Guia do Programador

Sistema de Desenvolvimento para placas Digivoice.

MAN0048 - Versão: 4.3.0.1 rc31
Copyright © 2010 Digivoice Eletrônica

Conteúdo

Introdução

Recursos para o Desenvolvedor

Primeiros Passos

14. Windows

14. 32/64 bits
16. Instalação no Windows
19. Testando a placa instalada
20. Linux

20. Instalação no Linux
22. Preparando o ambiente
23. Módulos do kernel
25. Compilando a VoicerLib
26. Exemplo em linguagem C

Guia de Programação

28. Conceitos Básicos - API

32. Conceitos Básicos - ActiveX

33. Guia de Migração de Versões Anteriores

41. Instruções de instalação de placas

44. Inicializando os Serviços

45. Finalizando os Serviços

46. Detectando Ring

47. Atendendo e Desligando

48. Supervisão de Linha

60. Detecção de Silêncio

61. Detecção de Tons

63. Detecção de Dígitos

Copyright © 2010 Digivoice Eletrônica

Conteúdo

67. Identificação de Chamadas
68. Portas Virtuais
71. Gravando uma Conversa
77. Reproduzindo Mensagens
79. Conferência entre portas
81. Streaming de Áudio
84. Utilizando a placa VB0404GSM
91. Gravação em Paralelo
92. Placas FXO
93. Placas E1
97. Programação das Placas E1
98. Configurações de Sincronismo
99. Alarmes
101. Protocolo CAS Customizado
101. Introdução CAS Customizado
102. Protocolo ISDN PRI
102. Inicialização Protocolo ISDN PRI
105. Funções de Controle da Thread ISDN PRI
106. Efetuando Chamadas
108. Recebendo Chamadas
110. Finalizando Chamadas
111. Protocolo R2D MFC
111. Inicialização Protocolo R2D MFC
112. Efetuando Chamadas
115. Recebendo Chamadas
116. Funções de Controle da Thread R2
117. Depurando aplicativos para placas E1 com R2MFC
118. Funções Especiais
118. Introdução Funções Especiais
119. Funções de Idle
121. Funções de Prompt
122. Funções de Menu
123. Funções de Discagem e Transferência

Copyright © 2010 Digivoice Eletrônica

Conteúdo

126. Reproduzindo Data
127. Reproduzindo Números Cardinais
128. Reproduzindo Números Dígito a Dígito
129. Reproduzindo Valores por Extenso
130. Reproduzindo Lista de Mensagens
134. Distribuindo uma Aplicação

Guia de Referência

135. Mensagens de Erro

141. Funções/Métodos

142. AbortCall
143. AudioMonitor (dg_AudioMonitor)
144. CancelGetDigits (dg_CancelGetDigits)
145. ChatAddPort (dg_ChatAddPort)
146. ChatDisablePort (dg_ChatDisablePort)
147. ChatEnablePort (dg_ChatEnablePort)
149. ChatRemovePort (dg_ChatRemovePort)
150. CheckCode (dg_CheckCode)
151. ClearDigits (dg_ClearDigits)
152. ConnectAudioChannels (dg_ConnectAudioChannels)
153. ConfigCallProgress (dg_ConfigCallProgress)
158. ConfigCustomCAS (dg_ConfigCustomCAS)
161. ConfigE1Thread (dg_ConfigE1Thread)
164. ConfigGSMThread (dg_ConfigGSMThread)
166. ConfigISDNThread (dg_ConfigISDNThread)
168. CreateCallProgress (dg_CreateCallProgress)
170. CreateChatRoom (dg_CreateChatRoom)
171. CreateCustomCAS (dg_CreateCustomCAS)
173. CreateE1Thread (dg_CreateE1Thread)
175. CreateGSMThread (dg_CreateGSMThread)
177. CreateISDNThread (dg_CreateISDNThread)
178. CreateLoggerControl (dg_CreateLoggerControl)
180. CreateLoggerCCS (dg_CreateLoggerCCS)
182. DefinePortResource (dg_DefinePortResource)
184. DestroyCallProgress (dg_DestroyCallProgress)

Copyright © 2010 Digivoice Eletrônica

Conteúdo

185. DestroyChatRoom (dg_DestroyChatRoom)
186. DestroyCustomCAS (dg_DestroyCustomCAS)
187. DestroyE1Thread (dgDestroyE1Thread)
188. DestroyGSMThread (dg_DestroyGSMThread)
189. DestroyISDNThread (dg_DestroyISDNThread)
190. DestroyLoggerControl (dg_DestroyLoggerControl)
191. DestroyLoggerCCS (dg_DestroyLoggerCCS)
192. dg_SetEventCallback
193. Dial (dg_Dial)
195. DisableAGC (dg_DisableAGC)
196. DisableAnswerDetection (dg_DisableAnswerDetection)
198. DisableAutoFramers (dg_DisableAutoFramers)
199. DisableCallProgress (dg_DisableCallProgress)
200. DisableDebug (dg_DisableDebug)
201. DisableDTMFFilter (dg_DisableDTMFFilter)
202. DisableE1Thread (dg_DisableE1Thread)
203. DisableEchoCancelation (dg_DisableEchoCancelation)
204. DisableGSMThread (dg_DisableGSMThread)
205. DisableInputBuffer (dg_DisableInputBuffer)
206. DisableISDNThread (dg_DisableISDNThread)
207. DisableMailBoxDetection (dg_DisableMailBoxDetection)
208. DisablePulseDetection (dg_DisablePulseDetection)
209. DisableSilenceDetection (dg_DisableSilenceDetection)
210. DisconnectAudioChannels (dg_DisconnectAudioChannels)
211. EnableAGC (dg_EnableAGC)
212. EnableAnswerDetection (dg_EnableAnswerDetection)
214. EnableCallProgress (dg_EnableCallProgress)
216. EnableDebug (dg_EnableDebug)
217. EnableDTMFFilter (dg_EnableDTMFFilter)
219. EnableE1Thread (dg_EnableE1Thread)
220. EnableEchoCancelation (dg_EnableEchoCancelation)
222. EnableFSKDetection (dg_EnableFSKDetection)
223. EnableGSMThread (dg_EnableGSMThread)
224. EnableInputBuffer (dg_EnableInputBuffer)
226. EnableISDNThread (dg_EnableISDNThread)
227. EnableMailBoxDetection (dg_EnableMailBoxDetection)
229. EnablePulseDetection (dg_EnablePulseDetection)

Copyright © 2010 Digivoice Eletrônica

Conteúdo

231. EnableSilenceDetection (dg_EnableSilenceDetection)
233. Flash (dg_Flash)
235. ForceSingleSpan (dg_ForceSingleSpan)
236. GenerateMF (dg_GenerateMF)
238. GetAbsolutePortNumber (dg_GetAbsolutePortNumber)
239. GetAlarmStatus (dg_GetAlarmStatus)
240. GetBusyCause (dg_GetBusyCause)
244. GetCallerID (dg_GetCallerID)
246. GetCardBus (dg_GetCardBus)
247. GetCardInfo (dg_GetCardInfo)
248. GetCardInterface (dg_GetCardInterface)
249. GetCardNumber (dg_GetCardNumber)
250. GetCardPortsCount (dg_GetCardPortsCount)
251. GetCardsCount (dg_GetCardsCount)
252. GetCardSlot (dg_GetCardSlot)
253. GetCardType (dg_GetCardType)
254. GetDigits (dg_GetDigits)
256. GetDriverEnabled (dg_GetDriverEnabled)
257. GetE1Number (dg_GetE1Number)
259. GetE1ThreadStatus (dg_GetE1ThreadStatus)
260. GetISDNThreadStatus (dg_GetISDNThreadStatus)
261. GetLibVersion (dg_GetLibVersion)
262. GetLoggerCallType (dg_GetLoggerCallType)
263. GetNameID (dg_GetNameID)
265. GetPlayFormat (dg_GetPlayFormat)
266. GetPortCardType (dg_GetPortCardType)
268. GetPortInterface (dg_GetPortInterface)
269. GetPortsCount (dg_GetPortsCount)
270. GetPortStatus (dg_GetPortStatus)
271. GetRecordFormat (dg_GetRecordFormat)
272. GetRelativeChannelNumber (dg_getRelativeChannelNumber)
273. GetText (dg_GetText)
274. GetVersion (dg_GetVersion)
275. GSMCallControl (dg_GSMCallControl)
278. GSMCheckSignalQuality (dg_GSMCheckSignalQuality)
279. GSMClearAllSMS (dg_GSMClearAllSMS)
280. GSMDeleteSMS (dg_GSMDeleteSMS)

Copyright © 2010 Digivoice Eletrônica

Conteúdo

281. GSMGetIndexList (dg_GSMGetIndexList)
282. GSMGetLastCommand (dg_GSMGetLastCommand)
283. GSMGetMemory (dg_GSMGetMemory)
284. GSMGetMessage (dg_GSMGetMessage)
285. GSMGetSignalQuality (dg_GSMGetSignalQuality)
286. GSMGetSMS (dg_GSMGetSMS)
287. GSMGetSMSConfirmation (dg_GSMGetSMSConfirmation)
288. GSMListSMS (dg_GSMListSMS)
289. GsmRawToWave (dg_GsmRawToWave)
290. GsmRawToWave49 (dg_GsmRawToWave49)
291. GSMReadAndDeleteSMS (dg_GSMReadAndDeleteSMS)
292. GSMRestartPort (dg_GSMRestartPort)
293. GSMSendCommand (dg_GSMSendCommand)
295. GSMSendSMS (dg_GSMSendSMS)
297. GSMSetPinNumber (dg_GSMSetPinNumber)
298. GsmToWave (dg_GsmToWave)
299. GsmToWave49 (dg_GsmToWave49)
300. HangUp (dg_HangUp)
301. IdleAbort (dg_IdleAbort)
302. IdleSettings (dg_IdleSettings)
304. IdleStart (dg_IdleStart)
305. IsCallInProgress
306. IsPlaying (dg_IsPlaying)
307. IsRecording (dg_IsRecording)
308. LocalBridgeConnect (dg_LocalBridgeConnect)
309. LocalBridgeDisconnect (dg_LocalBridgeDisconnect)
310. MakeCall
311. MenuAbort
312. MenuErrorSettings
313. MenuStart
315. PauseInputBuffer (dg_PauseInputBuffer)
316. PickUp (dg_PickUp)
318. PlayBuffer (dg_PlayBuffer)
320. PlayCardinal
321. PlayCurrency
322. PlayDate
323. PlayFile (dg_PlayFile)

Copyright © 2010 Digivoice Eletrônica

Conteúdo

325. PlayList
326. PlayListAdd
327. PlayListClear
328. PlayListGetCount
329. PlayListRemoveItem
330. PlayNumber
331. PlayTime
332. PromptAbort
333. PromptSettings
334. PromptStart
336. R2AskForGroupII (dg_R2AskForGroupII)
337. R2AskForID (dg_R2AskForId)
338. R2SendGroupB (dg_SendGroupB)
339. ReadDigits (dg_ReadDigits)
340. RecordFile (dg_RecordFile)
342. RecordPause (dg_RecordPause)
343. ResetError (dg_ResetError)
345. ResetPortResource (dg_ResetPortResource)
346. ReturnCodeGSMToString (dg_ReturnCodeGSMToString)
347. ReturnCodeToString (dg_ReturnCodeToString)
348. ReturnCodeToStringCauseBusy (dg_ReturnCodeToStringCauseBus
349. SendISDNCommand (dg_SendISDNCommand)
351. SendR2Command (dg_SendR2Command)
353. SetAlarmMode (dg_SetAlarmMode)
355. SetAudioInputCallback (dg_SetAudioInputCallback)
356. SetBearerCapability (dg_SetBearerCapability)
358. SetCallAfterAnswer
359. SetCallAfterPickup
360. SetCallBusyPhrase
361. SetCallBusyReturnFlash
362. SetCalledNumber (dg_SetCalledNumber)
364. SetCallingNumber (dg_SetCallingNumber)
366. SetCallFlashTime
367. SetCallNoAnswerPhrase
368. SetCallNoAnswerReturnFlash
369. SetCallNoAnswerRingCount
370. SetCallPauseBeforeAnalysis

Copyright © 2010 Digivoice Eletrônica

Conteúdo

371. SetCallStartFlash
372. SetCallWaitForDialTone
373. SetCardDetections (dg_SetCardDetections)
375. SetCardSyncMode (dg_SetCardSyncMode)
377. SetDetectionType (dg_SetDetectionType)
379. SetDialDelays (dg_SetDialDelays)
381. SetDigitFrequency(dg_SetDigitFrequency)
383. SetDigitGain(dg_SetDigitGain)
385. SetDTMFConfig (dg_SetDTMFConfig)
386. SetE1CRC4Option (dg_SetE1CRC4Option)
387. SetFastDetection (dg_SetFastDetection)
389. SetFaxFrequencies (dg_SetFaxFrequencies)
391. SetFramerLoop (dg_SetFramerLoop)
393. SetFrequency (dg_SetFrequency)
395. SetFXCardType (dg_SetFXCardType)
396. SetGSMMode (dg_SetGSMMode)
397. SetHangUpCause (dg_SetHangUpCause)
399. SetLoggerSilenceThreshold (dg_SetLoggerSilenceThreshold)
401. SetNameID (dg_SetNameID)
402. SetNextE1RxCount (dg_SetNextE1RxCount)
404. SetPlayFormat (dg_SetPlayFormat)
406. SetPortChatLog(dg_SetPortChatLog)
408. SetPortGain (dg_SetPortGain)
410. SetPortID (dg_SetPortID)
411. SetRecordFormat (dg_SetRecordFormat)
413. SetReverseCharging (dg_SetReverseCharging)
414. SetRecordGain (dg_SetRecordGain)
415. SetSilenceThreshold (dg_SetSilenceThreshold)
417. SetStartE1RxCount (dg_SetStartE1RxCount)
419. SetText (dg_SetText)
420. SetTwist (dg_SetTwist)
422. ShutdownVoicerLib (dg_ShutdownVoicerLib)
423. StartVoicerLib (dg_StartVoicerLib)
425. StopPlayBuffer (dg_StopPlayBuffer)
426. StopPlayFile (dg_StopPlayFile)
427. StopRecordFile (dg_StopRecordFile)
428. TxRxMixEnable (dg_TxRxMixEnable)

Copyright © 2010 Digivoice Eletrônica

Conteúdo

429. Wave49ToGsm (dg_Wave49ToGsm)
430. Wave49ToGsmRaw (dg_Wave49ToGsmRaw)
431. WaveToGsm (dg_WaveToGsm)
432. WaveToGsmRaw (dg_WaveToGsmRaw)
433. WriteCode (dg_WriteCode)
434. Eventos
435. OnAfterDial (EV_AFTERDIAL)
436. OnAfterFlash (EV_AFTERFLASH)
437. OnAfterMakeCall
438. OnAfterPickUp (EV_AFTERPICKUP)
439. OnAnswerDetected (EV_ANSWERED)
440. OnAudioSignalDetected (EV_AUDIO_SIGNAL)
441. OnBusyDetected (EV_BUSY)
442. OnCallerID (EV_CALLERID)
443. OnCalling (EV_CALLING)
444. OnCallStateChange
445. OnChannel (EV_CHANNEL)
446. OnDialToneDetected (EV_DIALTONE)
447. OnDigitDetected (EV_DTMF)
448. OnDigitsReceived (EV_DIGITSRECEIVED)
449. OnE1Alarm (EV_E1_ALARM)
450. OnE1FramerResponse (EV_FRAMER)
451. OnE1GroupB (EV_GROUP_B)
452. OnE1StateChange (EV_E1CHANGESTATUS)
454. OnErrorDetected (EV_ERRORDETECTED)
455. OnErrorNodeConfig (EV_ERROR_NODE_CONFIG)
456. OnFaxDetected (EV_FAX)
457. OnGSMSMSConfirmation (EV_GSMSMSCONFIRMATION)
458. OnGSMError (EV_GSMERROR)
459. OnGSMMemory (EV_GSMMEMORY)
460. OnGSMMemoryFull (EV_GSMMEMORYFULL)
461. OnGSMMessage (EV_GSMMESSAGE)
462. OnGSMOtherCall (EV_GSMOTHERCALL)
463. OnGSMReady (EV_GSMREADY)
464. OnGSMReturnOK (EV_GSMRETURNOK)
465. OnGSMSIM (EV_GSMSIM)

Copyright © 2010 Digivoice Eletrônica

Conteúdo

466. OnGSMSignalQuality (EV_GSMSIGNALQUALITY)
467. OnGSMSMSReceived (EV_GSMSMSRECEIVED)
468. OnGSMSMSSent (EV_GSMSMSSENT)
469. OnGSMTimeout (EV_GSMTIMEOUT)
470. OnLineOff (EV_LINEOFF)
471. OnLineReady (EV_LINEREADY)
472. OnLoggerEvent (EV_LOGGEREVENT)
473. OnMailBoxDetected(EV_MBDETECTED)
474. OnMenu
475. OnNameID (EV_NAMEID)
476. OnPlayStart (EV_PLAYSTART)
477. OnPlayStop (EV_PLAYSTOP)
478. OnPrompt
479. OnR2Received (EV_R2)
480. OnRecording (EV_RECORDING)
481. OnRecordStart (EV_RECORDSTART)
482. OnRecordStop (EV_RECORDSTOP)
483. OnReverseCharging (EV_REVERSECHARGING)
484. OnRingDetected (EV_RINGS)
485. OnSilenceDetected (EV_SILENCE)
486. OnText (EV_TEXT)
487. Propriedades exclusivas do ActiveX

487. ConfigPath
488. StockSigPath

Copyright © 2010 Digivoice Eletrônica

Introdução

Bem vindo ao Guia do Programador da VoicerLib 4!

A VoicerLib 4 é a versão da VoicerLib para a família de placas DigiVoice tanto para Windows como para Linux. As placas suportadas são:

- Todos os modelos de placas digitais E1
- Placa FXO de 4 ou 8 canais
- Placa GSM 2 ou 4 canais
- Placa FXS

As placas VoicerBox PCI/1 e VoicerPhone PCI/4 não são suportadas por esta versão da VoicerLib. Para estes modelos deve ser utilizada a VoicerLib 2.x para Windows e a VoicerLib 3.1.x para Linux.

