS_INSERTSERVICE : Inbound Outbound call 가 .
- GCLS_MAINTENANCE : outbound call 가 . Inbound call 가 .
- GCLS_OUT_OF_SERVICE : Inbound Outbound call 가 .

mode : EV_ASYNC :

EV_SYNC :

■ Example

```
#include <windows.h>
#include "SmartSDK.h"
#include "gcerr.h"

/* Assume following was done:
 * IF not in the Null state, THEN
 * issue lc_DropCall() function (if needed) and then the
 * lc_ReleaseCall() function.
 */
int set_channel_InService(LINEDEV ldev)
{
    int state; /* State to which channel has to be set */
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    /*
     * Set channel to "INSERVICE" state
     */
    state = GCLS_INSERTSERVICE; /* constant describing channel state */
    if (lc_SetChanState(ldev, state, EV_SYNC) < 0) {
        /* process error return as shown */
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg((long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
            ldev, lc_error, msg);
        return(lc_error);
    }
    /*
     * Application can change state again when necessary.
     */
    return (0);
}
```


lc_Close()	closes a previously opened device
--------------------	--

lc_Close()	closes a previously opened device
--------------------	--

Name	int lc_Close(linedev)
Inputs	LINEDEV linedev : line device handle
Returns	0 -1
Includes	SmartSDK.h
Category	system control and tools
Mode	Synchronous

open device . Application access

■ Termination Events

■ Dialogic

Parameter

linedev : close Logical Call line device

■ Example

```
#include <windows.h>
#include "SmartSDK.h"
#include "gcerr.h"
#define MAXCHAN 30 /* max. number of channels in system */
/*
 * Data structure which stores all information for each line
 */
```



```

struct linebag {
    LINEDEV ldev; /* LogicalCall line device handle */
    CRN crn; /* LogicalCall API call handle */
    int state; /* state of first layer state machine */
} port[MAXCHAN+1];
struct linebag *pline; /* pointer to access line device */
int close_line_device(int port_num)
{
    LINEDEV ldev; /* LogicalCall line device handle
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    /* Find info for this time slot, specified by 'port_num' */
    /* (Assumes port_num is valid) */
    pline = port + port_num;
    ldev = pline -> ldev;
    /*
    * close the line device to remove the channel from service
    */
    if (lc_Close(ldev) < 0) {
        /* process error return as shown */
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg((long) lc_error, &msg);
        printf ("Error closing linedev 0x%lx, \"%s\"\n", ldev, msg);
        return(lc_error);
    }
    return(0);
}

```


lc_CRN2LineDev()		matches a CRN to its line device ID	
Name	int lc_CRN2LineDev (crn, linedevp)		
Inputs	CRN crn :	Call reference number	
	LINEDEV *linedevp	linedev	
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	system control and tools		
Mode	Synchronous		

■

 crn match line device 가 .

■ **Termination Events**

.

■ **Dialogic**

.

■

 Parameter

 .

crn : Call Reference Number

linedevp :output LINEDEV ID 가 . line device

 lc_Open()

 .

■ **Example**

```

#include <windows.h>
#include " SmartSDK.h"

```



```
#include "gcerr.h"
```

```
int crn_to_linedev(CRN crn, LINEDEV *ldevp)
{
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    if (lc_CRN2LineDev(crn, ldevp) < 0) {
        /* process error return as shown */
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg( (long) lc_error, &msg);
        printf ("Error on converting CRN to linedev \"%s\\n", msg);
        return(lc_error);
    }
    return(0);
}
```


lc_ErrorValue() gets an error value/failure reason code

Name	int lc_ErrorValue(metaeventp, lc_errorp)
Inputs	METAEVENT * metaeventp : METAEVENT int *lc_errorp: error code
Returns	0 -1
Includes	SmartSDK.h
Category	system control and tools
Mode	Synchronous



가 logical call 가 , Error code
가 . lc_ResultMsg() ASCII string
.

■ Termination Events

.

■ Dialogic

Dialogic DNA SDK 3.3 control library Id call control library error code
가 가 . lc_ErrorValue()
, METAEVENT error code
.



lc_errorp : LogicalCall error code 가 pointer

■ Example

#include <windows.h>


```

#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"

void print_error_values(void)
{
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    /* This could be called when any function fails;
    * to print the error values */
    if (lc_ErrorValue(&metaevent, &lc_error)< 0) {
        lc_ResultMsg((long) lc_error, &msg);
        /* LogicalCall errors will return an integer for
        the print error value. */
        printf("LogicalCall error 0x%x - %s\n", lc_error, msg);
    } else {
        printf("Could not get error value\n");
    }
}

```


lc_GetCallState() **acquires the state of the call**

Name	int lc_GetCallState(crn, state_buf)	
Inputs	CRN crn :	Call reference number
	Int *state_buf	call state가
.		
Returns	0	
	-1	
Includes	SmartSDK.h	
Category	system control and tools	
Mode	Synchronous	

■

CRN	call	.	error가
application	call	가	.

■ **Termination Events**

.

■ **Dialogic**

.

■ Parameter

Crn : Call reference number

state_buf : call state가

■ **Example**

#include <windows.h>

#include <stdio.h>

#include " SmartSDK.h"


```
#include "gcerr.h"
```

```
#define MAXCHAN 30 /* max. number of channels in system */  
/*
```

```
 * Data structure which stores all information for each line  
 */
```

```
struct linebag {
```

```
    LINEDEV ldev; /* LogicalCall line device handle */
```

```
    CRN crn; /* LogicalCall API call handle */
```

```
    int state; /* state of first layer state machine */
```

```
    } port[MAXCHAN+1];
```

```
    struct linebag *pline; /* pointer to access line device */
```

```
    int get_call_state(int port_num)
```

```
    {
```

```
        LINEDEV ldev; /* line device ID */
```

```
        CRN crn; /* call reference number */
```

```
        int call_state; /* current state of call */
```

```
        int lc_error; /* LogicalCall error code */
```

```
        char *msg; /* points to the error message string */
```

```
        /* Find info for this time slot, specified by 'port_num' */
```

```
        /* (Assumes port_num is valid) */
```

```
        pline = port + port_num;
```

```
        crn = pline->crn;
```

```
        /*
```

```
         * Retrieve the call state and save it.
```

```
        */
```

```
        if (crn) {
```

```
            if (lc_GetCallState( crn, &call_state) < 0 ) {
```

```
                /* process error return as shown */
```

```
                lc_ErrorValue(&metaevent, &lc_error);
```

```
                lc_ResultMsg( (long) lc_error, &msg);
```

```
                if (lc_CRN2LineDev( crn, &ldev) < 0) {
```

```
                    /* get and process error */
```

```
                }
```

```
                printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
```



```
        ldev, lc_error, msg);
        return(lc_error);
    }
}
pline->state = call_state;
return (0);
}
```


lc_GetCRN()		gets the CRN		
Name	int lc_GetCRN(crnnp, metaeventp)			
Inputs	CRN *crnp	return	CRN	pointer
	METAEVENT *metaeventp	metaevent block		
Returns	0			
	-1			
Includes	SmartSDK.h			
Category	system control and tools			
Mode	Synchronous			

가

line device

CRN

. metaeventp pointer

lc_GetMetaEvent()

METAEVENT struct

.