Copyright © 2010 Digivoice Eletrônica

Recursos para o Desenvolvedor

Ao adquirir as placas DigiVoice e a VoicerLib, o desenvolvedor pode fazer download de uma coleção de exemplos prontos em nosso site www.digivoice.com.br com o objetivo de explicar o funcionamento das diversas funções da VoicerLib.

Além disso, o desenvolvedor tem disponível a área de suporte do site da DigiVoice na internet, através do endereço http://www.digivoice.com.br/suporte. Lá, estão disponíveis o Fórum de Discussões, Arquivos para Download (exemplos atualizados, instaladores, manuais etc..), Documentos Técnicos e uma seção de Perguntas Frequentes para ajudar o desenvolvedor nas dúvidas mais comuns.

É **muito importante** também ler com atenção toda a seção *Guia de Programação* deste manual pois contém toda a base necessária para programar com as placas DigiVoice, importante para novatos ou para desenvolvedores que já trabalharam com outros hardwares.

Copyright © 2010 Digivoice Eletrônica

Neste capítulo será discutido os passos necessários para instalar a VoicerLib e executar os primeiros procedimentos para verificar se tudo está funcionando corretamente.

Windows

32/64 bits

Utilizando a VoicerLib em Windows 32 bits e 64 bits

O sistema operacional Windows é oferecido em versões para 32 e 64 bits, a VoicerLib, a partir da versão 4.3.0.0, está disponível para rodar em 32 bits e 64 bits.

Apesar do Windows permitir que aplicações de 32 bits rodem em sistemas operacionais de 64 bits a VoicerLib não tem esta opção porque a VoicerLib interage diretamente com o hardware através dos drivers das placas e a comunicação tem que ser de 64 bits para 64 bits, ou seja, para um Windows de 64 bits só será instalada a versão de 64 bits.

Compatibilidade

A partir da versão 4.3.0.0 da VoicerLib, muitas alterações internas foram feitas no sentido de adaptá-la ao Windows 64 bits e a otimizações visando maior desempenho, mas o desenvolvimento de aplicativos manteve-se totalmente compatível com as versões anteriores.

Programas de teste

A VoicerLib é fornecida com alguns programas de teste para suas placas FXO, FXS, E1 e GSM. Estes programas foram compilados para 32 bits e 64 bits. Como estes programas foram desenvolvidos com uma ferramenta Microsoft que utiliza a

Copyright © 2010 Digivoice Eletrônica

tecnologia .NET, é necessário que um pacote Microsoft .NET Framework esteja instalado no computador onde a VoicerLib for instalada para que os programas de testes rodem.

No setup de instalação da VoicerLib (no final da instalação) há a opção de instalar o Microsoft .NET Framework v.2.0. Se o computador já possuir um framework instalado, não é necessário instalar o framework do setup da VoicerLib.

A VoicerLib não depende do Microsoft .NET Framework para ser utilizada, somente os programas de teste.

Visual Studio

Por enquanto, o Visual Studio da Microsoft (2008 e 2010), como a maioria dos programas atuais, é uma aplicação de 32 bits que permite o desenvolvimento de aplicações de 64 bits. Isto tem uma implicação na importação de componentes OCX como a VoicerLib e desenvolvimento de aplicativos de 64 bits.

Quando uma aplicação de 32 bits roda no Windows de 64 bits, o sistema operacional roda a aplicação utilizando a interface WOW64, que é "Windows on Windows 64", ou se for instalado um componente ou dll 32 bits no Windows 64, estes componente serão instalados na pasta "sysWOW64" ao invés da pasta "system32" (padrão do Windows).

Mesmo sendo meio confuso, a Microsoft optou por manter o nome system32 para a pasta de instalação dos componentes ou dll de 64 bits em um Windows de 64 bits.

Ao se desenvolver uma aplicação que exija um componente OCX de 64 bits, como a VoicerLib, será necessário instalar uma OCX de 32 bits para o desenvolvimento de aplicativos, mesmo com a opção de compilação para gerar executáveis de 64 bits.

Ao ser instalada, a Voicerlib coloca uma OCX de 32 bits na pasta "sysWOW64" e uma de 64 bits na pasta "system32", mas o componente a ser importado no Visual Studio tem que ser o de 32 bits da pasta "sysWOW64".

Copyright © 2010 Digivoice Eletrônica

Instalação no Windows

Requisitos mínimos de instalação:

- Windows XP/2000/2003
- Processador Core 2 Duo
- 1 GB de memória

Sistemas Operacionais suportados (32 bits e 64 bits):

- Windows XP Professional SP1/SP2/SP3
- Windows 2003 Server
- Windows Vista Business
- Windows 2008 Server
- Windows 7

Quantidade máxima de placas por computador : 10 placas

Programa de instalação da VoicerLib4:

setup_vlib.exe - Programa de instalação para o desenvolvedor, contendo os drivers e a documentação.

A instalação física das placas DigiVoice é explicada pelo manual *Kit Integrador* que é fornecido impresso acompanhando o hardware. Quanto à instalação do software leia os tópicos a seguir deste manual.

Se você já instalou fisicamente a placa, o Windows a reconhecerá na primeira inicialização. Se você não instalou o software ainda, simplesmente cancele a tela que o Windows mostrar.

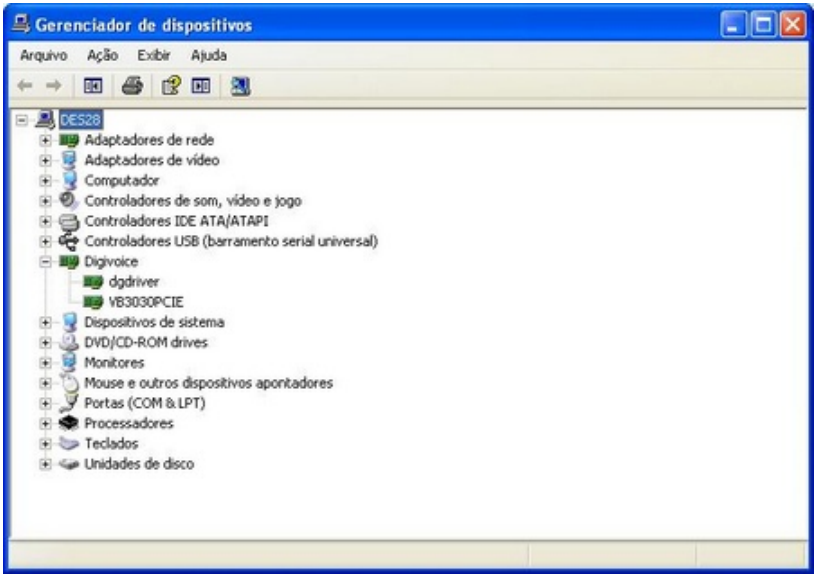
Em seguida, localize e execute o arquivo de instalação da voicerib chamado setup_vlib.exe (que está no CD adquirido ou no site da Digivoice). Após seguir os passos indicados, reinicie o computador quando isso for solicitado pelo programa de instalação.

Ao reiniciar, o Windows mostrará novamente que detectou um novo hardware (Controlador Multimedia). Desta vez, siga os

Copyright © 2010 Digivoice Eletrônica

passos para instalação do hardware. O Windows perguntará se você quer instalar automaticamente ou a partir de um caminho específico. Escolha a segunda opção, apontando para o diretório escolhido na instalação, o caminho padrão é *\%PROGRAM FILES%\VoicerLib4.*

Isto deverá ser suficiente para que o Windows reconheça a placa, como por exemplo "*Digivoice VB3030PCIE*". Se tudo estiver correto, a placa estará pronta para ser utilizada. Caso tenha alguma dúvida, acesse o Gerenciador de Dispositivos do Windows, que deverá apresentar um novo item Digivoice:



Gerenciador de dispositivos - Dispositivos por tipo

O programa de instalação criará o grupo de programas em seu menu Iniciar, com o nome *Digivoice VoicerLib4* contendo:

- Configurador da placa E1 - Programa de configuração e testes
- Programa para placa GSM - Exemplo de funcionalidade da placa

Copyright © 2010 Digivoice Eletrônica

GSM
- Programa de diagnósticos Vlib_Diag 0408 - Programa de testes para a placa VB0408
- Programa de diagnósticos Vlib_Diag 0404FX - Programa de testes para a placa VB0404FX
- Monitor de Sinalização R2MFC (E1)
- Manual do Kit Integrador
- Link para o site da DigiVoice
- Atalho para remover a instalação

Em nosso site www.digivoice.com.br estão disponíveis diversos exemplos e o manual do programador.

Caso a instalação seja feita em um computador de sistema operacional Vista/W2008/W7 ou superior, e por algum motivo esta placa sejam reinstalada em uma máquina XP/2000/2003, é necessário executar um arquivo chamado VlibUpdateRetro.exe, que está no diretório: *\%PROGRAM FILES%\VoicerLib4\VlibUpdateRetro\.*

Execute o arquivo e se ocorrer algum erro, aparecerá na tela e será necessário apertar alguma tecla para continuar, nesse caso, entre em contato com a Digivoice e nos informe a mensagem de erro, caso execute com sucesso apenas uma tela de atualização aparecerá rapidamente, é obrigatório a reinicialização do computador.

Copyright © 2010 Digivoice Eletrônica

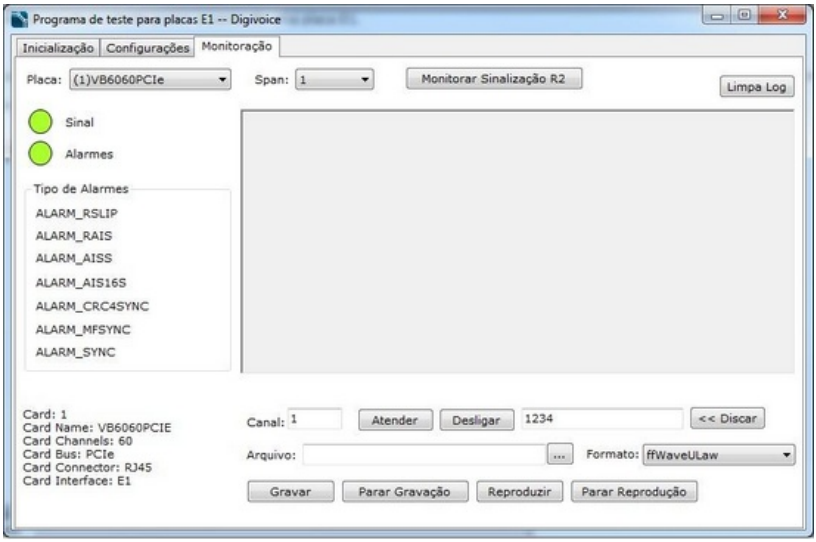
Testando a placa instalada

Nesse tópico iremos mostrar o teste passo a passo de uma placa E1.

Placa E1

Para verificar se a placa E1 foi corretamente instalada e está operacional, execute o programa *Configurador da placa E1* que está no menu Iniciar do Windows, sob o grupo *Digivoice VoicerLib4***.**

A seguinte tela aparecerá:



Tela do configurador E1

A primeira ação a fazer é clicar sobre a opção *Detectar Placas*, no menu apresentado na parte inferior esquerda da janela. O Configurador iniciará a detecção e inicialização das placas.

Copyright © 2010 Digivoice Eletrônica

Com isso, é possível ter certeza que as placas foram corretamente instaladas e estão operacionais.

Linux

Instalação no Linux

Requisitos mínimos de instalação:

- Celeron 1 Ghz com 256Mb de memória
- Distribuição Linux com kernel versão 2.4.x. ou 2.6.x

Quantidade máxima de placas por computador : 7 placas

O arquivo que contém todo o material necessário chama-se *voicerlib-4.x.x.tar.gz* (4.x.x será a versão corrente) sendo que para descompactá-lo, utilize o comando:

```
tar xvzf voicerlib-4.x.x.x.tar.gz
```

Com isso, o diretório *voicerlib-4.x.x.x* será criado contendo os fontes da voicerlib e do device driver.

- *voicerlib-4.x.x.x* – Contém os arquivos relativos à API

```
|
|___/driver/linux
| |__ Device driver para as placas DigiVoice.
|
|___/firmware
| |__ Arquivos dos firmwares das placas e outros de
configuração (CallProgress, etc...)
|
|___/samples/dll_so
| |__ c - programa exemplo/teste desenvolvido em ANSI C
|
|___/src_common
```


Copyright © 2010 Digivoice Eletrônica

```
| |__ Código-fonte da VoicerLib - arquivos comuns
(multiplataforma)
|
||___/src_linux
| |__ Código-fonte da VoicerLib (shared object) para
Linux
|
```

Para trabalhar com o Asterisk, é necessário também baixar e instalar o dgvchannel - channel driver para Asterisk. Acesse o site http://www.digivoice.com.br.

Copyright © 2010 Digivoice Eletrônica

Preparando o ambiente

Para estar apto a compilar a VoicerLib no seu ambiente será necessário que esteja disponível um sistema de desenvolvimento (gcc, make, etc...) bem como os fontes do kernel.

Todos estes arquivos normalmente encontram-se disponíveis no CD original de sua distribuição. A VoicerLib foi desenvolvida e testada em versões 2.4 e 2.6 do kernel, utilizando o compilador gcc versão 3.0.1 ou superior (a versão 2.95 do gcc causará diversos erros de compilação).

O gcc 4 também tem sido utilizado sem problemas. Para compilar o exemplo em C, será necessário a biblioteca ncurses-devel versão 5.3 ou superior.

A maioria dos passos necessários para colocação da VoicerLib e aplicativos para funcionar precisam ser executados com o usuário root, portanto todo o cuidado é necessário para não danificar o seu sistema.

Lembramos que, em ambiente de produção, não é necessário que o usuário tenha direitos de administrador (aliás o contrário e recomendado).

A partir da versão 4.0.5, é possível compilar e instalar todos as bibliotecas, drivers e aplicativos através do comando: `make; make install; make config` a partir do diretório raiz.

Após a execução destes comandos, o ambiente já estará pronto para operar. Um teste de verificação útil é a execução do programa vlib_diag que encontra-se no diretorio ../samples/dll_so.

Nos próximos itens deste capítulo será explicado a compilação e instalação de cada módulo separadamente, porém a sua execução só será necessária caso haja algum problema no passo que acabou de ser explicado.

Copyright © 2010 Digivoice Eletrônica

Módulos do kernel

Os módulos ou device drivers são os responsáveis em criar e gerenciar a troca de dados entre o hardware e os aplicativos. Caso eles não estejam funcionando corretamente, nada mais funcionará.

No sub-diretório *../driver/linux* encontram-se todos os arquivos necessários para compilar e instalar os módulos. O mesmo device driver é responsável por gerenciar todos os modelos de placas.

Para compilar o módulo desejado, você precisa executar os seguintes comandos:

`make`: compila o módulo

`make install`: instala o módulo no diretório correto e carregá-lo na memória

`make config`: instala o módulo na inicialização do sistema - testado em Debian, Suse, RedHat e derivações.

Após a compilação (`make`) deverá ser gerado o arquivo *vlibd.o* ou *vlibd.ko*

Para carregar o módulo na memória manualmente depois de reiniciar (se não tiver sido executado o `make config`), execute novamente o `make install`.

Para verificar se o módulo foi instalado corretamente, digite em linha de comando '`dmesg`'. Este programa exibe informações dos device drivers instalados. Se tudo estiver correto, você verá uma mensagem como a apresentada a seguir:

```
vlibd: VoicerLib Device Driver was loaded successfully!
```

Se a mensagem exibida for diferente, aparentando algum tipo de erro, verifique e repita os passos anteriores ou mesmo se a placa

Copyright © 2010 Digivoice Eletrônica

está fisicamente instalada.

Usuários *RED HAT 3.2 AS/ES* ou *CentOS-3*, para compilar o módulo **executem**: `make RH3=1`.

Os outros procedimentos não são alterados.

Copyright © 2010 Digivoice Eletrônica

Compilando a VoicerLib

Como a VoicerLib é multiplataforma, existem diretórios específicos para plataforma Linux e Windows. Para compilar a VoicerLib para Linux, vá para o diretório *../src_linux* e execute os seguintes comandos:

```
make
make install
```

O comando `make install` copiará o arquivo resultante *(libdigivoice.so.4.x.x.x)* para o diretório */usr/lib* e o tornará disponível para todas as ferramentas de desenvolvimento como uma shared object.

Também copia os headers (*.h) para o diretório padrão */usr/include/voicerlib* permitindo o uso da voicerlib por outros aplicativos. Os arquivos de configuração e firmware são instalados em */var/lib/voicerlib*. Todos os aplicativos de teste irão buscar os arquivos de firmware neste diretório como padrão.

O funcionamento da VoicerLib será tratado no capítulo Guia de Programação e seus comandos estão listados no capítulo Guia de Referência.

Copyright © 2010 Digivoice Eletrônica

Exemplo em linguagem C

O programa no diretório *../samples/dll_so/c* chama-se vlib_diag.

Trata-se de um programa em modo caracter (console) que permite interagir com as placas DigiVoice. É a melhor forma de testar o funcionamento da placa bem como estudar o funcionamento da API.

Este projeto necessita da biblioteca ncurses-devel com versão maior ou igual a 5.3.

Para compilar, digite `make`.

Ao executar *./vlib_diag* você poderá efetuar diversos testes com as placas E1, FXO, FXS e GSM.

Copyright © 2010 Digivoice Eletrônica

A VoicerLib 4 pode ser utilizada tanto no ambiente Windows como no Linux, sendo que existem semelhanças e algumas diferenças no modo de funcionamento da biblioteca nas duas plataformas.

Semelhanças

A VoicerLib tem uma camada de funções primitivas que foram implementadas na forma de uma API em C. Em Linux, utiliza-se Shared Objects e em Windows DLLs para interfacear com as aplicações. Desta forma, apesar de ter sido construída em C, a VoicerLib API permite sua utilização em qualquer ferramenta de desenvolvimento que tenha acesso à DLLs ou SOs. Para isso, é necessário que os headers da VoicerLib API sejam traduzidos de C para a linguagem desejada.

Na camada de API, a VoicerLib fornece todos os métodos primitivos de acesso aos recursos da placa. Entenda por métodos primitivos, todas as funções que não se enquadram na classe de funções especiais, discutidas mais adiante.

Diferenças

A DigiVoice fornece, somente para ambiente Windows, um ActiveX chamado VoicerLib.OCX, que tem por objetivo facilitar a programação. Nesse conjunto, estão incluídas todas as funções primitivas e especiais (MakeCall, PlayList, etc...).

A seguir serão mostrados os conceitos específicos de cada modo de programação. Mais a frente no manual, os conceitos diversos serão explicados independente do uso da API ou do ActiveX, sempre sendo feito menção dos métodos e/ou funções utilizadas em cada modo.

Se você programa em Linux ou pretende programar em C para Windows acessando as funções primitivas, leia somente as descrições de Conceitos Básicos - API. Caso a plataforma utilizada seja Visual Basic, Delphi, etc... recomenda-se a utilização do ActiveX, por isso leia as descrições de Conceitos Básicos - ActiveX.

Copyright © 2010 Digivoice Eletrônica

Para facilitar a documentação, as explicações das funções API e ActiveX são feitas conjuntamente, sempre colocando a função API e o método ActiveX entre parênteses, salvo situações em que as diferenças são explícitas.

Conceitos Básicos - API

Para os desenvolvedores acostumados com a VoicerLib em Windows, os conceitos predominantes são as propriedades, eventos e métodos, comuns à linguagem orientada a eventos como o Visual Basic. Como a VoicerLib API foi desenvolvida para ser um Shared Object (Linux) e uma DLL (Windows), o funcionamento é um pouco diferente.

Numa tradução simplificada, as propriedades e métodos serão acessados através de funções comuns com nome e parametrização similar à versão em Windows (ex.: o método PickUp tornou-se a função dg_pickup). Todas as funções da VoicerLib API começam com "dg_" para evitar conflitos de nomes com outras bibliotecas dinâmicas que por ventura existam no ambiente.

Os eventos existentes no ActiveX em Windows são chamados pela VoicerLib de forma transparente, bastando ao programador associar uma função a um determinado evento (ex.: OnRingDetected -> TrataRing). Em um shared-object os eventos continuam existindo, porém, a rotina que associa determinado evento (ou mensagem) a um bloco de código (ou função) passa a ser de responsabilidade do programador.

Na prática, é necessário criar uma função que receba todos os eventos da placa. Esta função será chamada de Gerenciador de Eventos**.** Um "esqueleto" dessa função é mostrado a seguir:

Copyright © 2010 Digivoice Eletrônica

```
void ReceiveEvents(void *context_data)
{
  struct dg_event_data_strucuture *EventContext;

    /* Copy received Data */
    EventContext = ((struct
dg_event_data_strucuture*)context_data);

     switch (EventContext->command)
       {
              case EV_RING:
                     //recebeu ring
                     //....
              case EV_DTMF:
                     //recebeu digito
                     //...
       }
}
```

Mesmo conhecendo pouco a linguagem C, é fácil de perceber que o Gerenciador de Eventos recebe um parâmetro chamado *context_data*. Neste parâmetro estarão as informações pertinentes a cada evento (porta, dígito, etc...). Caberá ao programador analisar o conteúdo de *context_data->command* para saber qual evento ocorreu. A variável *context_data->port* indica a porta que ocorreu o evento e a variável *context_data->data* fornece um dado relativo ao evento (alguns eventos não possuem dados associados).

Context_data é definido como:

```
struct  dg_event_data_strucuture *context_data;
```

Sendo que *dg_event_data_strucuture* está definido no arquivo *voicerlib.h* conforme segue:

```
typedef struct {
       unsigned short command;
       unsigned short data;
       unsigned short port;
       unsigned short data_aux;
       unsigned short card;
} dg_event_data_structure;
```


Copyright © 2010 Digivoice Eletrônica

Um primeiro conceito que já é conhecido dos desenvolvedores Windows e que foi mantido na versão Linux é que todo o funcionamento da VoicerLib é assíncrono. Isto significa que ao executar a chamada a uma determinada função, o programa seguirá seu fluxo normal. As respostas às funções são manipuladas através de uma função eleita para tratar todas as mensagens, que no exemplo é a ReceiveEvents.

Por exemplo, quando a função dg_GetDigits é chamada, o programa continua normalmente e só após os dígitos serem recebidos (ou der time-out) a VoicerLib chamará a função ReceiveEvents passando no context_data a mensagem EV_DIGITSRECEIVED e o status de retorno da função.

Este tipo de funcionamento é muito importante pois a biblioteca tem que gerenciar várias portas simultaneamente. Se a aplicação ficasse presa na execução de uma função, não conseguiria tratar os outros eventos das outras placas, por isso o tratamento de determinada mensagem dentro do *switch-case* deve ser o mais rápido possível.

A função ReceiveEvents na verdade pode ter qualquer nome mas sempre deverá receber o endereço da estrutura context_data, independente da linguagem de programação utilizada.

No início do programa, antes de iniciar a placa com o método dg_StartVoicerLib, é obrigatório associar a função de tratamento de mensagens à VoicerLib através da função dg_SetEventCallback**.** Se a associação não for feita corretamente, o programa gerará falhas de execução (Segmentation Fault).

Então, um esqueleto de programa utilizando a VoicerLib API deverá ter a seguinte característica:

```
void ReceiveEvents(void *context_data)
{
       //Tratamento de todos os eventos vindos da placa
}

void main() //Inicio do programa
{
```


Copyright © 2010 Digivoice Eletrônica

```
	//Primeiro associa a função callback
	dg_SetEventCallback(ReceivedEvents,&event_context
);

	//Inicia a voicerlib
	dg_StartVoicerLib("../../firmware");
}

void finaliza()
{
	//Finaliza o driver
	dg_ShutdownVoicerlib();
}
```

Consulte o Guia de Referência VoicerLib API a respeito de todas as funções disponíveis.

As placas E1 têm um comportamento diferente em relação à vários tópicos abordados neste capítulo, porém, a DigiVoice procurou deixar a manipulação destas características da maneira mais similar às placas FXO, visando minimizar o tempo de aprendizado. A leitura do tópico Programação da Placa E1 é de extrema importância para a compreensão destas diferenças e do modo de operação a serem utilizados.

Copyright © 2010 Digivoice Eletrônica

Conceitos Básicos - ActiveX

O componente VoicerLib é baseado na estrutura de um looping infinito, que fica monitorando os eventos que acontecem no hardware (ring, tons, etc...) e recebendo e passando os comandos gerados a partir da aplicação (propriedades e métodos).

Devido a esta característica, a maioria das funções da biblioteca são assíncronas. Isto significa que ao executar a chamada a um determinado método, o programa seguirá seu fluxo normal. As respostas aos métodos são manipuladas através de eventos específicos relativos a cada acontecimento.

Por exemplo, quando o método GetDigits é chamado, o programa continua normalmente e só após os dígitos serem recebidos (ou der time-out) que a resposta ao GetDigits será tratada dentro do evento OnDigitsReceived

Este tipo de funcionamento é muito importante pois a biblioteca tem que gerenciar várias portas simultaneamente.

Se a aplicação ficasse presa na execução de um método, não conseguiria tratar os outros eventos das outras portas. Lembre-se que é possível que uma porta esteja reproduzindo a mensagem enquanto outra esteja recebendo um ring.

Observações importantes para reinstalar a OCX da VoicerLib

Quando a versão da voicerlib for atualizada, é necessário que a aplicação seja recompilada.

Em Delphi7: Apague os arquivos de nome "VoicerLib_TLB" no diretório "*\%PROGRAM FILES%\Borland\Delphi7\Imports",* importe novamente a OCX no Delphi7*: Component -> Import ActiveX* -> Localize *Digivoice VoicerLib ActiveX* -> clique em *Install*...

Em Visual Basic 6: No VB6 -> botão direito na barra lateral

Copyright © 2010 Digivoice Eletrônica

General->Components->Controls-> Selecione Digivoice->*Browse* -> *\windows\system32\voicerlib.ocx* -> Aplicar ->OK .

Guia de Migração de Versões Anteriores

Apesar de ter como objetivo manter a biblioteca o mais compatível possível com as versões anteriores, algumas alterações nos fontes serão necessárias para atualizar aplicações para a nova versão, principalmente devido às novas tecnologias empregadas nas placas FXO e E1. Essas alterações fizeram com que novos métodos fossem criados, outros fossem excluídos e alguns outros tiveram alterações nos parâmetros que recebem e enviam.

Recomendamos a leitura deste capítulo antes de efetuar a adaptação de sistemas desenvolvidos com versões anteriores da VoicerLib.

1. Métodos/Propriedades removidas

- **SetAnswerSensitivity** – Agora é um parâmetro do método ConfigCallProgress (Maiores detalhes no capítulo Supervisão de Linha).
- **SetAnswerThreshold** - Esse tipo de configuração não é mais necessária (veja também SetSilenceThreshold).
- **SetVolume** - Sem função nas novas placas - exclusivo para placa de 1 canal com headset.
- **SetImpedance** - Sem função nas novas placas - exclusivo para placa de 1 canal com headset.
- **SetDTMFAttenuatingHigh/Low** - A configuração de ganho de dígitos é mais ampla e feita pelo método SetDigitGain.
- **SetDTMFTwist/SetToneTwist** - Devido aos novos algorítmos não é mais necessária a configuração de twists.
- **Propriedade CardType** - Como agora é permitida a mistura de placas diferentes, não é possível utilizar uma propriedade para atribuir ou ler o tipo de placa.

Copyright © 2010 Digivoice Eletrônica

- **SetFrequencyTime** - Agora é um parâmetro do método ConfigCallProgress (Maiores detalhes no tópico Supervisão de Linha).
- **AutoClearDigits** - A propriedade foi retirada por causar certa confusão no funcionamento dos métodos que detectavam dígito. Agora o programador deverá sempre apagar explicitamente os dígitos com o método ClearDigits, sempre que for necessário.
- **Formato ffSig -** Está sendo removido gradualmente por ser idêntico em tamanho ao ffWaveULaw ou ffWaveALaw. Para manter a compatibilidade, ao escolher este formato, será utilizado no lugar o ffWaveULaw.
- **Propriedades DelayDot**, **DelayComma** e **DelaySemicolon** - Foram removidas, agora é necessário utilizar o método SetDialDelays.
- **ReadEEPROM** - Foi removida e ao invés dela, foi criado um método CheckCode.

2. Mudanças de nome e/ou parâmetros

- **SetFastDetection** - Agora para setar o modo rápido, é preciso chamar este método (o parâmetro enable pode receber os valores DG_ENABLE /DG_DISABLE).
- **EnableDetections -** Mudou para SetDetectionType, agora a voicerlib não detecta nenhum tipo como padrão. O programador deve chamar explicitamente este método sempre quando for necessário. Os métodos de detecção mudaram bastante portanto leia atentamente o guia de referência sobre este método.
- **Evento OnAswerDetected** - Recebe um novo parâmetro chamado AnswerType, indicando se o atendimento foi por detecção de áudio (AUDIO_DETECTED) ou timeout (TIMEOUT_DETECTED).
- **SetCardSyncMode para placas E1** - Para setar o sincronismo deve ser chamado o SetCardSyncMode passando como parâmetros a placa e o tipo de sincronismo SYNC_INTERNAL, SYNC_LINE_A, SYNC_LINE_B.
- **dg_StartVoicerLib** - Na DLL e no SO só tem o path como parâmetro (Isso não é aplicado ao ActiveX) e agora retorna DG_EXIT_SUCCESS (0) no caso de sucesso e não mais 99.

Copyright © 2010 Digivoice Eletrônica

- **PlayBuffer** - Parâmetro remaining_size incluso no Active X.
- **WriteEEPROM -** Foi renomeada para WriteCode.
- **ConfigGSMThread -** O segundo parâmetro possui mais possibilidades de configuração: GSMCFG_USSD_ENABLE, GSMCFG_CALL_WAITING_ENABLE, GSMCFG_RETRY_TIMEOUT e GSMCFG_ANSWER_TIMEOUT. Consulte o método ConfigGSMThread.
- **GSMClearAllGSM** - Incluso segundo parâmetro que indica se todas as mensagens do módulo serão ou não apagadas. Consulte o método GSMClearAllSMS.
- **ALARM_RESERVED -** A constante foi renomeada para ALARM_CRC4SYNC. Verifique em OnE1Alarm.
- **EnableMailBoxDetection** - Incluso novo parâmetro *silencetime.*
- **OnE1Alarm -** Quando o protocolo for ISDN PRI, há mais um tipo de alarme ALARM_LINK (OnE1Alarm).

3. Novas funções/eventos

- **Evento OnAudioSignal -** Indicando para a aplicação o tipo de áudio detectado. Útil para análises de tipos de tom e suas cadências.
- **Funções de canais virtuais** - Permitindo utilizar a numeração de portas independentemente da posição física da porta na placa.
- **DefinePortResource** – Associa uma determinada porta (lógica) a uma placa/porta física.
- **ResetPortResource** – Reseta a configuração, assumindo o padrão de inicialização das funções de callprogress **-** O CallProgress agora é mais completo e versátil. Leia atentamente o capítulo de Supervisão de Linha - Funções relacionadas: CreateCallProgress, ConfigCallProgress, DestroyCallProgress e EnableCallProgress.
- **EnableAnswerDetection/DisableAnswerDetection** – O método para habilitar deve ser chamado explicitamente após a chamada do EnableCallProgress, para que a thread de controle de callprogress gere o evento de atendimento.
- **EnableInputBuffer/DisableInputBuffer** – Agora para efetuar a gravação de portas na voicerlib, é necessário habilitar o envio

Copyright © 2010 Digivoice Eletrônica

de amostras da placa através do método EnableInputBuffer. Isso deve ser feito antes de chamar o RecordFile. Esta alteração foi feita para suportar streaming de áudio das placas para as aplicações diretamente.

- **PlayBuffer / StopPlayBuffer -** Envia amostras diretamente para a placa reproduzir. Útil para aplicações VoIP.
- **SetGSMMode** - Pode assumir GSM_DIGIVOICE ou GSM_RAW (padrão). Ambos os formatos têm a mesma codificação sendo que, no padrão Digivoice é inserido um cabeçalho no arquivo.
- **GetPortCardType** - Devolve o tipo de placa a partir da porta informada.
- Implementada funcionalidade de **detecção de silêncio**, através dos métodos EnableSilenceDetection/DisableSilenceDetection e do evento OnSilenceDetected.
- **GsmToWave** - Converte arquivo de áudio do formato GSM para o formato Wave.
- **GsmToWave49** - Converte arquivo de áudio do formato GSM para o formato Wave (GSM 6.10 modificado).
- **GsmRawToWave** - Converte arquivo de áudio do formato GSMRaw (sem cabeçalho) para o formato Wave.
- **GsmRawToWave49** - Converte arquivo de áudio do formato GSMRaw (sem cabeçalho) para o formato Wave49 (GSM 6.10 modificado).
- **WaveToGsm** - Converte arquivo de áudio do formato Wave para o formato GSM.
- **WaveToGsmRaw** - Converte arquivo de áudio do formato Wave para o formato GsmRaw (sem cabeçalho).
- **Wave49ToGsm** - Converte arquivo de áudio do formato Wave49 para o formato GSM.
- **Wave49ToGsmRaw** - Converte arquivo de áudio do formato Wave49 para o formato GSMRaw (sem cabeçalho).
- **SetFXCardType** - Permite configurar o tipo da porta (FXS/FXO) da placa VB0404FX.
- **ReturnCodeToString -** Retorna qual a mensagem de erro, a partir do código de retorno.
- **CheckCode -** Compara a string de segurança da memória da placa com a string a ser gravada.
- **SetLoggerSilenceThreshold** - Permite setar o limiar de silêncio para as portas da placa VB6060 no caso de aplicações utilizando a thread de Logger.

Copyright © 2010 Digivoice Eletrônica

- **GSMCheckSignalQuality -** Este comando força o módulo GSM correspondente a porta especificada, verificar o nível de sinal recebido na antena.
- **GSMGetMessage -** Este comando obtém a mensagem SMS recebidas pelo módulo. Método usado para fins de debug.
- **GSMGetSignalQuality -** Recupera a string com o valor da qualidade do sinal após o recebimento do evento OnGSMSignalQuality .
- **GSMGetSMS -** Permite ler uma mensagem SMS recebida.
- **GSMSendCommand -** Envia um comando ao módulo GSM correspondente à porta especificada.
- **GSMSendSMS -** Permite o envio de mensagem SMS nas portas correspondentes.
- **GSMSetPinNumber -** Ao ser iniciada a thread GSM a VoicerLib consulta os módulos GSM sobre a necessidade ou não do envio de PIN number ou PUK.
- **OnGSMError -** Ocorre quando um erro é detectado na porta ou na troca de sinalização nos módulos GSM.
- **OnGSMMessage -** Ocorre quando uma mensagem é detectada pela placa.
- **OnGSMReady -** Ocorre quando a thread GSM foi criada e inicializada com sucesso.
- **OnGSMSignalQuality -** Ocorre em resposta ao método GSMCheckSignalQuality, informa que o valor respectivo a qualidade do sinal já está disponível.
- **OnGSMSMSReceived -** Ocorre quando uma mensagem SMS foi recebida e está disponível para leitura através do método GSMGetSMS.
- **OnGSMSMSSent -** Ocorre ao término do envio de uma mensagem SMS na porta especificada.
- **OnGSMTimeout -** Ocorre quando um comando enviado pela Voicerlib para o módulo GSM correspondente a porta da placa VB0404GSM não obtém uma resposta no tempo especificado no método ConfigGSMThread.
- **GSMClearAllSMS -** Este comando apaga todas as mensagens SMS do módulo GSM correspondente a porta especificada.
- **GSMDeleteSMS -** Apaga a mensagem SMS do módulo GSM.
- **GSMListSMS -** Solicita ao módulo GSM a lista de mensagens GSM armazenadas em sua memória.
- **GSMGetIndexList -** Recupera a lista dos índices das

Copyright © 2010 Digivoice Eletrônica

mensagens SMS do módulo GSM.