■ Termination Events

.

■ Dialogic

.

.

crnnp :call reference number가

memory address

pointer

metaeventp :lc_GetMetaEvent()

, lc_GetMetaEventEx()

METAEVENT

pointer

■ Example

```
#include <windows.h>
#include <stdio.h>
#include " SmartSDK.h"
#include " gcerr.h"
```



```

/*
* Assume the following has already been done:
* 1. Opened line devices for each time slot on dpmB1C1.
* 2. Wait for a call using lc_WaitCall()
* 3. An event has arrived and has been converted to a metaevent
* using lc_GetMetaEvent() or lc_GetMetaEventEx() (Windows 2000)
*/
CRN get_crn(METAEVENT *metaeventp)
{
    CRN crn; /* call reference number */
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    if (lc_GetCRN(&crn, metaeventp) < 0 ) {
        /* process error return as shown */
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg( (long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
            metaevent.evtdev, lc_error, msg);
        return(0);
    }
    else {
        /*
        * Else return the CRN and next issue the LogicalCall function call
        * using the CRN.
        */
        return(crn);
    }
}

```


lc_GetLineDev()		gets a line device	
<hr/>			
Name	int lc_GetLineDev(linedevp, metaeventp)		
Inputs	LINEDEV	*linedevp	return line device handle
		METAEVENT *metaevent	metaevent block
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	system control and tools		
Mode	Synchronous		
<hr/>			

■

 event queue event line device .

 가 Logical Call 가

 *linedevp 0 . METAEVENT

 linedev field .

■ **Termination Events**

 .

■ **Dialogic**

 .

■

 .

linedevp : output LINEDEV가 pointer.

metaeventp :lc_GetMetaEvent() lc_GetMetaEvent()

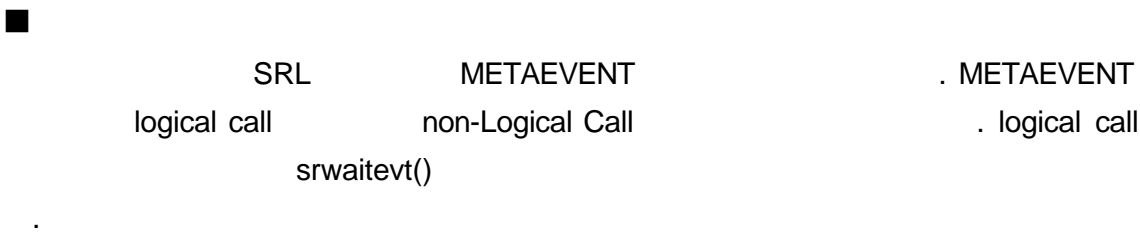
 METAEVENT pointer.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"
/*
 * Assume the following has been done:
 * 1. Opened line devices for each time slot on dpmB1C1.
 * 2. Wait for a call using lc_WaitCall()
 * 3. An event has arrived and has been converted to a metaevent
 * using lc_GetMetaEvent() or lc_GetMetaEventEx() (Windows 2000)
 * 4. The event is determined to be a GCEV_OFFERED event
 */
int get_linedev(LINEDEV *ldevp)
{
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    /*
     * Get Line Device corresponding to this event
     */
    if (lc_GetLineDev(ldevp, &metaevent) < 0) {
        /* process error return as shown */
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg((long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
            metaevent.evtdev, lc_error, msg);
        return(lc_error);
    }
    /*
     * The line device ID may then be used for functions like
     * lc_SetParm(), lc_SetUsrAttr(), and lc_GetVoiceH().
     */
    return (0);
}
```


Ic_GetMetaEvent() maps the current SRL event into a metaevent

Name	int Ic_GetMetaEvent(metaeventp)	
Inputs	METAEVENT *metaeventp	METAEVENT block
Returns	0 -1	
Includes	SmartSDK.h	
Category	system control and tools	
Mode	Synchronous	



■ **Termination Events**

■ **Dialogic**

■

metaeventp : METAEVENT

■ **Example**

```
if(sr_waitevt(timeout) != -1) { /* i.e. an event occurred */
retcode = Ic_GetMetaEvent(&metaevent);
```



```

if (retcode <0) {
/* get and process the error */
} else {
/* Continue with normal processing */
}
}
OR
handler(...)
{
retcode = lc_GetMetaEvent(&metaevent);
if (retcode <0 ) {
/* get and process the error */
} else {
/* Continue with normal processing */
}
}

code    logical call    가
event type, event data pointer, event length    , event device    sr_waitevt( )
lc_GetMetaEvent()    가
retcode = lc_GetMetaEvent(&metaevent);
if (retcode <0) {
/* get and process error */
} else {
/* Can now access SRL information for any LogicalCall or
non-LogicalCall event using: */
/* metaevent.evtdatap */
/* metaevent.evtlen */
/* metaevent.evtdev */
/* metaevent.evttype */
if (metaevent.flags & GCME_GC_EVENT) {
/* process LogicalCall event here */
} else {
/* process non-LogicalCall event here */
}
}
}

```

The following code illustrates retrieving event information from the

METAEVENT structure while making a call:

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

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

```
#include "SmartSDK.h"
```

```
#include "gcerr.h"
```

```
#define MAXCHAN 30 /* max. number of channels in system */
```

```
#define NULL_STATE 0
```

```
#define DIALING_STATE 1
```

```
#define ALERTING_STATE 2
```

```
#define CONNECTED_STATE 3
```

```
/*
```

```
* Data structure which stores all information for each line
```

```
*/
```

```
struct linebag {
```

```
LINEDEV ldev; /* network line device handle */
```

```
CRN crn; /* LogicalCall API call handle */
```

```
int state; /* state of first layer state machine */
```

```
} port[MAXCHAN+1];
```

```
struct linebag *pline; /* pointer to access line device */
```

```
/*
```

```
* Assume the following has been done:
```

```
* 1. Opened line devices for each time slot on dpmB1C1.
```

```
* 2. Application is in the NULL state
```

```
* Examples are given in ASYNC mode
```

```
* Error handling is not shown
```

```
*/
```

```
int makecall(int port_num, char *numberstr)
```

```
{
```

```
    int lc_error; /* LogicalCall error code */
```

```
    char *msg; /* points to the error message string */
```

```
    long evttype; /* type of event */
```

```
    /* Find info for this time slot, specified by 'port_num' */
```

```
    /* (Assumes port_num is valid) */
```

```
    pline = port + port_num;
```

```
    if (lc_MakeCall(pline -> ldev, &pline -> crn, numberstr, NULL, 0, EV_ASYNC) < 0) {
```