- **GSMReadAndDeleteSMS -** Este comando lê a mensagem SMS especificada pelo índice e em seguida apaga a mensagem do módulo GSM.
- **GSMRestartPort -** Este comando reinicia o módulo especificado pela porta.
- **OnGSMReturnOK -** Evento gerado após a execução dos métodos GSMListSMS (status GSM_LIST) e GSMClearAllSMS (status GSM_CLEAR).
- **GSMCallControl** - Este comando executa as funções de atendimento de segunda chamada e conferência nos módulos.
- **OnGSMOtherCall -** Quando o módulo GSM já atendeu uma chamada e recebe uma "outra chamada" este evento é gerado.
- **GSMGetLastCommand** - Este método obtém o último comando enviado para o módulo. Deve ser chamado no evento OnGSMMessage.
- **GSMGetMemory -** Esse método disponibiliza ao usuário a quantidade de mensagens SMS no módulo GSM após o recebimento do evento OnGSMMemory.
- **OnGSMMemory -** Ocorre em resposta ao método GSMClearAllSMS, informa que a quantidade de mensagens armazenadas no módulo GSM está disponível para ser lida com o método GSMGetMemory.
- **OnGSMMemoryFull -** É gerado quando o módulo GSM atinge sua capacidade máxima de mensagem. Utilize o método GSMClearAllSMS para limpar as mensagens do módulo.
- **EnableFSKDetetection -** Habilita detecção de identificação de assinante no padrão FSK.
- **GetNameID -** Recupera o nome do assinante chamador quando diponível no sistema FSK ou quando o protocolo for ISDN PRI.
- **GetLibVersion -** Informa o número de versão da OCX ou da DLL.
- **CreateLoggerCCS** - Inicia o tratamento automático de gravação em paralelo para sinalização ISDN PRI.
- **DestroyLoggerCCS** - Finaliza o tratamento automático de gravação em paralelo para sinalização ISDN PRI.
- **ConnectAudioChannels -** Efetua uma conexão bi-direcional entre duas portas de qualquer placa.
- **DisconnectAudioChannels -** Efetua uma desconexão de duas portas conectadas de qualquer placa.

Copyright © 2010 Digivoice Eletrônica

- **EnableMailBoxDetection -** Habilita detecção de caixa postal (secretária eletrônica).
- **DisableMailBoxDetection -** Desabilita detecção de caixa postal (secretária eletrônica).
- **OnMailBoxDetected** - Evento gerado quando detecta caixa postal (secretária eletrônica).
- **GSMGetSMSConfirmation -** Permite ler a string de confirmação do recebimento de uma mensagem SMS por parte do destinatário. Este método deve ser chamado no evento que confirma
  o recebimento de uma mensagem SMS por parte do destinatário (OnGSMSMSConfirmation).
- **OnGSMSMSConfirmation -** Permite receber a confirmação do recebimento de uma mensagem SMS por parte do destinatário (este recurso depende da disponibilidade da operadora).
- **OnGSMSIM -** Ocorre na inserção ou retirada do SIM Card (chip) nos módulos GSM, permitindo por exemplo que o programador reinicie a thread GSM após a inserção de um chip na placa.
- **AudioMonitor -** Monitora em tempo real o áudio de uma porta específica na saída de áudio do PC.
- **GetCardInfo -** Informa dados sobre a placa.
- **ConfigISDNThread -** Configura as opções de tratamento da thread em relação ao protocolo ISDN PRI.
- **CreateISDNThread -** Inicia o tratamento do protocolo ISDN PRI de troncos digitais.
- **DestroyISDNThread -** Termina o tratamento do protocolo ISDN PRI.
- **DisableISDNThread -** Desabilita o tratamento do protocolo ISDN PRI de troncos digitais.
- **EnableISDNThread -** Habilita o tratamento do protocolo ISDN PRI.
- **GetBusyCause -** Retorna a causa do desligamento da chamada em caso de ocupado.
- **GetISDNThreadStatus -** Retorna o status da thread ISDN da placa e span especificados.
- **GetNameID -** Recupera o nome do assinante chamador quando diponível no sistema FSK e no protocolo ISDN PRI.
- **GetText -** Retorna uma mensagem de texto recebida pela VoicerLib.
- **ReturnCauseToStringCauseHangUp -** Retorna qual a

Copyright © 2010 Digivoice Eletrônica

descrição da causa de ocupado, a partir do código de retorno.

- **SendISDNCommand -** Envia um comando ISDN referente a estabelecimento de chamadas entrantes.
- **SetBearCapability -** Configura o tipo da chamada do span.
- **SetCalledNumber -** Configura as propriedades do número a ser discado.
- **SetCallingNumber -** Configura as propriedades do número do identificador de A (BINA).
- **SetHangUpCause -** Configura a causa do desligamento da porta no protocolo ISDN PRI.
- **SetNameID -** Especifica o nome da identificação de uma porta no tronco digital quando se utiliza protocolo ISDN PRI.
- **SetReverseCharging -** Solicita a inversão de cobrança para chamada sainte.
- **SetText -** Especifica uma mensagem de texto a ser enviada pela VoicerLib por uma porta no tronco digital no protocolo ISDN PRI.
- **OnChannel -** Indica a mudança de porta de uma chamada.
- OnErrorNodeConfig - Ocorre quando há erro de configuração no ISDN PRI
- **OnNameID -** Ocorre quando a placa detecta a descrição do identificador da chamada (nome).
- **OnReverseCharging -** Ocorre sempre quando a chamada entrante requisitar a inversão de cobrança da chamada no protocolo ISDN PRI(Ligação a cobrar).
- **OnText -** Ocorre sempre que um texto for recebido pela aplicação (VoicerLib) no protocolo ISDN PRI.

Copyright © 2010 Digivoice Eletrônica

Instruções de instalação de placas

Instalação das placas Digivoice em Windows Vista e Windows 2008.

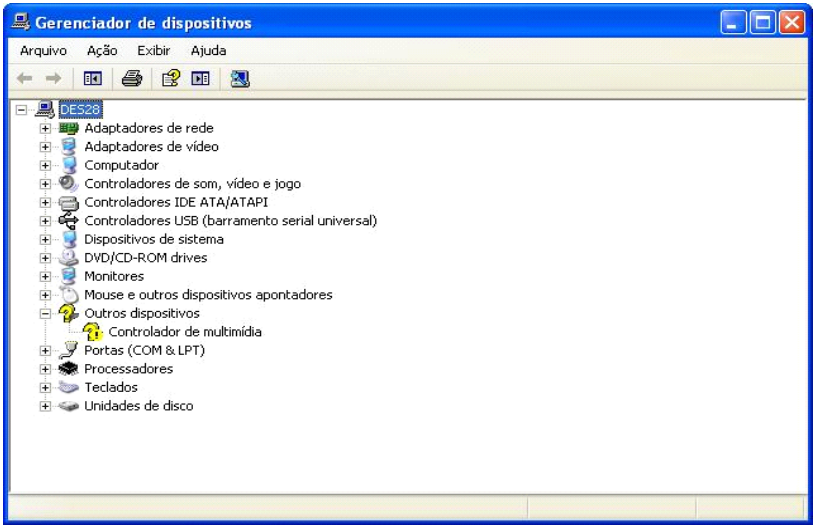
Devido às exigências dos novos sistemas operacionais da Microsoft (Windows Vista e Windows 2008) e para manter compatibilidade com produtos anteriores as placas Digivoice com barramento PCI, ao instalar as placas nos sistemas operacionais Windows Vista ou Windows 2008, é necessário fazer uma atualização no hardware feita por uma configuração em uma memória FLASH das placas. Isto é feito por programas fornecidos pela Digivoice (VlibUpdate.exe). Caso seja necessário fazer um downgrade do sistema operacional, por exemplo de Vista para XP é necessário desfazer a alteração com o programa VlibUpdateRetro.

Esta documentação se aplica às placas:

VB0408PCI, VB0408PCIe, VB3030PCI, VB3030PCIe, VB6060PCI, VB6060PCIe, VB0404GSM, VB1200PCIe e VB0404FX.

Após executar o setup da VoicerLib, verifique se a placa foi corretamente instalada da seguinte forma:
- Em Painel de Controle - Sistema - Gerenciador de Dispositivos, a placa deve reconhecer o driver da digivoice "dgdriver" e a(s) placa(s) instalada(s).
- Caso isso não ocorra, clique com o botão direito do mouse no dispositivo "Controlador de multimídia" na opção "Atualizar driver" e indique o caminho de instalação da Voicerlib. Geralmente em C:\%PROGRAM FILES%\Voicerlib4.

Copyright © 2010 Digivoice Eletrônica



Gerenciador de Dispositivos - Painel de Controle

Caso 1: Computador com sistema operacional Windows XP/2000/2003.

Com as placas plugadas e devidamente parafusadas no slot, o setup da Voicerlib é instalado.
Após a instalação, verifique se as placas foram detectadas e instaladas corretamente.

A cada placa adicionada nesse ambiente (sistema operacional continuando o mesmo) depois da instalação, a verificação e instalação no Gerenciador de Dispositivos deve ser feita.

Caso 2: Computador com sistema operacional Windows Vista/2008.

Com as placas plugadas e devidamente parafusadas no slot, o setup da Voicerlib é instalado.
Após a instalação, verifique se as placas foram detectadas e instaladas corretamente.

Copyright © 2010 Digivoice Eletrônica

A cada placa adicionada nesse ambiente, após a instalação da VoicerLib, (sistema operacional continuando o mesmo) será necessário executar o arquivo VlibUpdate.exe que está geralmente no diretório "C:\%PROGRAM FILES%\VoicerLib4\VlibUpdate". Lembrando que o arquivo deve ser executado com privilégio de administrador da máquina (botão direito - executar como....administrador).

O computador deve ser reiniciado e a placa instalada.

Caso 3: Adicionar uma ou mais placas Digivoice em um computador com XP/2000/2003 que estavam instaladas e funcionando em um computador com Windows Vista/2008.

Executar o arquivo "VlibUpdateRetro.exe" que está geralmente no diretório: "C:\%PROGRAM FILES%\VoicerLib4\VlibUpdateRetro".

O computador deve ser reiniciado e a placa instalada.

Caso 4: Adicionar uma ou mais placas Digivoice em um computador com Vista/2008 que estavam instaladas e funcionando em um computador com Windows XP/2000/2003.

Executar o arquivo "VlibUpdate.exe" que está geralmente no diretório: "C:\%PROGRAM FILES%\VoicerLib4\VlibUpdate". (Ressaltando que essa ação deve ser feita com privilégio de administrador).

O computador deve ser reiniciado e a placa instalada.

Copyright © 2010 Digivoice Eletrônica

Inicializando os Serviços

Sempre que iniciar a aplicação, antes de executar qualquer outra função, é necessário iniciar os serviços (device driver). Isto pode ser feito através do método StartVoicerLib.

A VoicerLib detecta todas as placas disponíveis no computador e as inicializa na ordem de colocação dos slots do computador. Com isso é possível ter, por exemplo, uma placa E1 de 60 canais e uma placa analógica de 8 canais. Se a placa E1 for lida primeiro, receberá as portas de 1 a 60 e a placa analógica de 61 a 68. A posição física dos slots dos computadores variam de fabricante, modelo e sistema operacional, portanto é necessário que a configuração na máquina de produção seja versátil para se adaptar a essas diferenças.

Em um mesmo computador, uma vez instaladas, as placas sempre serão reconhecidas na mesma ordem.

Exclusivo API: Como a API não gera eventos, como conhecido nos componentes visuais, é necessário criar uma função para manipular os eventos da placa, conforme foi explicado no tópico Conceitos Básicos. Lembre-se que a associação da Callback de eventos deverá ser feita antes de inicializar o driver.

Como a inicialização é a primeira coisa a ser feita, recomenda-se sua colocação no início da aplicação. Não é prática recomendável ficar inicializando e finalizando a VoicerLib diversas vezes na aplicação.

Copyright © 2010 Digivoice Eletrônica

Finalizando os Serviços

Antes de fechar a aplicação é necessário chamar o método ShutdownVoicerLib, que finaliza todos os serviços e reseta a placa.

A não utilização deste método poderá causar travamento no sistema operacional ou algum comportamento imprevisível.

É possível verificar se os serviços do hardware foram finalizados corretamente através do retorno do método ShutDownVoicerLib, que retorna DG_EXIT_SUCCESS em caso de sucesso.

O melhor local para se colocar este método é em algum evento finalizador, que antecede o encerramento da aplicação (Unload, OnClose, FormClose etc...).

Copyright © 2010 Digivoice Eletrônica

Detectando Ring

A VoicerLib informa quando tem uma ligação entrante chegando através do ring.

Sempre quando o ring for detectado, a função que atua como Gerenciador de Eventos receberá na variável context_data->command o valor EV_RINGS (evento OnRingDetected no ActiveX).

Nas placas FXO e GSM, como o ring ocorre em situações bem específicas a sua detecção está sempre habilitada não havendo métodos para ligar ou desligar.

Já na placa E1, o ring é um evento que ocorrerá quando a thread de controle (R2D MFC ou ISDN PRI) estiver habilitada.
Para maiores detalhes, veja o capítulo Programação da Placa E1.

Copyright © 2010 Digivoice Eletrônica

Atendendo e Desligando

Para atender a ligação ou tomar uma linha para discagem, o método que deve ser utilizado é o Pickup . O método pede 2 parâmetros. O primeiro é a porta de atendimento que vai de 1 a N. O segundo parâmetro é uma pausa após o atendimento, em milissegundos (1000 = 1 segundo). Esta pausa é útil em aplicações de atendimento automático em instalações com bloqueio de chamada a cobrar, etc... Ela deve ser utilizada em conjunto com o evento OnAfterPickUp.

Para desligar a ligação, o método utilizado é o HangUp. Deve ser passado somente a porta que se deseja desligar. Ambas as funções retornam DG_EXIT_SUCCESS caso tenham sido executadas com sucesso.

Um diferencial em relação aos modelos anteriores é que quando a ligação é atendida ou desligada os eventos OnLineReady e OnLineOff são gerados mesmo que a linha/ramal não esteja em paralelo.

A placa E1 é um caso especial, pois o PickUp refere-se a solicitação de ocupação de uma porta. Quando executado no contexto da thread (CreateE1Thread, CreateISDNThread), o pickup é utilizado para ocupar uma porta na ligação sainte ou atender uma ligação entrante. Se a thread não estiver iniciada, o sinal de ocupação também é enviado, porém todo o tratamento deverá ser feito pela aplicação final.

Copyright © 2010 Digivoice Eletrônica

Supervisão de Linha

A supervisão de linha permite ao programador identificar e tratar diversos eventos relativos a sinais enviados pela linha, a saber:

- Sinal de Ocupado
- Detecção de Fax
- Detecção de sinal de discagem (tom de linha)
- Detecção de sinal de chamada (ringback)
- Detecção de Atendimento

Nas placas digitais (E1) não existe propriamente um sinal de ocupado, tom de discagem ou sinal de chamada já que toda a parte de sinalização é feita pelo protocolo R2D, porém, a VoicerLib simula estes sinais e eventos para deixar a programação similiar às placas FXO. Isso só é possível se utilizar as threads de controle (ConfigE1Thread).

Acionamento das detecções nas placas FXO

Na VoicerLib 4, a supervisão de linha foi implementada através de threads de controle que são criadas através do método CreateCallProgress. A thread de Call Progress é responsável pelo acompanhamento de chamadas, ou seja, a supervisão de um tom genérico, de tons de linha, de chamar (ringback), de ocupado, de fax e o atendimento pelo assinante chamado. O método CreateCallProgress cria a thread de controle e em seguida dorme portanto pode ser criado no início da aplicação sem prejuízo de processamento, devendo ser chamada para cada porta específica.

Todos os parâmetros de configuração da thread de Call Progress ficam armazenados em um arquivo texto de configuração localizado na pasta firmware (Linux) ou \%PROGRAM FILES%\VoicerLib4 (Windows). O arquivo padrão é o cp_default.cfg porém o desenvolvedor poderá criar outros de acordo com suas necessidades, já que o nome do arquivo a ser utilizado é passado no método CreateCallProgress.

Copyright © 2010 Digivoice Eletrônica

Também é possível configurar o CallProgress sem a utilização do arquivo. Todos os parâmetros podem ser configurados através do método ConfigCallProgress pois cada item do arquivo de configuração tem sua constante correspondente. Veja todas estas opções no Guia de Referência, no método ConfigCallProgress**.**

Depois da criação da thread de controle, é necessário chamar o método EnableCallProgress para efetivamente monitorar a linha. O acompanhamento de chamadas foi dividido em 4 opções básicas para agilizar sua utilização e economizar recursos de hardware:

1 - CP_ENABLE_GENERIC_TONE - Desenvolvida para detectar a presença de um tom qualquer pré-configurado ou o atendimento de chamada, esta facilidade supervisão de atendimento pode ser habilitada na VoicerLib pelo método EnableAnswerDetection.

2 - CP_ENABLE_LINETONE_OR_BUSY - Desenvolvida para a detecção rápida de tom de discar (tom de linha) ou ocupado antes do início de uma discagem ou após um Flash.

3 - CP_ENABLE_BUSY_OR_FAX - Desenvolvida para a detecção rápida de tom de ocupado ou sinal de fax após o atendimento de uma chamada.

4 - CP_ENABLE_ALL - Desenvolvida para a detecção de tons de discar, de chamada, de ocupado, de fax e atendimento entre uma discagem e o atendimento pelo assinante chamado.

A princípio, esta opção atende qualquer situação, mas devido às restrições das várias cadências dos tons, a discriminação de um determinado tom, pode demorar mais tempo que em uma opção específica como a CP_LINETONE_OR_BUSY. Isso significa que o desenvolvedor poderá simplificar sua aplicação sempre habilitando o callprogress utilizando esta opção, mas terá uma detecção mais rápida e eficiente se utilizar a opção específica para cada situação.

Copyright © 2010 Digivoice Eletrônica

Então, como sugestão, o desenvolvedor pode utilizar a seguinte regra em uma discagem com supervisão:

- Com a porta desligada, ao iniciar uma ligação (PickUp), chama-se o EnableCallProgress passando o valor CP_ENABLE_LINETONE_OR_BUSY
- Ao detectar o tom de linha, efetua a discagem e após a discagem, chama-se novamente o EnableCallProgress passando o valor CP_ENABLE_ALL**.**
- Após a discagem, é necessário habilitar a detecção de atendimento também, utilizando o método EnableAnswerDetection.
- Logo após o atendimento, é necessário desabilitar sua detecção chamando o método DisableAnswerDetection e chamar o EnableCallProgress com o valor CP_ENABLE_BUSY_OR_FAX, já que, após o atendimento só é necessário esperar o tom de ocupado e o fax.

O DisableCallProgress deve ser chamado para desabilitar qualquer tipo de supervisão e o método DestroyCallProgress deve ser chamado no término da aplicação para evitar a perda de recursos de memória (memory leak).

Conforme já foi dito, todos os parâmetros de configuração da thread de CallProgress ficam armazenados em um arquivo texto de configuração cp_default.cfg (padrão) que contém diversos comentários explicando cada parâmetro. Por isso ele é demonstrado a seguir:

```
[CallProgress]

;-------------------------------------------------------
; AnswerSensitivity refere-se a quantos "audios" deverao
ser
; recebidos na sequencia para considerar atendimento
; Valor minimo = 0
; Padrao = 1
; NUNCA ALTERE ESTE PARAMETRO POIS PODE DIFICULTAR A
; DETECCAO DE ATENDIMENTO
;-------------------------------------------------------
```


Copyright © 2010 Digivoice Eletrônica

```
AnswerSensitivity=1

;-------------------------------------------------------
; AnswerSensitivityTime e´ a duracao *minima* de um
; audio para se considerar atendimento.
; Para diminuir a sensibilidade
; aumente o valor. O padrao e´ 200ms
;-------------------------------------------------------

AnswerSensitivityTime=10

;-------------------------------------------------------
; GenericToneTimeout determina o tempo *minimo* que o
; sistema esperara´ para reconhecer o tom generico.
; Caso nao venha neste tempo, gerara o evento de
; timeout.
; Valor em milisegundos
; Opcoes de Call Progress:
; CP_ENABLE_GENERIC_TONE
;-------------------------------------------------------

GenericToneTimeout=15000

;-------------------------------------------------------
; GenericToneTime determina o tempo *minimo* para
; que o sistema reconheca o audio como tom generico
; Valor em milisegundos
; Opcoes de Call Progress:
; CP_ENABLE_GENERIC_TONE
;-------------------------------------------------------

GenericToneTime=500

;-------------------------------------------------------
; LineToneTimeout determina o tempo *minimo* que o
; sistema esperara´ para reconhecer o tom de linha.
; Caso nao venha neste tempo, o evento de timeout
; sera´ gerado.
; Valor em milisegundos
; Opcoes de Call Progress:
; CP_ENABLE_LINETONE_OR_BUSY
;-------------------------------------------------------

LineToneTimeout=15000

;-------------------------------------------------------
```


Copyright © 2010 Digivoice Eletrônica

```
; LineToneTime determina o tempo *minimo* para que
; o sistema reconheca o audio como tom de linha
; Valor em milisegundos
; *** PRECISA SER MAIOR QUE BusyMaxTime ***
; Opcoes de Call Progress:
; CP_ENABLE_LINETONE_OR_BUSY
;-----------------------------------------------------

LineToneTime=750

;-----------------------------------------------------
; FaxToneTimeout determina o tempo *minimo* que o
; sistema esperara´ para reconhecer o tom de fax.
; Caso nao venha neste tempo, gerara´ evento de timeout.
; Valor em milisegundos
; Opcoes de Call Progress:
; CP_ENABLE_BUSY_OR_FAX
;-----------------------------------------------------

FaxToneTimeout=15000

;-----------------------------------------------------
; FaxToneTime determina o tempo *minimo* para que
; o sistema reconheca o audio como tom de fax
; Valor em milisegundos
; Opcoes de Call Progress:
; CP_ENABLE_BUSY_OR_FAX e CP_ENABLE_ALL
;-----------------------------------------------------

FaxToneTime=200

;-----------------------------------------------------
; CallProgressTimeout e´ o tempo de espera apos
; a discagem para se considerar que houve um
; atendimento por timeout
; Valor em milisegundos
; Opcoes de Call Progress:
; CP_ENABLE_ALL
;-----------------------------------------------------

CallProgressTimeout=15000

;-----------------------------------------------------
; BusyMinTime determina o tempo *minimo* do
; tom de ocupado
; Valor em milisegundos
```


Copyright © 2010 Digivoice Eletrônica

```
; Opcoes de Call Progress:
; CP_ENABLE_LINETONE_OR_BUSY e CP_ENABLE_BUSY_OR_FAX e
; CP_ENABLE_ALL
;-------------------------------------------------------

BusyMinTime=200

;-------------------------------------------------------
; BusyMaxTime determina o tempo *minimo* do tom
; de ocupado
; Valor em milisegundos
; Opcoes de Call Progress:
; CP_ENABLE_LINETONE_OR_BUSY e CP_ENABLE_BUSY_OR_FAX e
; CP_ENABLE_ALL
;-------------------------------------------------------

BusyMaxTime=550

;-------------------------------------------------------
; BusySensibility determina quantos tons de
; ocupado devem ser detectados para que seja
; gerado um evento para a aplicacao.
; O padrao é 1 e este parametro e´ aplicado
; somente no CP_ENABLE_BUSY_OR_FAX
;-------------------------------------------------------

BusySensibility=5

;-------------------------------------------------------
; CallingMinToneTime determina o tempo *minimo* para se
; considerar um tom de chamada (apos a discagem). Sera´
; testado como intervalo junto com CallingMaxToneTime
; Valor em milisegundos
; *** PRECISA SER MAIOR QUE BusyMaxTime ***
; Opcoes de Call Progress:
; CP_ENABLE_ALL
;-------------------------------------------------------

CallingMinToneTime=450

;-------------------------------------------------------
; CallingMaxToneTime determina o tempo *maximo* para se
; considerar um tom de chamada (apos a discagem). Sera´
; testado como intervalo junto com CallingMinToneTime
; Valor em milisegundos
; No CP_ENABLE_ALL este tempo + 500ms tambem  e´
```


Copyright © 2010 Digivoice Eletrônica

```
; utilizado para medir o tom de discagem
; Opcoes de Call Progress:
; CP_ENABLE_ALL
;-------------------------------------------------------

CallingMaxToneTime=2000

;-------------------------------------------------------
; CallingMinSilTime o tempo *minimo* para se
; considerar silencio. Sera´ testado como intervalo
; junto com CallingMaxSilTime
; Valor em milisegundos
; Opcoes de Call Progress:
; CP_ENABLE_ALL
;-------------------------------------------------------

CallingMinSilTime=700

;-------------------------------------------------------
; CallingMaxSilTime determina o tempo *maximo* para se
; considerar silencio. Sera´ testado como intervalo
; junto com CallingMinSilTime
; Valor em milisegundos
; Opcoes de Call Progress:
; CP_ENABLE_ALL
;-------------------------------------------------------

CallingMaxSilTime=5000

;-------------------------------------------------------
; ToneInterruptionMinTime determina o tempo *minimo* do
; intervalo entre cada tom de linha. A deteccao do tom
; de chamando depende tambem dos tempos do intervalo
; entre eles.
; Valor em milisegundos
; Opcoes de Call Progress:
; CP_ENABLE_ALL
;-------------------------------------------------------

ToneInterruptionMinTime=20

;-------------------------------------------------------
; ToneInterruptionMaxTime determina o tempo *minimo* do
; intervalo entre cada tom de linha. O deteccao do tom
; de chamando depende tambem dos tempos do intervalo
; entre eles.
```


Copyright © 2010 Digivoice Eletrônica

```
; Valor em milisegundos
; Opcoes de Call Progress:
; CP_ENABLE_ALL
;-------------------------------------------------------

ToneInterruptionMaxTime=280

;-------------------------------------------------------
; LineToneMinTime determina o tempo *minimo* para se
; considerar o tom de linha
; Valor em milisegundos
; Opcoes de Call Progress:
; CP_ENABLE_ALL
;-------------------------------------------------------

LineToneMinTime=2500

;-------------------------------------------------------
; LineToneMaxTime determina o tempo *maximo* para se
; considerar o tom de linha
; Valor em milisegundos
; Opcoes de Call Progress:
; CP_ENABLE_ALL
;-------------------------------------------------------

LineToneMaxTime=2700

;-------------------------------------------------------
; Aqui sao definidas as frequencias utilizadas
; nas configuracoes a seguir associadas ao
; tipo de deteccao (ocupado, chamando, etc)
;-------------------------------------------------------
tone1=425,0
tone2=1100,0
tone3=2100,0
tone4=0,0
tone5=0,0
tone6=0,0
tone7=0,0
tone8=0,0

;-------------------------------------------------------
; Configuracao das frequencias para cada tipo de
; deteccao
; Aqui devera´ ser informado o indice de configuracao da
; voicerlib:
```


Copyright © 2010 Digivoice Eletrônica

```
; CP_SILENCE usar valor -> (0x20) 32
; CP_AUDIO   usar valor -> (0x21) 33
; CP_TONE1   usar valor -> (0x22) 34  (425Hz)
; CP_TONE2   usar valor -> (0x23) 35  (1100Hz)
; CP_TONE3   usar valor -> (0x24) 36  (2100Hz)
; custom comands
; CP_TONE4   usar valor -> (0x25) 37
; CP_TONE5   usar valor -> (0x26) 38
; CP_TONE6   usar valor -> (0x27) 39
; CP_TONE7   usar valor -> (0x28) 40
; CP_TONE8   usar valor -> (0x29) 41
;-------------------------------------------------------
Audio=0x21              ;default CP_AUDIO - nao mudar!
Silence=0x20            ;default CP_SILENCE - nao mudar!
LineToneFreq=0x22       ;default refere-se a CP_TONE1
CallingToneFreq=0x22    ;default refere-se a CP_TONE1
BusyToneFreq=0x22       ;default refere-se a CP_TONE1
Fax1ToneFreq=0x23       ;default refere-se a CP_TONE2
Fax2ToneFreq=0x24       ;default refere-se a CP_TONE3
GenericToneFreq=0x25    ;default refere-se a CP_TONE4
GenericToneFreq2=0x26   ;default refere-se a CP_TONE5
GenericToneFreq3=0x27   ;default refere-se a CP_TONE6
GenericToneFreq4=0x28   ;default refere-se a CP_TONE7
GenericToneFreq5=0x29   ;default refere-se a CP_TONE8

;-------------------------------------------------------
; Terminador de Arquivo
; Mantenha o proximo item sempre como o ultimo deste
; arquivo para que a voicerlib possa detectar se o
; arquivo esta´ danificado
;-------------------------------------------------------

end=1
```


Copyright © 2010 Digivoice Eletrônica

No caso da opção CP_ENABLE_ALL, como no Brasil todos os tons tem a mesma frequência (425Hz) a distinção de chamando, ocupado e tom de discar (linha) é feita pela análise de cadência, ou seja, duração de tons e silêncio, portanto tome cuidado com os tempos anteriores respeitando a seguinte relação:

```
(BusyMaxTime < CallingMinToneTime) e
(CallingMaxToneTime < LineToneMinTime)

  |-----|-----| Busy
        Min   Max
  |----------------|--------| Calling
                   Min      Max
  |--------------------------------|------|Line
                                    Min    Max
```

Acionamento das detecções nas placas E1

Conforme foi dito anteriormente, nas placas E1 não é necessário habilitar as detecções de tom de linha, ocupado ou atendimento pois estas respostas vêm automaticamente pelo protocolo. Já a detecção de fax deve ser habilitada explicitamente através da chamada do método SetDetectionType para as frequências 1100Hz e 2100Hz. Estas frequências já são previamente definidas através das constantes DETECT_TONE2 e DETECT_TONE3, as quais deverão ser passadas no segundo parâmetro do método SetDetectionType. Consulte o Guia de Referência para maiores detalhes.

Reconhecimento dos sinais de supervisão

Sinal de Ocupado: Ao ser detectado o sinal de ocupado, o evento OnBusyDetected é gerado, podendo ser tratado pelo usuário da maneira como quiser.

Copyright © 2010 Digivoice Eletrônica

Placa FXO/FXS/GSM: O evento é gerado sempre que for detectado o ocupado, portanto, caso o sinal de ocupado permaneça por muito tempo sem desligar, o programa gerará várias vezes o evento. Para evitar isso, o programador deve desabilitar a supervisão de linha com o método DisableCallProgress, após o primeiro ocupado. Outro efeito que pode ocorrer muito raramente é o sinal de ocupado ser detectado durante a conversação. Isto acontece com certos tons e candências de voz. Para prever esta situação, recomendamos ao programador sempre esperar pelo menos dois sinais de ocupado antes de tomar alguma atitude. Isto reduz muito os casos de interpretação errada pois é muito difícil uma voz gerar dois tons de ocupado na mesma cadência do tom real.

Exclusivo para placa E1: O evento de ocupado é gerado no caso de porta bloqueada, término de ligação, etc. Como não se trata de um tom intermitente e sim de dado R2D, o evento OnBusyDetected só ocorrerá uma vez. Por característica do protocolo R2D, é possível detectar o desligamento independente de quem tenha gerado a ligação.

Detecção de Fax: É útil principalmente em aplicações de atendimento automático. É possível fazer uma rotina, por exemplo, que ao perceber um sinal de fax, transfere para o ramal do aparelho de fax. O evento que é gerado neste caso é o OnFaxDetected.

Detecção de Atendimento: A VoicerLib permite ao programador detectar quando uma ligação foi atendida do outro lado da linha. O evento OnAnswerDetected é gerado sempre quando este sinal for detectado. Lembre-se que, nas placas FXO/FXS/GSM, é necessário desabilitar a detecção de atendimento logo após a ocorrência da primeira detecção para evitar detectar falsos atendimentos durante a conversação.

Detecção de sinal de discagem: Em linhas analógicas, o sinal de discagem é aquele tom contínuo que se escuta ao tirar o fone do gancho. Em alguns casos, este sinal pode demorar alguns segundos, principalmente em centrais congestionadas. Este evento é controlado pelos métodos EnableCallProgress e

Copyright © 2010 Digivoice Eletrônica

DisableCallProgress . O evento OnDialToneDetected é gerado sempre quando este sinal for detectado.

Já em linhas digitais (E1), não existe o tom de discagem propriamente dito. No momento de ocupar a porta, já é enviado imediatamente se a porta estiver bloqueada, por exemplo. Quando a porta da placa recebe a confirmação de ocupação, significa que está pronta para discar. A VoicerLib gera o evento OnDialToneDetected quando esta confirmação é recebida.

Em uma situação prática, é mais correto esperar o tom de discagem para então utilizar o método Dial para discar e não atender e discar em seguida.

Detecção de sinal de chamada: Ao discar para um determinado número, sempre ouvimos o sinal de chamada, que indica que o telefone está tocando do "outro lado" da linha. Este evento também é controlado pelos métodos EnableCallProgress e DisableCallProgress. Com este sinal podemos contar quantos toques são dados até a ligação ser atendida ou ainda detectar que a ligação não foi completada (no caso de o sinal não ser detectado).

O evento OnCalling ocorre sempre quando o sinal for detectado, o que faz com que ele seja chamado várias vezes até a ligação ser atendida.

Nas placas E1 com a thread de controle habilitada, o evento OnCalling é gerado independente de chamar Enable/DisableCallProgress e independe da chegada do tom audível, pois funciona através de um timer interno da VoicerLib. Para a aplicação final, essa diferença de funcionamento não altera nenhum procedimento. O tom de controle de chamada (ringback) ou ocupado é gerado localmente pela placa para que o usuário tenha conhecimento do andamento da ligação.

Copyright © 2010 Digivoice Eletrônica

Detecção de Silêncio

A VoicerLib 4 permite fazer detecção de silêncio através de um comando específico para isso. Em muitas situações, perceber a existência de silêncio é muito importante. Um exemplo típico de aplicação é a gravação de mensagens de correio de voz onde o desenvolvedor queira interromper a gravação caso a ligação fique em silêncio por N segundos. Outro exemplo é na gravação de conversas telefônicas com a linha em paralelo pois permite que a gravação seja pausada ou terminada caso haja silêncio por um tempo pré-determinado.

O método que habilita esta funcionalidade é o EnableSilenceDetection que tem como parâmetro a porta, o tempo mínimo para considerar silêncio (silence_time) e o tempo mínimo para considerar áudio (audio_time). Estes dois últimos parâmetros devem ser informados em milisegundos (1000 = 1s).

O parâmetro audio_time permite ainda receber o valor zero, indicando que, ao primeiro sinal de áudio a aplicação será avisada. Dependendo da aplicação e do ambiente de produção este valor pode ser aumentado para um valor maior pois assim a VoicerLib desprezará sinais de áudio que tiverem duração menor que a especificada. Esse comportamento equivale a uma diminuição da sensibilidade da detecção de áudio. Se não houver situações de ruído na linha, etc, recomenda-se utilizar o valor zero nesse parâmetro(audio_time).

O evento OnSilenceDetected será gerado sempre que houver uma mudança no estado, ou seja, quando for detectado o silêncio, o evento será gerado uma vez, informando no parâmetro SignalCode o valor DG_SILENCE_DETECTED (1). Quando receber um sinal de áudio, o evento será gerado novamente com o parâmetro SignalCode recebendo o valor DG_AUDIO_DETECTED (0). Depois disso, o evento só será gerado de novo se for detectado um silêncio e assim por diante.

Para desabilitar esta detecção, é necessário chamar o método DisableSilenceDetection.

Copyright © 2010 Digivoice Eletrônica

Detecção de Tons

Uma das maiores evoluções da VoicerLib 4 em relação à versão 2.x é nas detecções de tons. Agora existem muito mais opções de detecções, dando ao desenvolvedor grande liberdade de trabalho mesmo em situações incomuns.

Por padrão, a VoicerLib permite a detecção de até oito tipos de tons, utilizados em supervisão de linha, etc. Destes 8 tipos de tons, temos pré-configurados três:

- DETECT_TONE1 - Tom puro 425Hz, o padrão brasileiro para tons de supervisao de linha
- DETECT_TONE2 - Tom de 1100Hz, utilizado para detectar FAX
- DETECT_TONE3 - Tom de 2100Hz, utilizado para detectar FAX

Estes tons puros são suficientes para a supervisão de linha padrão e detecção de fax na maioria dos casos. O tópico "Supervisão de Linha" aborda detalhadamente como utiliza-lo e a chamada a estas funções está embutido nas chamadas de callprogress, portanto, uma configuração manual não é necessária a não ser em casos especiais.

O método que habilita/desabilita as detecções é o SetDetectionType que tem como parâmetro a porta, o comando (DETECT_TONE1, etc...) e o flag que habilita ou desabilita a detecção (DG_ENABLE/DG_DISABLE). Consulte o guia de referência para ver todas as opções.

A chamada deste método tem efeito cumulativo, ou seja, fazendo uma chamada para habilitar TONE3 por exemplo, e outra chamada para habilitar TONE1 ocasionará a detecção dos dois tipos.

Este mesmo método habilita a detecção de dígitos, que será abordado no tópico seguinte.

Copyright © 2010 Digivoice Eletrônica

Mudando a configuração padrão

O desenvolvedor poderá reconfigurar todos os tipos de tons (TONE1 a TONE8) a serem detectados, através do método SetCardDetections, passando como parâmetro a placa, o tom a ser modificado e o par de frequências que passará a ser monitorado. No Guia de Referência das funções e métodos são apresentadas todas as opções.