```

/* process error and return */
}
pline -> state = DIALING_STATE;
for (;;) {
    sr_waitevt(-1); /* wait forever */
    /* Get the next event */
    if (lc_GetMetaEvent(&metaevent) < 0) {
        /* process error */
    }
    evttype = metaevent.evttype;
    if (metaevent.flags & GCME_GC_EVENT) {
        /* process LogicalCall event */
        switch (pline -> state) {
        case DIALING_STATE:
            switch (evttype) {
            case GCEV_ALERTING:
                pline -> state = ALERTING_STATE;
                break;
            case GCEV_CONNECTED:
                pline -> state = CONNECTED_STATE;
                /*
                 * Can now do voice functions, etc.
                 */
                return(0); /* SUCCESS RETURN POINT */
            default:
                /* handle other events here */
                break;
            }
        break;
        case ALERTING_STATE:
            switch (evttype) {
            case GCEV_CONNECTED:
                pline -> state = CONNECTED_STATE;
                /*
                 * Can now do voice functions, etc.
                 */

```



```

        return(0); /* SUCCESS RETURN POINT */
        default:
        /* handle other events here */
        break;
    }
    break;
}
/* Process non-LogicalCall event */
}
} // end of for
} // end makeCall function

```

code	incomming call	METAEVENT
가	.	

```

#include <windows.h>
#include <stdio.h>
#include " SmartSDK.h"
#include " gcerr.h"

#define MAXCHAN 30 /* max. number of channels in system */
#define NULL_STATE 0
#define CONNECTED_STATE 3
#define OFFERED_STATE 4
#define ACCEPTED_STATE 5
/*
* Data structure which stores all information for each line
*/
struct linebag {
    LINEDEV ldev; /* network line device handle */
    CRN crn; /* Call reference number */
    int state; /* state of first layer state machine */
} port[MAXCHAN+1];
struct linebag *pline; /* pointer to access line device */
/*
* Assume the following has been done:
* 1. Opened line devices for each time slot on dpmB1C1.

```



```

* 2. Application is in the NULL state
* 3. A lc_WaitCall() has been successfully issued
*
* Examples are given in ASYNC mode
* Error handling is not shown
*
*/
int waitcall(int port_num)
{
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    long evttype; /* type of event */
    /* Find info for this time slot, specified by 'port_num' */
    /* (Assumes port_num is valid) */
    pline = port + port_num;
    for (;;) {
        sr_waitevt(-1L); /* wait forever */
        /* Get the next event */
        if (lc_GetMetaEvent(&metaevent) < 0) {
            /* process error return */
        }
        evttype = metaevent.evttype;
        if (metaevent.flags & GCME_GC_EVENT)
        {
            /* process LogicalCall event */
            switch (pline -> state) {
            case NULL_STATE:
                switch (evttype) {
                case GCEV_OFFERED:
                    pline -> state = OFFERED_STATE;
                    accept_call();
                    break;
                default:
                    /* handle other events here */
                    break;
                }
            }
        }
    }
}

```



```

        break;
    case OFFERED_STATE:
        switch (evtttype) {
            case GCEV_ACCEPT:
                pline -> state = ACCEPTED_STATE;
                answer_call();
                break;
            default:
                /* handle other events here */
                break;
        }
    break;
    case ACCEPTED_STATE:
        switch (evtttype) {
            case GCEV_ANSWERED:
                pline -> state = CONNECTED_STATE;
                /*
                 * Can now do voice functions, etc.
                 */
                return(0); /* SUCCESS RETURN POINT */
            default:
                /* handle other events here */
                break;
        }
    break;
}
} // end Logical call process
else
{
    /* Process non-LogicalCall event */
}
} // end of for
}

```


lc_GetMetaEventEx() maps an event handle's SRL event into a metaevent

Name	int lc_GetMetaEventEx(metaeventp, evt_handle)		
Inputs	METAEVENT *metaeventp	METAEVENT BLOCK	
		pointer	
	unsigned long evt_handle		SRL event handle
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	system control and tools		
Mode	Synchronous		

■ event handle SRL event metaevent . SRL
evt_handle parameter event handle .

■ **Termination Events**

.

■ **Dialogic**

.

■ Parameter .

metaeventp : fill metaevent data pointer.
evt_handle : event SRL event handle.

■ Example

```
/*
 * Do SRL event processing
 */
hdlcnt = 0;
hdlcnt[ hdlcnt++ ] = GetLogicalCallHandle();
hdlcnt[ hdlcnt++ ] = GetVoiceHandle();
/* Wait selectively for devices that belong to this thread */
rc = sr_waitevtEx( hdlcnt, hdlcnt, PollTimeout_ms, &evtHdl);
if (rc != SR_TMOUT) {
/*
 * Update
 */
rc = lc_GetMetaEventEx(&g_Metaevent, evtHdl);
if (rc < 0) {
    CheckError(rc, "lc_GetMetaEventEx");
    return -1;
}
rc = vProcessCallEvents( );
}
```


lc_GetNetworkH()

returns the network device handle

Name	int lc_GetNetworkH(linedev, networkhp)		
Inputs	LINEDEV linedev :	line device handle	
	Int *networkhp	network device handle	
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	system control and tools		
Mode	Synchronous		



line device network device handle return

.

■ Termination Events

.

■ Dialogic

.



linedev : LogicalCall line device handle.

networkhp : network device handle address.

■ Example

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

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



```

#include "SmartSDK.h"
#include "gcerr.h"
/*
* Assume the following has been done:
* 1. A line device (ldev) has been opened
* For example, 'devicename' could be "dpmB1C1".
*/
int GetNetworkHandle(LINEDEV ldev)
{
    int lc_error; /* LogicalCall error code */
    lchar *msg; /* points to the error message string */
    int networkh; /* network device handle */
    if (lc_GetNetworkH(ldev, &networkh) < 0) {
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg((long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
            ldev, lc_error, msg);
        return(-1);
    }
    return (networkh);
}

```


lc_GetUsrAttr()

retrieves the attribute

Name	int lc_GetUsrAttr(linedev, usr_attrp)
Inputs	LINEDEV linedev : line device handle Void **usr_attrp: attribute info가
Returns	0 -1
Includes	SmartSDK.h
Category	system control and tools
Mode	Synchronous

■ lc_SetUsrAttr() attribute data , lc_Open()
attribute data 가 .

■ Termination Events

.

■ Dialogic

.

■ Parameter .

linedev : LogicalCall line device handle
usr_attrp : attribute 가 return address.

■ Example