ATENÇÃO: É importante ressaltar que a alteração das frequências a serem detectadas só podem ser feitas na placa inteira e não por porta individualmente. Isso não constitui uma limitação já que normalmente uma alteração deste tipo afeta todo o sistema e não portas específicas.

Copyright © 2010 Digivoice Eletrônica

Detecção de Dígitos

A VoicerLib permite detectar dígitos tanto em tons multifrequenciais como em pulso nativamente.

A detecção de tons multifrequenciais é habilitada ou desabilitada através do método SetDetectionType que tem como parâmetro a porta, o comando (DETECT_DTMF etc...) e o flag que habilita ou desabilita a detecção de dígitos (DG_ENABLE/DG_DISABLE). Consulte o guia de referência para ver todas as opções.

O segundo parâmetro, o comando pode assumir as seguintes opções:

- DETECT_DTMF - Detecção do DTMF, o mais comum para aplicações com interação com o usuário
- DETECT_MFF e DETECT_MFT - São os sinais multifrequenciais comuns à sinalização R2D
- DETECT_MF - Sinal multifrequencial customizável

A detecção de pulso é habilitada pelo método EnablePulseDetection e desabilitada pelo método DisablePulseDetection. No método EnablePulseDetection é passado a sensibilidade de detecção, sendo o padrão zero. Esta sensibilidade pode ser alterada caso se perceba uma dificuldade ou facilidade maior nas detecções de pulso.

O funcionamento das duas detecções é praticamente o mesmo, exceto pelo fato que a detecção de tom pode ser feita durante a reprodução de uma mensagem e a de pulso somente no silêncio após a mensagem.

Habilitando o cancelamento de eco, é possível ter boa performance na detecção de pulsos sobre a mensagem.

Copyright © 2010 Digivoice Eletrônica

A VoicerLib permite tratar o reconhecimento de dígitos de duas maneiras diferentes.

A primeira é através do evento OnDigitDetected, que é gerado sempre que um dígito qualquer for detectado pela placa. O dígito detectado será passado através do parâmetro Digit do evento. Este parâmetro conterá o código ASCII do dígito.

Caso o programador implemente algum tratamento neste evento, deve tomar cuidado para saber o momento em que ele ocorreu, pois pode acontecer uma detecção pelos dígitos gerados pela própria placa. O mais aconselhável é só habilitar as detecções nos momentos que ela seja necessária, para evitar situações de talk-off. É possível otimizar a detecção utilizando o método SetTwist (consulte o Guia de Referência para maiores detalhes).

TALK-OFF: A voz humana, em uma conversa normal, pode conter a mesma frequência dos dígitos detectados pela placa, portanto, quando o operador ou o interlocutor falar, algum dígito pode ser detectado e se o tratamento no OnDigitDetected não for adequado, o sistema pode interpretar errôneamente os sinais recebidos.

Uma aplicação típica que é recomendada a utilização do evento OnDigitDetected é na identificação de chamadas (BINA).

Exemplo:

Neste exemplo, o dígito é detectado o tempo inteiro e mostrado na tela.

```
Private Sub VoicerLibX1_OnDigitDetected(Port As Integer,
Digit As Integer)
    lblStatus.Caption = "Detectou Digito " +
Chr$(voicerlibx1.ReadDigits(Port))
End Sub
```

A segunda maneira de detectar dígitos é através do método GetDigits.

Copyright © 2010 Digivoice Eletrônica

O método GetDigits inicia a monitoração de dígitos, sendo que é possível determinar o número máximo de dígitos, se esperará um dígito terminador, e tempos máximos para recepção destes dígitos.

Após o início da monitoração, o evento OnDigitsReceived pode ocorrer a qualquer momento. Este evento ocorre quando uma das condições impostas pelo método GetDigits for satisfeita. O método ReadDigits permite ler o conteúdo do buffer de dígitos da porta.

Durante a monitoração é possível cancelar o andamento do GetDigits através do método CancelGetDigits. Se esse método for chamado, o GetDigits será cancelado mas o evento OnDigitsReceived não será gerado e o buffer de dígitos será apagado. O método CancelGetDigits deverá ser chamado principalmente em situações onde a ligação foi terminada durante o GetDigits.

Uma mudança importante em relação às versões anteriores da VoicerLib é que agora o buffer de dígitos deve sempre ser apagado explicitamente pelo programa, através do método ClearDigits.

Exemplo:

No exemplo abaixo o GetDigits inicia esperando até 5 dígitos, ou até receber o terminador # por no máximo 10 segundos de espera total ou 5 segundos de intervalo entre cada dígito:

```
Private Sub Espera Digito()
  VoicerLibX1.GetDigits Porta,5,"#",10000,5000
End Sub


'No evento OnDigitsReceived é que será tratado os
digitos recebidos, ou verificado timeout

Private Sub VoicerLibX1_OnDigitsReceived(Port As
Integer,Status As VoicerLib.TxWaitDigit)
```


Copyright © 2010 Digivoice Eletrônica

```
  Select Case Status

    Case edMaxDigits:
    'Alcançou o máximo de digitos, disca para o ramal

    Case edTermDigit:
    'Recebeu um número com # no fim

    Case edDigitTimeOut:
    'Timeout global de 10 segundos

    Case edInterDigitTimeOut:
    'Ocorreu timeout entre dois dígitos

  End Select

End Sub
```

O evento OnDigitsReceived também pode ter a variável Status com valor edDigitOverMessage. Neste caso, o evento terá sido gerado pela detecção de um dígito durante a reprodução de uma mensagem a partir dos métodos Playxxx.

Copyright © 2010 Digivoice Eletrônica

Identificação de Chamadas

A identificação de chamadas (BINA) está facilmente disponível através dos métodos IdleXXX nas placas FXO e FXS e da thread de controle nas placas E1 (CreateE1Thread, CreateISDNThread). Consulte os referidos tópicos para maiores detalhes.

Entretanto, o desenvolvedor também pode desenvolver sua própria rotina de detecção de identificador de chamadas, normalmente fazendo uso do método SetDetectionType para configurar o tipo de sinalização e manipulando estes dígitos através do evento OnDigitDetected. Neste caso, toda a montagem da string com o número identificado é manual.

Copyright © 2010 Digivoice Eletrônica

Portas Virtuais

Quando a VoicerLib é inicializada, todas as placas presentes no computador são analisadas e o número da porta é assumido na ordem em que as placas estão posicionadas no barramento PCI. Por exemplo, se houverem 2 placas FXO VB0408PCI de 8 canais cada e uma placa E1 VB6060PCI de 60 canais, as portas de 1 a 16 serão atribuídas às duas placas FXO e as portas 17 a 77 serão atribuídas à placa E1.

Aparentemente não existe padrão entre os fabricantes de motherboard quanto à ordem das placas no barramento. Também os sistemas operacionais podem reconhecer as placas de maneiras diferentes no mesmo hardware. Entretanto, não existem alteração nessa ordem depois de tudo instalado corretamente.

Essa alocação automática de portas é adequada à maioria das aplicações, porém existem situações onde é interessante o desenvolvedor determinar o número da porta de cada canal físico da placa. A gravação em paralelo em placas E1 de 60 canais é um exemplo típico pois somente os primeiros 30 canais de cada placa são utilizados o que faz com que a aplicação tenha que gerenciar os canais de 1-30, 61-90, 121-150 e assim por diante. Com a utilização dos métodos de canais virtuais, o desenvolvedor pode, no início da aplicação associar os canais de maneira a ficar com uma numeração de portas continua (de 1 a 90, por exemplo).

Associando um canal físico à uma porta virtual

O método DefinePortResource permite atribuir uma numeração qualquer a um canal físico de determinada placa, tendo a seguinte sintaxe:

```
DefinePortResource(PortaVirtual, Placa, Canal)
```


Copyright © 2010 Digivoice Eletrônica

Onde PortaVirtual é o número lógico a ser utilizado, Placa é o número da placa (de 1 a *n*, na ordem do barramento PCI) e Canal é o canal físico de Placa (ex.: uma placa FXO de 8 canais tem canais de 1 a 8).

A PortaVirtual deverá respeitar o número máximo de portas reconhecida pelo sistema, ou seja, não poderá ter valor maior do número máximo de portas, valor esse, que pode ser obtido através do método GetPortsCount.

A única verificação que o método DefinePortResource faz, é com relação aos valores de placa e canal. Não existe nenhum feedback se determinada porta virtual já foi alocada ou não. Este controle deverá ser feito pela aplicação.

Após fazer a associação, a portavirtual passará a ser utilizada pela VoicerLib em todos os métodos e eventos para referenciar a porta física da placa selecionada.

Restaurando a configuração padrão de inicialização

A qualquer momento é possível voltar a numeração das portas para o reconhecimento automático, bastando para isso chamar o método ResetPortResource. Este método não exige nenhum parâmetro e fará com que a VoicerLib efetue o reconhecimento de portas feito na inicialização.

Verificando a configuração de portas virtuais

Pode ser útil ao desenvolvedor saber como está a configuração das portas virtuais. Para isso pode-se fazer uso dos métodos GetPortsCount, GetCardNumber e GetRelativeChannelNumber**.** O veja o seguinte código em VisualBasic que exibe a configuração atual das portas do sistema:

```
'Funcao que retorna string com identificacao da placa

Private Function PegaNomePlaca(nCard As Integer) As
String

  Select Case voicerlib1.GetCardType(nCard)
```


Copyright © 2010 Digivoice Eletrônica

```
    Case VB0408PCI
      PegaNomePlaca = "VB0408PCI"
    Case VBE13030PCIe
      PegaNomePlaca = "VBE13030PCI Express"
    Case VBE16060PCI
      PegaNomePlaca = "VBE16060PCI"
  End Select

End Function


Private Sub cmdAnalisa_Click()
Dim i As Integer
Dim sMsg As String
Dim card As Integer, card_ch As Integer

  MostraStatus 0, "Numero total de portas = " &
voicerlib1.GetPortsCount()
  For i = 1 To voicerlib1.GetPortsCount()
    card = voicerlib1.GetCardNumber(i)
    sMsg = PegaNomePlaca(card)
    MostraStatus i, "Placa " & card & " tipo: " & sMsg &
" - ch da placa: " &
voicerlib1.GetRelativeChannelNumber(i)
  Next i

End Sub
```


Copyright © 2010 Digivoice Eletrônica

Gravando uma Conversa

A VoicerLib permite gravar conversas telefônicas nas placas com interface analógicas (FXS ou FXO) e digitais (E1). Nas placas com interface analógica, devido a característica dos circuitos telefônicos, a gravação registra tanto o áudio que é gerado pela placa (Tx) quanto o áudio que chega à placa (Rx), nas placas com interface digital (E1) o Rx e o Tx são separados e para gravá-los é preciso fazer uma gravação em paralelo, caso seja um E1 de um PABX, ou um recurso adicional (TxRxMixEnable) se for o caso de uma URA. A gravação em paralelo tem um tópico à parte, porém os conceitos básicos são apresentados aqui.

É possível efetuar a gravação em diversos formatos de arquivos, com nível de compactação diferentes entre si. Observe a tabela:

Tabela de formatos de gravação

Formato	Taxa (Kbits/s)	MB/h (*)	Kbytes/h (*)
ffWavePCM	128	57.6	57600
ffWaveULaw	64	28.8	28800
ffWaveALaw	64	28.8	28800
ffGsm610	13.2	5.9	5940
ffWave49	13.2	5.9	5940

(*) ***MB/h e KBytes/h*** indica a taxa de ocupação do arquivo gerado pela gravação no formato especificado.

O método SetRecordFormat indica para a VoicerLib qual o formato que determinada porta deverá assumir.

O método que determina o formato de gravação é o SetRecordFormat e não a extensão do arquivo, portanto, simplesmente associar a extensão .gsm ao arquivo não determinará o seu formato!

Para aplicações com extensivo uso de gravação recomenda-se os

Copyright © 2010 Digivoice Eletrônica

formatos GSM ou Wave49 que oferecem uma excelente relação entre tamanho e qualidade de áudio. Diferente da VoicerLib2, a codificação/decodificação do GSM e Wave49 são feitas diretamente pelo processador (DSP) da placa sendo portanto os formatos de áudio mais eficientes pois o computador não gasta tempo de CPU nos processos de codificação/decodificação e o I/O no barramento PCI também é muito menor que nos formatos Wave (PCM,ALaw e ULaw) .

Existe também o método SetGSMMode que indica a VoicerLib se deverá ser utilizado o GSM compatível com o Asterisk(c) (GSM_RAW (1)) ou o GSM padrão da Digivoice (GSM_DIGIVOICE(0)). A diferença entre os dois formatos consiste apenas na existência de um cabeçalho no padrão Digivoice. A codificação em si é a mesma e por isso não foi criado um novo formato de gravação. Este método afeta todas as portas de todas as placas e o padrão assumido atualmente é o GSM_RAW que não contém cabeçalho no arquivo.

Para efetuar a gravação de portas na voicerlib, é necessário habilitar o envio de amostras da placa através do método EnableInputBuffer. Este método espera dois parâmetros: a porta que gerará as amostras para a aplicação e flag que habilita ou não o AGC (Controle Automático de Ganho). Após chamar o EnableInputBuffer, o método RecordFile efetivamente iniciará a gravação.

É muito importante verificar os valores de retorno dos métodos EnableInputBuffer e RecordFile. Dependendo da carga de processamento, é possível que, ao chamar o método RecordFile a thread de controle iniciada pelo EnableInputBuffer ainda não esteja pronta. Neste caso a melhor alternativa é que, em caso de erro, o desenvolvedor chame o método RecordFile novamente.

O método RecordFile também permite determinar se algum dígito será utilizado como finalizador de gravação. Esta característica permite uma implementação do tipo: "Grave seu recado e ao final

Copyright © 2010 Digivoice Eletrônica

digite # se quiser falar com a telefonista".

O dígito detectado (se usado) é armazenado e deve ser recuperado, para isso utilize o método ReadDigits , que pode ser tratado posteriormente.

Dependendo do fluxo de operação do sistema desenvolvido, é necessário que seja acumulado os dígitos detectados, este é o comportamento padrão da VoicerLib.

Sempre quando for iniciada a gravação o evento OnRecordStart é gerado, permitindo tratamento nesta situação. É possível saber se determinada porta está gravando chamando o método IsRecording .

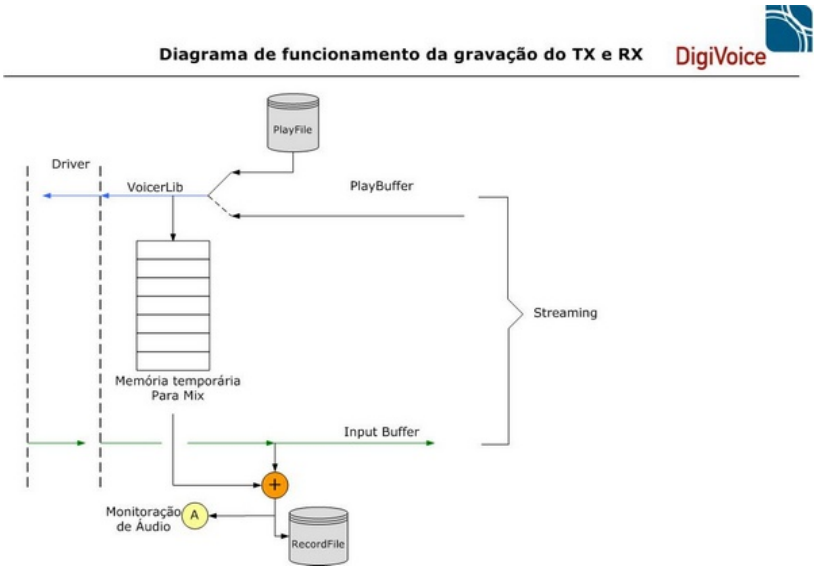
Ao término da gravação o evento OnRecordStop é gerado automaticamente, inclusive informando qual o status da interrupção da gravação.

Este status pode ser uma ação direta do operador (com a chamada do método StopRecordFile), um dígito recebido ou ainda um erro qualquer (disco cheio, por exemplo).

Para desativar a thread de controle de gravação (iniciada pelo EnableInputBuffer) deve ser chamado o método DisableInputBuffer, passando como parâmetro a porta. O melhor lugar para desabilitar a thread de controle de gravação é no evento OnRecordStop pois ali sabemos que a gravação foi efetivamente finalizada e a thread pode ser destruída.

A thread de controle de gravação é o processo que consome processamento pois com ela ocorre um intenso I/O entre o device driver e o hardware. Por isso é recomendado que ela seja sempre habilitada e desabilitada quando do início e fim das gravações.

Copyright © 2010 Digivoice Eletrônica



É recomendado chamar o EnableInputBuffer e DisableInputBuffer no começo e no fim da aplicação, respectivamente, pois o processo de criação de threads pode ser um pouco lento em relação à algumas situações onde o começo e fim de gravação é muito rápido (grande volume de ligações) e fazer com que a repetida chamada do EnableInputBuffer possa retornar erro devido ao fato do DisableInputBuffer anterior ainda não ter fechado o arquivo.

Sempre quando o EnableInputBuffer for chamado uma única vez no início da aplicação, o método PauseInputBuffer deverá ser chamado para evitar o consumo desnecessário de processamento em momentos onde não for necessária sua utilização. Este método é chamado com o parâmetro porta e com um segundo parâmetro que indica para colocar a thread em pausa (DG_PAUSE(1)) ou sair da paus (DG_RELEASE(0)).

Exemplo (usando o ActiveX):

Copyright © 2010 Digivoice Eletrônica

Após falar: *"Grave seu recado e ao final digite # se quiser falar com a telefonista"*, o sistema iniciará a gravação. Se for detectado o #, discará para o ramal 200.

Exemplo em VB6:

```
Private Sub DeixarRecado()

  VoicerLib.EnableInputBuffer 1,DG_ENABLE
  'Inicia gravação
  VoicerLib.RecordFile(1,"c:\recado.gsm,"#")
  'Lembre que o programa segue o fluxo normal após
iniciada a gravação

End Sub



'Rotina de tratamento do click de um botão para o caso
de o operador parar a gravação manualmente

Private Sub cmdParaGravar_Click()
  VoicerLib.StopRecordFile(1)
End Sub



'Evento que ocorre quando a gravação é finalizada é
necessário analisar o motivo

Private Sub VoicerLibX1_OnRecordStop(ByVal Port as
Integer, Status As VoicerLib.TxStopStatus)

  Select Case Status
    Case ssStopped:
      'Gravação interrompida pelo operador
      VoicerLib.HangUp 1
    Case ssDigitReceived:
      'Recebeu o digito finalizador
      If Digits = "#" then
        'Transfere para a telefonista
        VoicerLib.Flash 1,600,1000
        VoicerLib.Dial 1,"200",1000
        VoicerLib.HangUp 1
      End If
```


Copyright © 2010 Digivoice Eletrônica

```
  End Select
  VoicerLib.DisableInputBuffer Port
End Sub
```

Durante a gravação é possível monitorar o seu andamento através do evento OnRecording. No ActiveX, o parâmetro ElapsedTime indica a quantidade de segundos decorridos até aquele momento. O mesmo ocorre na variável context_data da API.

O método RecordPause, permite interromper temporariamente uma gravação e em seguida continuar no mesmo arquivo. Neste método é necessário passar a porta e um flag booleano indicando se coloca em pausa (TRUE) ou retira da pausa (FALSE).

Copyright © 2010 Digivoice Eletrônica

Reproduzindo Mensagens

A VoicerLib permite reproduzir qualquer mensagem gravada em formato Wave ou GSM através do método PlayFile .

O PlayFile detecta automaticamente o tipo de arquivo baseado na extensão. Se encontrar um arquivo com a extensão *.gsm* tentará reproduzir o arquivo usando o codec GSM. Se for encontrada a extensão .wav, a Voicerlib efetuará uma análise do cabeçalho para identificar o tipo de wave. Se for utilizado uma extensão desconhecida, assume o que está especificado no SetPlayFormat.

Além disso, é possível determinar se algum dígito será utilizado como finalizador da reprodução. Esta característica é interessante para, por exemplo, implantar um menu de opções em um sistema de auto-atendimento com a possibilidade de digitar a opção sobre a mensagem.

O dígito detectado (se usado) é armazenado internamente para cada porta da placa e pode ser recuperado através do método ReadDigits podendo ser tratado posteriormente.

A VoicerLib assume que o programador sempre deverá chamar explicitamente o método ClearDigits para apagar o buffer de dígitos sempre quando for necessário.

Sempre quando for iniciada a reprodução, o evento OnPlayStart é gerado. Uma finalidade deste evento é permitir atualizações de interface, como por exemplo, desabilitar botões e exibir mensagens ao usuário.

Ao término da reprodução o evento OnPlayStop é gerado automaticamente, inclusive informando qual o motivo da interrupção da reprodução. Este motivo pode ser uma ação direta do operador (com a chamada do método StopPlayFile), um dígito recebido, o simples término da mensagem ou ainda um erro qualquer (disco cheio, por exemplo).

Se um dígito for detectado durante a reprodução (desde que o

Copyright © 2010 Digivoice Eletrônica

parâmetro TermDigits tenha sido configurado), além do evento OnPlayStop, é gerado também o evento OnDigitsReceived passando na variável Status o valor edDigitOverMessage. Isto facilita a criação de menus de atendimento, já que toda a consistência do que foi digitado pode ser feita apenas no evento OnDigitsReceived.

O último parâmetro do PlayFile (Origin) permite começar a reproduzir a mensagem a partir de um determinado ponto. Por exemplo, para reproduzir uma mensagem a partir de 10 segundos do início dela, basta colocar o número 10 neste parâmetro. Isto é útil para funções de reprodução de mensagens. Se for passado (0) ZERO como parâmetro, a mensagem será reproduzida do início. Se for passado -1 como parâmetro a mensagem começará a ser reproduzida do final menos 2 segundos.

Exemplo:

```
VoicerLibX1.PlayFile(1,"c:\boas_vindas.wav,"345",0)
```

Neste exemplo é iniciada a reprodução de um arquivo chamado boas_vindas.wav e os dígitos 3,4 e 5 poderão interromper a reprodução da mensagem, caso sejam detectados.

Devido à características de implementação, a reprodução de mensagens não necessita que o buffer de amostras seja ativado antes da reprodução, como é feito na gravação. Para maiores detalhes de como enviar amostras para a placa diretamente, consulte o tópico Streaming de Áudio.

Copyright © 2010 Digivoice Eletrônica

Conferência entre portas

A VoicerLib na placa E1, oferece recursos que permitem que determinadas portas possam conversar com outras. Com isso é possível criar salas de conferência onde várias pessoas possam conversar entre si. Este mesmo recurso foi utilizado para a criação da thread de Logger, utilizada na gravação em paralelo para placas E1 explicada anteriormente.

É suportado até 30 salas de conferência com número variável de portas por sala. Para criar uma sala de conferência utiliza-se o método CreateChatRoom que possui como parâmetro a placa que gerenciará o recurso e o número máximo de portas que esta sala poderá receber. O retorno deste método é o identificador/handle da sala e será necessário nos demais métodos, portanto, sempre armazene o retorno do método de maneira que possa ser utilizado posteriormente.

Para excluir uma sala, o método utilizado é o DestroyChatRoom, que sempre deverá ser chamado antes de finalizar uma aplicação, pois além de alocar recursos da placa, também é utilizado estruturas de memória que só são liberadas quando uma sala é explicitamente destruída.

A inclusão de portas em uma sala de conferência deve sempre ser feita pelo método ChatAddPort passando a porta e o identificador da sala. Ao incluir uma porta na sala, esta não estará ainda disponível para ouvir e falar com os outros participantes, precisando ainda habilitar a porta na sala. Essa separação entre os procedimentos de adicionar e habilitar foi utilizada para que seja possível uma porta interagir com menus, por exemplo, sem deixar a sala.

Para habilitar uma porta na sala utilize o método ChatEnablePort que ainda permite dizer se a porta só falará, só ouvirá ou ambos em determinada sala, através da parâmetro Direction. Isso dá extrema versatilidade em termos de aplicações possíveis utilizando a VoicerLib.

Copyright © 2010 Digivoice Eletrônica

O método ChatDisablePort permite desabilitar uma porta da sala, sem removê-la definitivamente, como já foi explicado anteriormente. Em uma aplicação, isso permite, por exemplo que o participante possa, durante a conversa, teclar um dígito e entrar em um menu de opções qualquer. Depois de interagir com este menu, poderá voltar novamente à sala chamando o método ChatEnablePort novamente.

Para remover uma porta de uma sala definitivamente, utilize o método ChatRemovePort . Esta situação ocorreria quando o usuário fosse embora da sala de conferência, desligando ou movendo-se para outra sala.

Por fim, um recurso adicional é a possibilidade de gravação do que se fala na sala de conferência. Isto é possível através do método SetPortChatLog, que habilita ou desabilita determinada porta à receber o áudio de todas as outras portas participantes de uma conferência. É importante entender que a porta escolhida para ouvir e gravar a conferência não poderá estar participando desta ou de qualquer outra conferência enquanto a opção estiver habilitada.

Copyright © 2010 Digivoice Eletrônica

Streaming de Áudio

Uma importante evolução da Voicerlib2 para Voicerlib4 é o tratamento de streaming de áudio. A utilização deste recurso é necessário quando a aplicação precisa receber da placa ou enviar para a placa, amostras de áudio sem manipulação de arquivos, somente utilizando a memória. O exemplo típico é uma aplicação que "conversa" com outra pela rede (VoIP). O suporte ao Asterisk, disponível nesta versão da VoicerLib foi feito utilizando dos recursos de streaming.

Um conceito importante é que o streaming de áudio pode ser feito em qualquer formato suportado pela VoicerLib, inclusive o GSM (excelente para aplicações VoIP). Todos estes formatos são tratados pelo processador de cada placa, não utilizando recursos da CPU do micro para codificação/decodificação de formatos de áudio.

Enviando Áudio para uma porta da Placa

Para indicar qual o formato utilizado para envio de amostras para uma porta da placa Digivoice deve ser utilizado o método SetPlayFormat, o mesmo utilizado para a reprodução de mensagens gravadas. O método para enviar amostras para a placa é o PlayBuffer que tem como parâmetro a porta, um ponteiro de memória com o buffer de amostras a ser enviado e a quantidade de amostras que deverá ser enviada.

Quando uma aplicação enviar amostras para a placa, é importante respeitar a cadência de tempo que estas amostras devem ser enviadas para que se tenha uma reprodução contínua de áudio, sendo que, no formato GSM, são enviadas 33 amostras a cada 20ms e no caso do wave (uLaw ou aLAw) são 16 amostras a cada 2ms (muito rápido). Esta necessidade de velocidade que também reforça a utilização do GSM para este tipo de aplicação.

O método PlayBuffer permite o envio, em uma única chamada, de múltiplos destes valores(33 ou 16 amostras). Por exemplo, no formato wave é possível passar 32,64,etc amostras pois

Copyright © 2010 Digivoice Eletrônica

internamente essas amostras são bufferizadas para serem enviadas para a placa no momento certo. Porém é preciso tomar certos cuidados para não enviar uma quantidade de amostras maior do que a VoicerLib possa tratar. Nesta situação, o áudio perderá algumas amostras, aparecendo
picotado.

O último parâmetro do PlayBuffer permite que seja passado um ponteiro para um inteiro que receberá, no retorno do método, a quantidade de bytes livres no buffer.

A aplicação deverá monitorar se existem amostras a serem enviadas para a placa. Caso não haja, deverá chamar o método StopPlayBuffer para que a placa não fique reproduzindo o último buffer de amostras, o que acarretaria em um zumbido. Caso a aplicação queira enviar um ruído de conforto (confort noise), deverá gerar este ruído por conta própria e enviá-lo para a placa.

Recebendo Áudio de uma porta da Placa

Este procedimento é exatamente o inverso do anterior. Aqui as amostras que a porta da placa recebe são enviadas para a aplicação para daí serem tratadas quando for necessário. Como neste caso, é a VoicerLib que sabe quando as amostras estão disponíveis, o mecanismo é um pouco mais complexo do que o uso do PlayBuffer/StopPlayBuffer.

Conforme já foi explicado no tópico "Gravando uma Conversa", a VoicerLib 4 exige que o envio de amostras da placa para a aplicação seja habilitada através do método EnableInputBuffer e desabilitada pelo DisableInputBuffer. Uma vez que as amostras estão disponíveis, a VoicerLib pode decidir se vai gravar estas amostras em disco (RecordFile) ou as enviará para a aplicação final. Neste segundo caso, é necessário a utilização de uma função do tipo Callback.

Uma função do tipo CallBack é uma função convencional na linguagem de programação utilizada pelo desenvolvedor. A única diferença é que quem chamará esta função é a VoicerLib e não a aplicação final. Quando houver amostras disponíveis, a VoicerLib

Copyright © 2010 Digivoice Eletrônica

chamará diretamente a função da aplicação disponibilizando assim as amostras de áudio. Esse procedimento é muito similar aos eventos, porém com um compromisso de tempo real que os eventos não podem garantir. A utilização de linguagem C é recomendada para aplicações deste tipo, devido à facilidade que este tipo de recurso tem nesta linguagem. Observe o exemplo:

```
void CALLBACK InputStreaming(short port, int count, void
*data)
{
  //pega as amostras em *data
}
```

Para informar a VoicerLib qual a função Callback, utilize o método SetAudioInputCallback que passa como parâmetro o ponteiro da função callback. Depois de feita essa associação e de habilitado o envio de amostras (EnableInputBuffer) a função callback será acionada toda vez que houverem amostras disponíveis.

```
  SetAudioInputCallback(InputStreaming);
```

É importante ressaltar que é de responsabilidade da aplicação ter capacidade de receber e tratar todas as amostras recebidas, criando buffers se forem necessários.

Copyright © 2010 Digivoice Eletrônica

Utilizando a placa VB0404GSM

A placa VB0404GSM é uma placa de 2 ou 4 portas, voltada para aplicações de comunicação com telefonia móvel. Cada porta possui um módulo GSM responsável pela comunicação com a rede de telefonia móvel. A possibilidade de redução de custos por ligações de aparelhos de telefonia móvel para aparelhos de telefonia móvel, com tarifas muito menores que as tarifas de ligações entre aparelhos de telefonia móvel e telefonia fixa, impulsionou o mercado de sistemas GSM.

A placa VB0404GSM tem a maioria de suas funcionalidades compatíveis com as características da VoicerLib nas outras placas, como a VB0408PCI ou VB0408PCIe. Algumas novas funcionalidades foram incluídas principalmente no envio e recebimento de mensagens SMS.

Thread de GSM

Pelas características dos módulos GSM é necessária a utilização de uma thread para cada porta da placa VB0404GSM.

A thread de GSM deve ser iniciada após o método StartVoicerLib através do método CreateGSMThread.

Como a inicialização dos módulos é um processo lento, pois envolve conexão com a operadora de telefonia móvel e várias outras configurações, o método CreateGSMThread retorna antes da conclusão da inicialização e foi criado um evento OnGSMReady sinalizando se toda a inicialização foi concluida corretamente ou não. Este evento deve ser aguardado antes do envio de qualquer outro comando para a porta inicializada (port).

Antes de chamar o método ShutdownVoicerLib é necessário chamar o método DestroyGSMThread.

Copyright © 2010 Digivoice Eletrônica

Configurando a Thread de GSM

Alguns métodos foram desenvolvidos para a configuração da thread GSM.

Não é necessário iniciar a thread GSM para configurá-la, ou seja, os valores configurados serão assumidos ao criar a thread.

Porém se a thread estiver iniciada, e seja necessário configura-la, é necessário desabilitar a thread GSM com o método DisableGSMThread e após a configuração ter sido concluída, habilitar a thread (EnableGSMThread), dessa forma a thread será reiniciada com os valores configurados.

Métodos para configuração:

ConfigGSMThread: para as configurações de timeout de respostas de mensagens (GSMCFG_DIGIT_TIMEOUT), para restrição de envio de identificação de assinante em ligações de saída (GSMCFG_ID_RESTRICTION), para habilitar ou desabilitar o recebimento de notificação de segunda chamada (GSMCFG_CALL_WAITING_ENABLE), para configurar o tempo entre as tentativas de inicialização dos módulos (GSMCFG_RETRY_TIMEOUT) e para configurar o tempo de recebimento de resposta após o envio de um comando (GSMCFG_ANSWER_TIMEOUT).

GSMSetPinNumber: para a configuração do PIN Number quando a operadora de telefonia móvel exigir esta configuração.

Na utilização desse método não é necessário desabilitar e habilitar a thread GSM.

Mensagens SMS

O envio e recebimento de mensagens SMS é um recurso muito importante da placa VB0404GSM, possibilitando inúmeras funções de operação e manutenção de sistemas tais como envio de alarmes, comandos de reinicialização de servidores, além de servidores para prestadores de serviço de envio de SMS.

Copyright © 2010 Digivoice Eletrônica

As mensagens SMS não devem ultrapassar 160 caracteres. O gsm possui um set de caracteres que não é exatamente o set de caracteres normalmente utilizado em PCs. A VoicerLib faz uma tradução dos caracteres de ANSI para GSM no envio de mensagens SMS e de GSM para ANSI no recebimento de mensagens SMS. Programas cujo set de caracter seja diferente do ANSI poderão apresentar algumas letras de forma incorreta, principalmente as acentuadas, "ç" e alguns símbolos.

Envio de mensagens SMS

Para o envio de mensagem SMS foi criado o método GSMSendSMS. Neste método o número e um ponteiro para a string da mensagem a ser enviada são informados como parâmetros.

Obs. Tome cuidado ao enviar caracteres não imprimíveis ou cujo código ASCII ultrapasse 127.

Após o envio de uma mensagem SMS a VoicerLib enviará o evento OnGSMSMSSent indicando que a porta está pronta para nova operação.

Em caso de falha será gerado um evento de erro OnGSMError com o código de erro correspondente.

É possível receber a confirmação do recebimento de uma mensagem SMS por parte do destinatário através do evento OnGSMSMSConfirmation (este recurso depende da disponibilidade da operadora).

Para ler a string de confirmação do recebimento de uma mensagem SMS por parte do destinatário utilize o método GSMGetSMSConfirmation.

Recebimento de mensagem SMS

Ao receber uma mensagem SMS em uma das portas da placa VB0404GSM será gerado um evento OnGSMSMSReceived

Copyright © 2010 Digivoice Eletrônica

indicando que uma nova mensagem foi recebida.

Para ler a mensagem utilize o método GSMGetSMS, de preferência no tratamento do evento OnGSMSMSReceived para evitar a perda de alguma mensagem em caso de recebimento de várias mensagens SMS em um curto espaço de tempo.

Sempre que uma mensagem é recebida, esta é apagada do módulo GSM automaticamente se a thread GSM estiver rodando.

Listando o índice de mensagens e apagando mensagens SMS no módulo GSM

Se um módulo da placa receber uma mensagem SMS e a thread GSM não estiver ativa, a mensagem ficará armazenada na memória do módulo GSM.

Para poder apagar as mensagens armazenadas é preciso obter uma lista com os índices de cada mensagem armazenada com o método GSMListSMS. Após o recebimento das informações da lista de mensagens, a VoicerLib gera o evento OnGSMReturnOK.

A lista dos índices das mensagens poderá ser obtida através do método GSMGetIndexList.
Tais índices deverão ser utilizados nos métodos de apagamento das mensagens SMS (GSMDeleteSMS ou GSMReadAndDeleteSMS).

O método GSMClearAllSMS apaga todas as mensagens armazenadas no módulo GSM.

Atendimento de segunda chamada e conferência

A placa VB0404GSM permite o atendimento de mais que uma chamada por módulo e a conferência com até 7 participantes por porta GSM. Para utilizar estas facilidades é preciso se certificar que o serviço está disponível pela operadora de telefonia móvel e utilizar o método GSMCallControl.

Métodos especiais

Copyright © 2010 Digivoice Eletrônica

Alguns métodos e eventos foram criados principalmente para DEBUG e são apresentados a seguir.

OnGSMMessage é um evento enviado sempre que uma mensagem válida, não necessariamente SMS, é recebida por um módulo GSM. As mensagens recebidas podem ser obtidas pelo método GSMGetMessage.
Com esse evento é possível saber também, qual foi o último comando enviado para a placa, com o método GSMGetLastCommand.

GSMSendCommand envia um comando ao módulo da porta especificada. Este método deve ser utilizado com muito cuidado pois um comando errado pode causar mal funcionamento ou travamento dos módulos.