```

#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"

#define MAXCHAN 30 /* max. number of channels in system */
/*
 * Data structure which stores all information for each line
 */
struct linebag {
    LINEDEV ldev; /* LogicalCall line device handle */
    CRN crn; /* LogicalCall API call handle */
    int state; /* state of first layer state machine */
} port[MAXCHAN+1];
/*
 * Retrieves port_num that was set for this device
 */
int get_usrattr(LINEDEV ldev, int *port_num)
{
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    void *vattrp; /* to retrieve the attribute */
    /*
     * Assuming that a line device is opened already and
     * that its ID is ldev, let us retrieve the attribute set
     * for this ldev, previously set by the user using lc_SetUsrAttr()
     */
    if ( lc_GetUsrAttr( ldev, &vattrp) < 0 ) {
        /* process error return as shown */
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg((long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
            ldev, lc_error, msg);
        return(lc_error);
    }
    *port_num = (int) vattrp;
}

```



```
    /*  
    * Processing may continue using this retrieved attribute  
    */  
    return (0);  
}
```


lc_GetXmitSlot() returns the network SCbus time slot number

Name	Int lc_GetXmitSlot(linedev, sctsinfop)		
Inputs	LINEDEV linedev :	line device handle	
	SC_TSINFO *sctsinfop	Scbus time slot pointer	strusture
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	system control and tools		
Mode	Synchronous		

line device TX network SCbus time slot number return
. SCbus time slot information SC_TSINFO return device
transmit channel network SCbus time slot number .

■ **Termination Events**

.

■ **Dialogic**

.

■ Parameter .

linedev : LogicalCall line device handle.

sc_tsinfop : SCbus time slot pointer.

■ **Example**

#include <windows.h>


```

#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"

#define MAX_CHAN 30 /* max. number of channels in system */
/*
 * Data structure which stores all information for each line
 */
static struct linebag {
    LINEDEV ldev; /* LogicalCall API line device handle */
    CRN crn; /* LogicalCall API call handle */
    int blocked; /* channel blocked/unblocked */
    int networkh; /* network handle */
} port[MAX_CHAN+1];
struct linebag *pline; /* pointer to access line device */
/*
 * Assume the following has been done:
 * 1. Opened line devices for each time slot on dpmB1C1.
 * 2. Each line device and voice handle is stored in
 * linebag structure "port"
 */
int call_getxmitslot(int port_num)
{
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    SC_TSINFO sc_tsinfo; /* SCBus timeslot structure */
    long scts;
    /* Find info for this time slot, specified by 'port_num' */
    /* (assumes port_num is valid) */
    pline = port + port_num;
    /* Fill in the SCbus time slot information */
    sc_tsinfo.sc_numts = 1;
    sc_tsinfo.sc_tsarrayp = &scts;
    /* Get SCbus time slot connected to transmit of
     * digital channel on board .1 */
    if (lc_GetXmitSlot(pline->ldev, &sc_tsinfo) < 0)

```



```
{
    /* process error return as shown */
    lc_ErrorValue(&metaevent, &lc_error);
    lc_ResultMsg((long) lc_error, &msg);
    printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
    pline -> ldev, lc_error, msg);
    return(lc_error);
}
return (0);
}
```


lc_Listen() connects a channel to a network SCbus time slot

Name	int lc_Listen(linedev, sc_tsinfo)		
Inputs	LINEDEV linedev :	line device handle	
	SC_TSINFO *sc_tsinfo	Scbus time slot pointer	structure
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	system controls and tools		
Mode	Synchronous		

■

RX SC_TSINFO time slot network device

■ Termination Events

■ Dialogic

Dialogic DNA SDK 3.3 synchronous, Asynchronous
mode 가 , lc_Listen() mode

■	Parameter	.
linedev	LogicalCall line device handle.	
sc_tsinfo	SCbus time slot pointer	

■ Example

#include <windows.h>


```

#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"

#define MAX_CHAN 30 /* max. number of channels in system */
/*
 * Data structure which stores all information for each line
 */
static struct linebag {
    LINEDEV ldev; /* LogicalCall API line device handle */
    CRN crn; /* LogicalCall API call handle */
    int blocked; /* channel blocked/unblocked */
    int networkh; /* network handle */
    int chdev; /* voice handle */
} port[MAX_CHAN+1];
struct linebag *pline; /* pointer to access line device */
/*
 * Assume the following has been done:
 * 1. Opened line devices for each time slot on dpmB1C1.
 * 2. Each line device and voice handle is stored in linebag structure "port"
 */
int call_listen(int port_num)
{
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    SC_TSINFO sc_tsinfo; /* SCbus time slot structure */
    long scts;
    /* Find info for this time slot, specified by 'port_num' */
    /* (assumes port_num is valid) */
    pline = port + port_num;
    /* Fill in the SCbus time slot information */
    sc_tsinfo.sc_numts = 1;
    sc_tsinfo.sc_tsarrayp = &scts;
    /* Get SCbus time slot connected to transmit of voice
    channel on board .1 */
    if (vpm_getxmitslot(pline->chdev, &sc_tsinfo) == -1)

```



```

{
    printf("Error message = %s", ATDV_ERRMSGP(pline->chdev));
    exit(1);
}
/* Connect the receive of the digital channel 1 on board 1 to SCBus
time slot of voice channel 1 */
if (lc_Listen(pline->ldev, &sc_tsinfo) == -1)
{
    /* process error return as shown */
    lc_ErrorValue(&metaevent, &lc_error);
    lc_ResultMsg( (long) lc_error, &msg);
    printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
    pline -> ldev, lc_error, msg);
    return(lc_error);
}
return (0);
}

```


lc_Open()

opens a LogicalCall device

Name	int lc_Open(linedevp, NetworkDeviceName, VoiceDeviceName, usrattrp)		
Inputs	LINEDEV *linedev	return	line device pointer
	char * NetworkDeviceName	ASCII string	pointer
	char * VoiceDeviceName	ASCII string	pointer
	Void *usrattrp	User Attribute data pointer	
Returns	0 -1		
Includes	SmartSDK.h		
Category	system controls and tools		
Mode	Synchronous		

Logical Call device open . user
attribute data setting . Logical Call 가 line
device lc_GetUsrAttr() setting data 가 .
lc_SetUsrAttr() .
Network device name voice device name ,
network device voice device open
lc_GetNetworkH() lc_GetVoiceH() .
가 3 , network device TX, RX
voice device RX, TX full duplex routing .
voice device name , line device
network device . voice device voice device open
routing attach . voice device open vpm_open()
 , attach lc_Attach() .
voice device name vpmBxxCxx , dpmBxxCxx .

■ Termination Events

■ Dialogic

Dialogic DNA SDK 3.3 line device 가 2가 가 .
gc_Open() gc_OpenEx() . , logical call line
device lc_Open() .
lc_Open() Dialogic gc_Open() gc_OpenEx()
. dialogic gc_Open() gc_OpenEx()
lc_Open() 가 .
dialogic protocol Name , lc_Open()
protocol name . Korea R2
protocol .



Parameter

linedevp : line device address
NetworkDeviceName : Network Device Name, null terminated string
VoiceDeviceName : voice Device Name, null terminated string
usrattrp : 가 data가 address

■ Example

```
#include "SmartSDK.h"
#include "gcerr.h"
```

```
typedef struct port_tag {
    LINEDEV ldev;
    CRN crn;
    Int voiceH;
    Int networkH;
    Int nState;
} PORT;
METAEVENT g_metaevent;
```

```
void main(int argc, char *argv[])
```



```

{
    init_channel();
    create_event_thread();
    UINT thread_function();
    // inbound or outbound call initiate
}

void init_channel()
{
    PORT port[30];
    char sName[10];

    for(int i=0; i< 30; i++)
    {
        sprintf(sName, "dpmB1C%d", i);
        if (lc_Open(&port[i].ldev, sName, sName, (void *)&port[i]) < 0 )
        {
            // Error
            return ;
        }
        if( lc_GetNetworkH( port[i].ldev, & port[i].networkH ) < 0 )
        {
            // Error
            return;
        }
        if( lc_GetVoiceH( port[i].ldev, & port[i].voiceH ) < 0 )
        {
            // Error
            return;
        }
        port[i].nState = NULL;
    }
}

void create_event_thread()
{

```



```

        // Create thread with thread function
    }

UINT thread_function()
{
    int eventType;
    LINEDEV ldev;
    void *usrattrp;
    PORT * port;
    int voiceh;
    int networkh;

    while( 1 )
    {
        sr_waitevt(-1);
        if( lc_GetMetaEvent( &g_metaevent) < 0 )
        {
            // error
            return 0;
        }

        // for Logical Call Event
        if (g_metaevent.flags & GCME_GC_EVENT)
        {
            eventType = g_metaevent.evtttype;
            ldev = g_metaevent.linedev;
            if( lc_GetUsrAttr( ldev, usrattrp ) < 0 )
            {
                // Error
                return;
            }
            port = (PORT *)usrattrp;
            networkh = port->networkH;
            voiceh = port->voiceH;

            switch( eventType )

```



```

        {
        case GCEV_DETECTED:
            // event processing
            break;
        case GCEV_OFFERED:
            // event processing
            break;
        :
        :
        :
    }
    } // end of while
    return 0;
}

```


lc_ResetLineDev()

disconnects any active calls

Name	int lc_ResetLineDev(linedev, mode)
Inputs	LINEDEV linedev : line device handle unsigned long mode : async or sync
Returns	0 -1
Includes	SmartSDK.h
Category	system control and tools
Mode	Synchronous, Asynchronous

■

line device active call disconnect . ,
call set up . trunk error가 ,
. 가 call state NULL State
call initiate . ,

GCEV_RESETLINEDEV 가 .
call establishment GCEV_TASKFAIL
.

■ **Termination Events**

GCEV_RESETLINEDEV 가 , line device NULL
State .
GCEV_TASKFAIL .

■ **Termination Events**

.

■ **Dialogic**

.

■	Parameter
linedev	logical Call line device
mode	EV_ASYNC : EV_SYNC :

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"

#define MAXCHAN 30 /* max. number of channels in system */
/*
 * Data structure which stores all information for each line
 */
struct linebag {
    LINEDEV ldev; /* LogicalCall line device handle */
    CRN crn; /* LogicalCall API call handle */
    int state; /* state of first layer state machine */
} port[MAXCHAN+1];
/*
 * Assume the following has been done:
 * 1. Opened line devices on dpmB1C1.
 *
 * At this point, the application can 'reset'
 * all of it's line devices back to normal.
 * (Alternatively, this could be called at any time)
 */
int restart(void)
{
    int i; /* index for 'port' */
    int ts; /* network time slot number */
    /*
     * Clean up and get ready to generate/accept calls again.
     */
}
```



```

*/
for (ts = 1,i=1; ts <= MAXCHAN; ts++,i++) {
    if (lc_ResetLineDev(port[i].ldev, EV_SYNC) < 0) {
        /* get cause value and process error */
    }
    /*
    Application will need to re-issue lc_WaitCall() to wait
    * for incoming calls*/
}
return (0);
}

```


lc_ResultMsg() retrieves an ASCII string describing a result code

Name	int lc_ResultMsg(error_code, msg)
Inputs	long error_code : error code char **msg return message가
Returns	0 -1
Includes	SmartSDK.h
Category	system control and tools
Mode	Synchronous

■ Logical Call 가 ,
 ASCII string 가 . error_code
 lc_ErrorValue() .

■ Termination Events

■ Dialogic

Dialogic DNA SDK 3.3 가 call control library ID
 . , lc_ResultMsg() .

	Parameter	
error_code	lc_ErrorValue()	error value
msg	error message가	address

■ Example

```

#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"
  
```



```

void print_error_msg(long ldev)
{
    int error_code; /* LogicalCall error code */
    char *msg; /* pointer to error message string */
    if(lc_ErrorValue(&metaevent, & error_code);< 0)
    {
        // Error
    }
    if(lc_ResultMsg( (long) error_code, &msg) < 0 )
    {
        // Error
    }
    printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
}

```


lc_ResultValue()

retrieves the cause of an event

Name	int lc_ResultValue(metaeventp , lc_resultp)
Inputs	METAEVENT * metaeventp : METAEVENT int * lc_resultp : return result value가
Returns	0 -1
Includes	SmartSDK.h, gcerr.h
Category	system control and tools
Mode	Synchronous



가 . lc_ResultMsg()

ASCII string

gcerr.h

GCRV_NORMAL : normal completion

GCRV_NOANSWER : event caused by no answer

GCRV_BUSY : Line is busy

■ Termination Events

■ Dialogic

Dialogic DNA SDK 3.3 가 call control library ID call control
library result . , lc_ResultValue()



Parameter

metaeventp lc_GetMetaEvent()

	METAEVENT
lc_resultp	result code가

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"

int get_result_value(void)
{
    int lc_error; /* LogicalCall error code */
    int lc_result;
    char *msg; /* pointer to error message string */

    sr_waitevt(-1);

    if (lc_GetMetaEvent(&metaevent) < 0)
    {
        // error
        return -1;
    }
    if (metaevent.flags & GCME_GC_EVENT)
    {
        if (lc_ResultValue( &metaevent, &lc_result) < 0 )
        {
            // error
            return -1;
        }
        lc_ResultMsg( (long) lc_error, &msg);
        printf(" Result value : %d ( %s )\n", lc_result, msg);
        return lc_result;
    }
}
```


lc_SetEvtMsk()

sets the event mask

Name	int lc_SetEvtMsk(linedev, bitmask, action)		
Inputs	LINEDEV linedev :	line device handle	
	Unsigned long bitmask :	bitmask	
	int action :	masking	
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	system control and tools		
Mode	Synchronous		

■

	line device	event mask value	.	application
	clear	event	. Default	event
mask value	enable	.		

■ **Termination Events**

.

■ **Dialogic**

.

■

Parameter	.
-----------	---

linedev :logical Call line device handle.

bitmsk : masking bitmask value. bitwise operation ‘ OR’ 가 .

action : action bitmask . 3가 action .

Table 23. lc_SetEvtMsk() action

Action	description
GCACT_SETMSK	bitmsk enable event disable .
GCACT_ADDMSK	event bitmsk 가 enable . bitmsk
GCACT_SUBMSK	event bitmsk bitmsk disable .

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"
/*
* Assume the following has been done:
* 1. The line device has been opened.
*/
int set_event_mask(LINEDEV ldev)
{
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    /*
    * Set the event blocked and unblocked event masks to enable
    * for this application.
    */
    /*
    * Enable the Blocked and Unblocked events.
    */
    if (lc_SetEvtMsk(ldev, GCEV_DETECTED, GCACT_SUBMSK) < 0 ) {
        /* process error return as shown */
    }
}
```



```
    lc_ErrorValue(&metaevent, &lc_error);
    lc_ResultMsg( (long) lc_error, &msg);
    printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
    ldev, lc_error, msg);
    return(lc_error);
}
}
```


lc_SetUsrAttr()

sets an attribute defined by the user

Name	int lc_SetUsrAttr(linedev, usrattr)		
Inputs	LINEDEV linedev :	line device handle	
	void *usrattr	user attribute	
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	system control and tools		
Mode	Synchronous		

■

가 data line device binding . data
lc_GetUsrAttr() 가 . usrattr line device
call state 가
, linedev binding 가
lc_GetUsrAttr() 가 .
lc_Open() .

■ Termination Events

.

■ Dialogic

.

■ Parameter .

linedev : logical Call line device handle.

usrattr : 가 data address

■ Example

#include <windows.h>


```

#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"
#define MAXCHAN 30 /* max. number of channels in system */
/*
 * Data structure which stores all information for each line
 */
struct linebag {
    LINEDEV ldev; /* LogicalCall line device handle */
    CRN crn; /* LogicalCall API call handle */
    int state; /* state of first layer state machine */
} port[MAXCHAN+1];
/*
 * Associates port_num with ldev for later use
 * by other procedures - will save table searches
 * for the port_num corresponding to ldev
 */
int set_usrattr(LINEDEV ldev, int port_num)
{
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    /*
     * Assuming that a line device is opened already and
     * that its ID is ldev, let us store a meaningful number
     * for this ldev as an attribute for this ldev set by user
     */
    if (lc_SetUsrAttr(ldev, (void *) port_num) < 0 ) {
        /* process error return as shown */
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg( (long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
            ldev, lc_error, msg);
        return(lc_error);
    }
    return (0);
}

```


lc_UnListen() disconnects a receive channel from the network SCbus time slot

Name	int lc_UnListen(linedev)
Inputs	LINEDEV linedev : line device handle
Returns	0 -1
Includes	SmartSDK.h
Category	system controls and tools
Mode	Synchronous

■

lc_listen() network device RX Scbus time slot disconnect .

■ **Termination Events**

■ **Dialogic**

Dialogic DNA SDK 3.3 가 synchronous, Asynchronous
mode . , synchronous .
lc_UnListen() .

■ Parameter .

linedev :LogicalCall line device handle.

■ **Example**

```
#include <windows.h>
#include <stdio.h>
#include " SmartSDK.h"
```



```

#include "gcerr.h"
#define MAX_CHAN 30 /* max. number of channels in system */
/*
 * Data structure which stores all information for each line
 */
static struct linebag {
    LINEDEV ldev; /* LogicalCall API line device handle */
    CRN crn; /* LogicalCall API call handle */
    int blocked; /* channel blocked/unblocked */
    int networkh; /* network handle */
} port[MAX_CHAN+1];
struct linebag *pline; /* pointer to access line device */
/*
 * Assume the following has been done:
 * 1. Opened line devices on dpmB1C1.
 * 2. Each line device and voice handle is stored in linebag structure "port"
 */
int call_unlisten(int port_num)
{
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    /* Find info for this time slot, specified by 'port_num' */
    /* (assumes port_num is valid) */
    pline = port + port_num;
    /* Disconnect receive of digital board 1, timeslot 1
    from all SCbus timeslots */
    if (lc_UnListen(pline->ldev) == -1)
    {
        /* process error return as shown */
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg( (long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
        pline -> ldev, lc_error, msg);
        return(lc_error);
    }
    return (0);
}

```


}

lc_Attach()

attaches a voice resource

Name	int lc_Attach(linedev, voiceh, mode)		
Inputs	LINEDEV linedev :	line device handle	
	int voiceh	voice device handle	
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	interface specific		
Mode	Synchronous		

■ Description

line device voice resource attach . voice resource attach
logical call function voice resource .

■ Termination Events

.

■ Dialogic

.



Parameter

.

linedev : Logical Call line device handle.

voiceh : line device attach voice handle.

mode : EV_SYNC

■ Example

#include <windows.h>


```

#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"

void main()
{
    LINEDEV ldev;
    int voiceh;
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */

    char *networDeviceName = "dpmB1C1";
    char *voiceDeviceName = "dpmB1C1";

    if( lc_Open( &ldev, networDeviceName, NULL, 0 ) < 0 )
    {
        // Error
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg((long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
                                                         ldev, lc_error, msg);

        return;
    }

    if( (voiceh = vpm_open(voiceDeviceName, 0)) < 0 )
    {
        // Error
        return;
    }

    if( lc_Attach(ldev, voiceh, EV_SYNC) < 0 )
    {
        // Error
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg((long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",

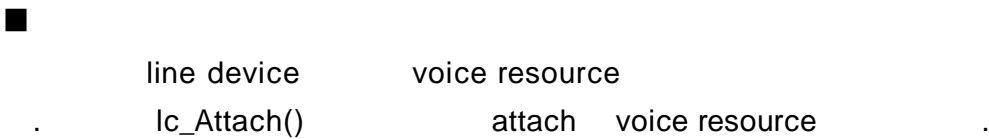
```



```
ldev, lc_error, msg);
```

```
    return;  
}  
if( nr_scroute(ldev, SC_DPM, voiceh, SC_DSP, SC_FULLDUP ) < 0 )  
{  
    //Error  
    return;  
}  
.  
.  
.  
}
```


lc_Detach()		logically detach a voice resource	
<hr/>			
Name	int lc_Detach(linedev, voiceh, mode)		
Inputs	LINEDEV linedev :	line device handle	
	int voiceh	voice device handle	
	unsigned long mode :	async or sync	
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	interface specific		
Mode	Synchronous		
<hr/>			



■ Termination Events

■ Dialogic

■	Parameter	.
---	-----------	---

linedev : Logical Call line device handle.

voiceh : line device detach voice handle.

mode : EV_SYNC

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"
/*
 * Assume the following has been done:
 * 1. The line device (ldev) has been opened.
 * 2. The voice and network resources have been routed together
 * 3. Voice resource is no longer needed for this line device
 */
/* detaches the ldev's voice handle from ldev */
int detach(LINEDEV ldev)
{
    int lc_error; /* LogicalCall error code */
    char *msg; /* points to the error message string */
    int voiceh; /* Voice handle attached to ldev */
    if (lc_GetVoiceH(ldev, &voiceh) == 0) {
        if (lc_Detach(ldev, voiceh, EV_SYNC) < 0) {
            /* process error return as shown */
            lc_ErrorValue(&metaevent, &lc_error);
            lc_ResultMsg((long) lc_error, &msg);
            printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
                ldev, lc_error, msg);
            return(lc_error);
        }
        /*
         * Application should now unroute the voice and network resources from
         * each other (using functions like nr_scunroute() or nr_scunroute()) to
         * complete the disassociation of them from each other.
         */
    } else {
        /* Process lc_GetVoiceH() error */
    }
}
```



```
    return (0);  
}
```


lc_GetVoiceH() returns the voice device handle

Name	int lc_GetVoiceH(linedev, voicehp)		
Inputs	LINEDEV linedev :	line device handle	
	int *voicehp	return voice device handle pointer	
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	interface specific		
Mode	Synchronous		



line device voice device handle return .

■ **Termination Events**

.

■ **Dialogic**

.



Parameter .

linedev	Logical Call line device handle.		
voicep	voice device handle	address	

■ **Example**

```
#include <windows.h>
#include <stdio.h>
#include " SmartSDK.h"
```



```

#include "gcerr.h"
/*
 * Assume the following has been done:
 * 1. A line device has been opened specifying voice resource
 * 2. A call associated with ldev is in the connected state
 */
int get_voice_handle(LINEDEV ldev, int *voicehp)
{
    int lc_error; /* LogicalCall error code */
    lchar *msg; /* points to the error message string */
    if (lc_GetVoiceH(ldev, voicehp) == 0) {
        /*
         * Application may now perform voice processing (e.g., play a prompt)
         * using the voice handle.
         */
        return(0);
    } else {
        /* process error return as shown */
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg( (long) lc_error, &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
            ldev, lc_error, msg);
        return(lc_error);
    }
}

```


Ic_GetCallInfo()		gets information for the call	
Name	int Ic_GetCallInfo(crn, info_id, valuep)		
Inputs	CRN crn :	Call reference number	
	Int info_id	call info ID	
	char *valuep	info buffer	pointer
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	interface specific		
Mode	Synchronous		

■ **Description**
 call (Table 20.) . call CHARGE
 NOCHARGE 가 .

■ **Termination Events**
 .

■ **Dialogic**
 .

■	Parameter	.
crn	Call Reference Number	
info_id	Type.	
valuep	가	buffer address

Table 20. Ic_GetCallInfo() info_id Parameter ID Definitions

info_id Parameter		valuep format
CALLINFOTYPE	call CHARGE NOCHARGE	String

■ Example

```

/*
 * Assume the following has been done:
 * 1. device has been opened
 * 2. lc_WaitCall() has been issued to wait for a call.
 * 3. lc_GetMetaEvent() or lc_GetMetaEventEx() (Windows 2000) has been
 * called to convert the event into metaevent.
 * 4. a GCEV_OFFERED has been detected.
 */
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"
/*
 * the variable info_id parameter(s) defines the information
 * requested from the network.
 * The variable valuep stores the returned information.
 */
int get_call_info(CRN crn, int info_id, char *valuep)
{
    LINEDEV ldev; /* Line device */
    int lc_error; /* LogicalCall Error Value */
    char *msg; /* Error Message */
    if(lc_CRN2LineDev(crn, &ldev) < 0) {
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg(lc_error, &msg);
        printf ("Error: lc_CRN2LineDev ErrorValue: %d - %s\n",
            lc_error, msg);
        return(lc_error);
    }
    if(lc_GetCallInfo(crn, info_id, valuep) < 0) {

```



```
        lc_ErrorValue(&metaevent, &lc_error);
        lc_ResultMsg(lc_error , &msg);
        printf ("Error on Device handle: 0x%lx, ErrorValue: %d - %s\n",
        ldev, lc_error, msg);
        return( lc_error);
    }
    return(0);
}
```


lc_StartTrace()		starts logging debug information	
<hr/>			
Name	int lc_StartTrace(linedev, filename)		
Inputs	LINEDEV linedev :	line device handle	
	char *filename	log	file name
Returns	0		
	-1		
Includes	SmartSDK.h		
Category	interface specific		
Mode	Synchronous		
<hr/>			

■ Description

	line device	logging debug
write		. Trace
write		lc_StopTrace()
		call start
.		

■ Termination Events

.

■ Dialogic

.

■	Parameter	.
<hr/>		
linedev	Logical Call line device handle.	
filename	trace	
		.

■ Example

```
#include <windows.h>
#include <stdio.h>
#include "SmartSDK.h"
#include "gcerr.h"

void main()
{
    LINEDEV ldev;
    if( lc_Open( &ldev, "dpmB1C1", "dpmB1C1", 0 ) < 0 )
    {
        // Error
        return;
    }
    char *filename="/tmp/trace.log";
    rc = lc_StartTrace(ldev, filename);
    if ( rc < 0 )
    {
        printf("Error in lc_StartTrace, rc = %x\n", rc);
    }
    else
    {
        /* continue */
    }
}
```


stops the trace

Name	int lc_StopTrace(linedev)
Inputs	LINEDEV linedev : line device handle
Returns	0 -1
Includes	SmartSDK.h
Category	interface specific
Mode	Synchronous

```

      Ic_StartTrace          trace          trace file          .
Log level          error      trace          .

```

■

■

linedev	Logical Call line device handle
----------------	---------------------------------

•

```
/*
 *
 * 1. device has been opened
 * 2. lc_StartTrace() function has been called.
 */
.
```



```
.  
rc = lc_StopTrace(ldev);  
if (rc < 0) {  
    printf("Error in lc_StopTrace, rc = %x\n", rc);  
} else {  
    /* Process event */  
}
```


5. DPM Data Structure and Device Parameters

DPM library data structures	DPM board parameter
DPM Library	Data Structures
dpm_setparm() and dpm_getparm()	Tables (Section 4.1) Parameter (Section 4.2)

5.1. DPM Library Data Structures

DX_EBLK	Event Block Structure
DX_CST	Call Status Transition Structure
METAEVENT	Meta Event Structure
GC_RATE_U	Billing Information Structure

5.1.1. DX_EBLK - Event Block Structure

DPM	가
voice device	DX_EBLK
Window NT	voice programmer's guide for

DX_EBLK	dpm_getevt()	R2 signal
---------	--------------	-----------

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;

ev_event device event
ev_event
```


DE_R2TONE	R2 signal received
DE_R2EOS	R2EOS received
DE_DTRING	rings received
DE_DTHOOKON	Hook on received
DE_ANSWERED	answered received

	dpm_getevt()	dpm_setevtmsk()
masking	.	
ev_data	ev_event	event data

Table 21. Values Returned in ev_data

Event	Data Returned in ev_data	
DE_R2TONE	8bits	Group number (R2_GROUP_1 or R2_GROUP_A)
	8bits	digit (1~ 15)

5.1.2. DX_CST - Call Status Transition Structure

DPM 가 .

voice device DX_CST voice programmer' s guide for Window NT

DX_CST dpm_sendevt() R2 signal

structure typedef .

```
typedef struct dx_cst {
    unsigned short cst_event;    // CST Event
    unsigned short cst_data;    // Data Associated with the Event
} DX_CST;
```

ev_event device event .

ev_event .

```
                                _Windows 2000) to
                                * fill in these fields */
                                /* only valid if an event was returned */
unsigned long flags; /* flags field */
void          *evtdatap; /* pointer to the event data block -
                           sr_getevtdatap */
                           /* will be f(event, cclib) */
long          evtlen;    /* event length - UNIX or Windows 2000 sr_getevtlen */
                           /* May change as libraries are added */
```



```

long      evtdev;          /* event device - sr_getevtdev */
long      evttype;         /* event type - sr_getevttype */
LINEDEV   linedev;         /* line device */
CRN        crn;            /* crn - if 0, no crn for this event */
long      rfu2;            /* reserved for future use */
void      *usrattr;        /* user attribute associated with linedev */
int        cclibid;        /* ID of cclib of associated event */
int        rfu1;          /* reserved for future use */
} METAEVENT, *METAEVENTP;

```

5.1.4. GC_RATE_U - Billing Information Structure

```

                                lc_SetBilling()          billing          .

structure      typedef          .

typedef union {
    struct {
        long cents;
    } ATT, *ATT_PTR;
} GC_RATE_U, *GC_RATE_U_PTR;

```


Appendix A

Standard Runtime Library

DPM 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 .

DPM 가
voice device Standard Runtime Library voice programmer's
guide for Window NT Appendix A .

Event Management Functions

The Event Management function DPM device termination
event , .

dpm_seizure()
dpm_release()
dpm_answer()

termination event Standard Runtime Library event
Management 가 Ic_GetMetaEvent()
METAEVENT evt type .

Ic_WaitCall()
Ic_AcceptCall()
Ic_AnswerCall()
Ic_DropCall()
Ic_MakeCall()
Ic_SetChanState()
Ic_ResetLineDev()

DPM board 가 Event Management function table
list . Table 25 Table 26 event management
list .

Table 25. DPM Device Inputs for Event Management Function

Event Management Function	DPM Device Input	Valid Value
sr_enbhdr() <i>Enable event handler</i>	evt_type	R2EV_SEIZURE
		R2EV_RELEASE
		R2EV_ANSWER
		R2EV_SIGEVT
		R2EV_R2TRACE
		R2EV_E1TRACE
		R2EV_E1ERRC
		GCEV_OFFERED
		GCEV_DETECTED
		GCEV_ACCEPT
		GCEV_ANSWERED
		GCEV_ALERTING
		GCEV_CALLSTATUS
		GCEV_CONNECTED
		GCEV_DROPCALL
		GCEV_DISCONNECTED
		GCEV_RESETLINEDEV
		GCEV_SETBILLING
		GCEV_SETCHANSTATE
		GCEV_TASKFAIL
		GCEV_BLOCK
		GCEV_UNBLOCK
		GCEV_CALLINFO
sr_dishdr() <i>Disable event handler</i>	evt_type	

Table 26. DPM Device Returns from Event Management Function

Event Management Function	Return Description	Returned Value
sr_getevtdev() Get SCT Device <i>handle</i> lc_GetMetaEvent() Get Metaevent struct : Metaevent.linedev	device	DPM device handle
sr_getevttype() Get event type	event type	R2EV_SEIZURE R2EV_RELEASE R2EV_ANSWER R2EV_SIG EVT R2EV_R2TRACE R2EV_E1TRACE R2EV_E1ERRC
lc_GetMetaEvent() Get Metaevent struct : Metaevent.ev t type	event type	GCEV_OFFERED GCEV_DETECTED GCEV_ACCEPT GCEV_ANSWERED GCEV_ALERTING GCEV_CALLSTATUS GCEV_CONNECTED GCEV_DROP CALL GCEV_DISCONNECTED GCEV_RESETLINEDEV GCEV_SETBILLING GCEV_SETCHANSTATE GCEV_TASKFAIL GCEV_BLOCK GCEV_UNBLOCK GCEV_CALLINFO
sr_getevtlen() Get event data length	event length	sizeof (DX_CST)

<i>lc_GetMetaEvent()</i> <i>Get Metaevent struct</i> <i>: Metaevent.evrlen</i>		
<i>sr_getevtdatap()</i> <i>Get event data pointer</i> <i>lc_GetMetaEvent()</i> <i>Get Metaevent struct</i> <i>: Metaevent.evtdatap</i>	event data	<i>DX_CST</i> structure pointer

Standard Attribute Functions

Standard Attribute function signaling bit .
Standard Attribute function DPM device information Table 11
list .

Table 27. Standard Attribute Functions

Standard Attribute Devices Function	Information Returned for DPM
DPMT_BDSGBIT()	time slot signaling bit .
DPMT_TSSGBIT()	time slot signaling bit .
DPMT_DNLDVER()	.

Appendix B

Publications

DPM hardware software product
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* .
- Voice device software *voice programmer's Guide for Windows 2000* .