OnGSMTimeout, quando um comando não obtém resposta do respectivo módulo ou a resposta chega incompleta, este evento pode chegar juntamente com OnGSMError.

GSMCheckSignalQuality é usado para obter a qualidade do sinal recebido por um canal da placa VB0404GSM. Após o envio deste método é gerado um evento OnGSMSignalQuality indicando que, um string com o valor da qualidade de sinal pode ser obtido através do método GSMGetSignalQuality.

Os valores podem variar de 0 a 31, sendo 0 igual ou inferior a -113 dBm e 31 igual ou superior a -51 dBm.

Para reiniciar um módulo GSM em uma porta específica utilize o método GSMRestartPort.

A VoicerLib gera um evento na inserção ou retirada do SIM Card (chip) nos módulos GSM, permitindo por exemplo que o programador reinicie a thread GSM após a inserção de um chip na placa (OnGSMSIM).

Casos particulares

Copyright © 2010 Digivoice Eletrônica

Antes de iniciar uma discagem não é necessária a utilização de um comando Pickup, bastando a utilização direta do comando Dial.

Para sobrediscagem de DTMF após o atendimento, ou seja, em caso de acesso à URAs (IVR) é necessário utilizar o comando Dial com o parâmetro "dtDirectMF" como nas placas E1.

OBS: A placa VB0404GSM, devido às suas características particulares não admite o uso do método MakeCall da OCX (exclusivo para Windows).

Resumo dos métodos:

dg_ConfigGSMThread	(ConfigGSMThread)
dg_CreateGSMThread	(CreateGSMThread)
dg_DestroyGSMThread	(DestroyGSMThread)
dg_DisableGSMThread	(DisableGSMThread)
dg_EnableGSMThread	(EnableGSMThread)
dg_GSMCallControl	(GSMCallControl)
dg_GSMCheckSignalQuality	(GSMCheckSignalQuality)
dg_GSMClearAllSMS	(GSMClearAllSMS)
dg_GSMDeleteSMS	(GSMDeleteSMS)
dg_GSMGetIndexList	(GSMGetIndexList)
dg_GSMGetLastCommand	(GSMGetLastCommand)
dg_GSMGetMemory	(GSMGetMemory)
dg_GSMGetMessage	(GSMGetMessage)
dg_GSMGetSMS	(GSMGetSMS)
dg_GSMGetSMSConfirmation	(GSMGetSMSConfirmation)
dg_GSMGetSignalQuality	(GSMGetSignalQuality)
dg_GSMListSMS	(GSMListSMS)
dg_GSMReadAndDeleteSMS	(GSMReadAndDeleteSMS)
dg_GSMRestartPort	(GSMRestartPort)
dg_GSMSendCommand	(GSMSendCommand)
dg_GSMSendSMS	(GSMSendSMS)
dg_GSMSetPinNumber	(GSMSetPinNumber)

Resumo dos eventos:

EV_GSMCONFIRMATION	(OnGSMSMSConfirmation)

Copyright © 2010 Digivoice Eletrônica

EV_GSMERROR	(OnGSMError)
EV_GSMMEMORY	(OnGSMMemory)
EV_GSMMEMORYFULL	(OnGSMMemoryFull)
EV_GSMMESSAGE	(OnGSMMessage)
EV_GSMOTHERCALL	(OnGSMOtherCall)
EV_GSMREADY	(OnGSMReady)
EV_GSMRETURNOK	(OnGSMReturnOk)
EV_GSMSIM	(OnGSMSIM)
EV_GSMSIGNALQUALITY	(OnGSMSignalQuality)
EV_GSMSMSRECEIVED	(OnGSMSMSReceived)
EV_GSMSMSSENT	(OnGSMSMSSent)
EV_GSMTIMEOUT	(OnGSMTimeout)

Copyright © 2010 Digivoice Eletrônica

Gravação em Paralelo

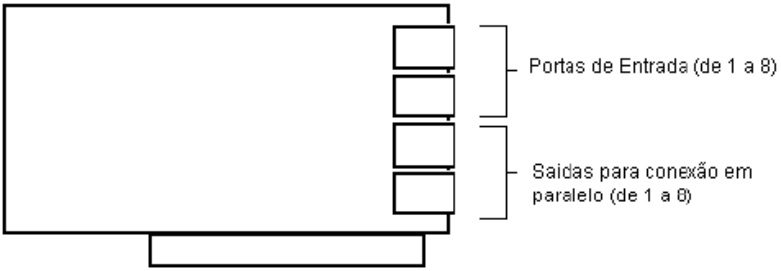
A gravação em paralelo é utilizada quando as placas DigiVoice ficam apenas monitorando (em paralelo) a linha, sem interferir no andamento da conversação. Neste caso, nunca utilize os métodos PickUp ou HangUp pois não se trata de uma ligação terminada em uma porta da placa.

Para montar a estrutura de gravação em paralelo é necessário que a forma como os cabos são conectados seja respeitada para cada tipo de placa.

Copyright © 2010 Digivoice Eletrônica

Placas FXO

A placa FXO pode ser configurada com 4 ou 8 canais. No primeiro e no segundo conector (de cima para baixo) estão as portas de 1 a 4 e de 5 a 8, respectivamente, sendo um par de fios para cada porta. O terceiro e quarto conectores devem ser utilizados para a saída da porta para a ligação em paralelo.



Placa FXO

Exemplo: A porta 1 entrará pelo primeiro par de fios do primeiro conector (mais acima) e sairá pelo primeiro par de fios do terceiro conector. (Em uma gravação terminada (sem paralelo), os dois conectores inferiores não são utilizados)

A explicação detalhada da conexão física está no manual do kit integrador no tópico da placa VB0408PCI e VB0408PCIe.

Ao tirar o telefone do gancho é gerado o evento OnLineReady e ao colocar o telefone no gancho é gerado o evento OnLineOff. Estes eventos deverão ser utilizados para iniciar (RecordFile) ou finalizar (StopRecordFile) a gravação. Para saber se determinada ligação é entrante ou sainte, o aplicativo deverá monitorar tons de linha,

Copyright © 2010 Digivoice Eletrônica

dígitos e a chegada de Bina. Não existe um método ou propriedade na VoicerLib que dê essa informação pronta ao programador.

Placas E1

A placa com a finalidade de gravação deverá ter obrigatoriamente 60 canais para que seja possível gravar 30 canais simultâneos. Utilizar as placas VB6060PCI/VB6060PCIe.

A conexão física, com a descrição dos cabos necessários é explicada no manual do Kit Integrador, que acompanha a placa e também está em nosso site www.digivoice.com.br.

A VoicerLib também gera os eventos OnLineReady e OnLineOff para a placa E1, mas isso não é automático e sua configuração difere bastante das placas FXO.

É muito importante entender que a placa E1 para gravação em paralelo será sempre uma placa de 60 portas (dois E1) mas disponibilizará somente as 30 primeiras portas para gravação. No caso da utilização de mais de uma placa, o programador deverá estar atento pois todos os eventos e métodos deverão referenciar apenas as 30 primeiras portas de cada placa portanto existirão apenas os canais 1-30, 61-90, ... e assim sucessivamente. Para evitar este "salto" de portas, o desenvolvedor e poderá utilizar os métodos de portas virtuais, onde é possível associar um número de porta qualquer a uma porta física da placa. Veja maiores detalhes no tópico "Portas Virtuais".

Para controlar a gravação, uma thread de controle deverá ser criada através do método CreateLoggerControl . Deve ser passado a porta e o tipo de protocolo utilizado. A VoicerLib suporta apenas o RD2 MF, portanto o segundo parâmetro receberá sempre a constante LOGGER_R2DMF (100).

Copyright © 2010 Digivoice Eletrônica

Ao criar esta thread de controle, todos as portas ficarão em estado de espera, monitorando a linha. Quando uma ligação estiver sendo iniciada (entrante ou sainte) a thread irá monitorar toda a troca de sinalização e o evento OnLoggerEvent será gerado diversas vezes, indicando para a aplicação as diversas etapas até o início da conversação propriamente dito.

O evento OnLoggerEvent receberá o parâmetro LoggerStatus, que poderá assumir os seguintes valores:

- **LOGGER_FREE_WITH_BILLING** - Linha de Assinante Livre com Tarifação
- **LOGGER_BUSY** - Porta ocupada
- **LOGGER_NUMBER_CHANGED** - Linha de assinante mudado
- **LOGGER_CONGESTION** - Congestionamento
- **LOGGER_FREE_WITHOUT_BILLING** - Assinante Livre sem tarifação
- **LOGGER_FREE_RETENTION** - Assinante Livre com tarifação e colocar retensão sob controle do assinante chamado
- **LOGGER_LEVEL_NUMBER_AVAILABLE** - Nível ou Número Vago
- **LOGGER_B_ENDCALL** - Assinante B desligou. Quando o assinante B desliga não significa necessariamente que a ligação terminou
- **LOGGER_B_RETURN** - Assinante B retornou a ligação. Ocorre após o LOGGER_B_ENDCALL caso o assinante B retorna à ligação
- **LOGGER_LINEREADY *(*)*** - LineReady significa o local do início efetivo da conversação
- **LOGGER_LINEOFF *(*)*** - LineOff significa o local do término efetivo da conversação

(*) O início e fim de conversação poderá ser tratado tanto nos eventos OnLineReady e OnLineOff como nos status LOGGER_LINEREADY e LOGGER_LINEOFF.

No início da conversação, a VoicerLib permite que seja extraído o número discado (através do método GetE1Number**)**, a identificação de A (método GetCallerID**)** e também se a ligação é de entrada ou saída (método GetLoggerCallType). Observe o

Copyright © 2010 Digivoice Eletrônica

fragmento de código a seguir, que mostra todas as informações necessárias, inicia e termina a gravação:

(ActiveX)

```
Private Sub vlib_OnLoggerEvent(ByVal Port As Integer, ByVal LoggerStatus
As Integer)

Dim sBina As String, sE1 As String, sTipo As String
Dim nRet As Integer

  Select Case LoggerStatus
    Case LOGGER_LINEREADY:
      sE1 = vlib.GetE1Number(Port)    'pega número do E1
      sBina = vlib.GetCallerID(Port)  'pega BINA

      If vlib.GetLoggerCallType(Port) = INCOMINGCALL Then
        sTipo = "Entrante"
      Else
        sTipo = "Sainte"
      End If

      MostraStatus Port, "Logger LineReady (" + sTipo + ") (Caller ID: "
& sBina & ", E1 Number: " & sE1 & ")"
      vlib.EnableInputBuffer Port, DG_ENABLE
      'Aqui inicia a gravação
      nRet = vlib.RecordFile(Port, App.Path & "\grava" & Port & ".gsm",
"")
      If nRet <> DG_EXIT_SUCCESS Then
        MostraStatus Port, "ERRO: Não foi possível gravar arquivo"
      Else
        MostraStatus Port, "Iniciando Gravação"
      End If

    Case LOGGER_LINEOFF:
      vlib.DisableInputBuffer Port
      'Termina gravação
      nRet = vlib.StopRecordFile(Port)

      If nRet <> DG_EXIT_SUCCESS Then
        MostraStatus Port, "ERRO ao tentar interromper gravação"
      Else
        MostraStatus Port, "Gravação Finalizada"
      End If

    End Select

End Sub
```

(API)

```
void ReceiveEvents(void *context_data)
{
  struct dg_event_data_strucuture *EventContext;
  char szCallerID[25];
  char szE1Number[25];
```


Copyright © 2010 Digivoice Eletrônica

```
  char szFileName[255];

  short ret;

    /* Copy received Data */
    EventContext = ((struct dg_event_data_strucuture*)context_data);

      switch (EventContext->command)
      {
        case EV_LOGGEREVENT:
          switch(EventContext->data)
          {
            case LOGGER_LINEREADY:
              dg_getcallerid(EventContext->port,szCallerID); //pega
número do E1
              dg_gete1number(EventContext->port,szE1Number); //pega BINA
              printf("LineReady (Caller ID: %s, E1 Number:
%s)\n",szCallerID,szE1Number);

              if (dg_getloggercalltype==INCOMINGCALL)
                printf("Ligação Entrante\n");
              else
                printf("Ligação Sainte\n");

              //Cria arquivo de gravacao
               dg_setrecordformat(EventContext->port,ffGSM610);
               sprintf(szFileName,"grava%d.gsm", EventContext->port);
               //Aqui inicia a gravação
             ret = dg_recordfile(EventContext->port,szFileName,"");

            if (ret !=DG_EXIT_SUCCESS)
                printf("Nao foi possivel iniciar gravacao");
          break;

          case LOGGER_LINEOFF:
            printf("LineOff - Fim da gravacao");
            //Termina gravação
             ret = dg_stoprecordfile(EventContext->Port);
           if (ret !=DG_EXIT_SUCCESS)
               printf("Nao foi possivel finalizar gravacao");

          break;
        }
      break;
  }
}
```

Sempre deverá ser chamado o método DestroyLoggerControl para fechar as threads de controle antes de finalizar a VoicerLib.

Copyright © 2010 Digivoice Eletrônica

Programação das Placas E1

Em um feixe E1 podemos ter vários protocolos de sinalização. Esta documentação diz respeito à sinalização de linha R2D e de registro MFC 5C e à sinalização ISDN PRI. Cada feixe E1 comporta 30 canais , mas como podemos ter 2 E1 em uma placa e mais que uma placa em um PC trataremos como porta o número sequêncial de canais nos vários E1, ou seja, podemos ter porta 1,2,3....121,122,123 etc.

Copyright © 2010 Digivoice Eletrônica

Configurações de Sincronismo

Devido à características existentes somente nas linhas digitais (E1) , o desenvolvedor deve prestar atenção às configurações de sincronismo após a inicialização. Sempre após inicializar o driver (StartVoicerLib) é necessário configurar o modo de operação e o sincronismo das placas. Caso esta configuração seja omitida, a aplicação poderá apresentar erros de sinalização, etc...

O sincronismo dependerá basicamente do ambiente onde as placas estão ligadas (tronco E1). Normalmente, se as placas estão conectadas diretamente à rede pública, esta fornece o sincronismo necessário, portanto as placas deverão ser configuradas com sincronismo EXTERNO. Em ambientes onde a placa E1 seja a responsável por fornecer o sincronismo, este deverá ser configurado como INTERNO.

O método SetCardSyncMode tem apenas 2 parâmetros, sendo que o primeiro parâmetro é a placa na qual será aplicada a configuração e o segundo indicará qual o modo de operação será configurado, veja a seguir:

- MASTER_INTERNAL_SYNC - Placa MASTER com sincronismo interno
- MASTER_SYNC_A - Placa MASTER com sincronismo externo no E1 A (Primeiro E1)
- MASTER_SYNC_B - Placa MASTER com sincronismo externo no E1 B (Segundo E1, em placas de 60 canais)

Uma forma de verificar se o sincronismo está configurado de maneira errada é a existência de eventos OnE1Alarm, principalmente com o status ALARM_RSLIP.

Exemplo (ActiveX):

```
Private Sub Iniciar()
Dim nRet as Integer

  'Inicia Driver
```


Copyright © 2010 Digivoice Eletrônica

```
  nRet = VoicerLib.StartVoicerLib
  If nRet = DG_EXIT_SUCCESS then
  'Placa iniciada com sucesso, configura sincronismo
interno na placa 1
    VoicerLib.SetCardSyncMode 1, MASTER_INTERNAL_SYNC
  End If

End Sub
```

Alarmes

A placa E1 gera eventos de alarmes indicando qualquer tipo de problema nos troncos E1. Estes alarmes podem ser causados por falhas de configuração de sincronismo, discutido no tópico anterior, ou por algum tipo de problema de conexão física (cabos, etc...). A monitoração dos alarmes é essencial para o funcionamento das aplicações que utilizam a placa E1.

Por padrão inicial a notificação dos alarmes é manual, ou seja, a placa não envia estes alarmes para a VoicerLib e esta consequentemente não gera o evento OnE1Alarm. Se for necessário saber o status do alarme, é necessário chamar o método GetAlarmStatus.

Este comportamento pode ser alterado através do método SetAlarmMode na qual passa-se a placa e o modo de operação que pode ser manual (padrão) ou automático. Como já foi dito, no modo manual (ALARM_MANUAL_NOTIFY), o evento OnE1Alarm só é gerado após a chamada do método GetAlarmStatus. Já no modo automático (ALARM_AUTOMATIC_NOTIFY), o evento OnE1Alarm ocorre sempre quando um alarme for detectado. É recomendado utilizar o modo automático mas somente depois de configurar o sincronismo das placas.

Os tipos de alarme possíveis são:

- ALARM_RSLIP (0x01) - Escorregamento (problema de sincronismo)
- ALARM_RAIS (0x02) - Alarme remoto

Copyright © 2010 Digivoice Eletrônica

- ALARM_AISS (0x04) - Indicação de alarme
- ALARM_AIS16S (0x08) - Indicação de alarme canal 16
- ALARM_LOSS (0x10) - Perda de sinal
- ALARM_CRC4SYNC (0x20) - Alarme de CRC4
- ALARM_MFSYNC (0x40) - Sincronismo de multiquadro
- ALARM_SYNC (0x80) - Sincronismo de quadro
- ALARM_LINK (0x100) - Sincronismo do protocolo ISDN

Copyright © 2010 Digivoice Eletrônica

Protocolo CAS Customizado

Introdução CAS Customizado

As placas E1 podem ser utilizadas também em posição de ramal, em PABX que oferecem este recurso. Normalmente, existem protocolos específicos por fabricante para emular a operação de um telefone analógico (Atender, Desligar, Flash, etc...).

A VoicerLib oferece alguns métodos que permitem implementar essas funcionalidades:

CreateCustomCAS
DestroyCustomCAS
ConfigCustomCAS

Nem todas as situações de ramais CAS poderão estar previstas nesses métodos. Novas implementações serão feitas pela DigiVoice à medida que novos protocolos apareçam.

Copyright © 2010 Digivoice Eletrônica

Protocolo ISDN PRI

Inicialização Protocolo ISDN PRI

O protocolo ISDN PRI é muito utilizado na comunicação entre centrais telefônicas. A VoicerLib permite a criação de threads que interagem com centrais públicas e PABX utilizando esse protocolo por meio de troncos E1.

Após iniciar o driver pelo método StartVoicerLib, é necessário configurar o modo de operação da placa E1, habilitando o uso de CRC4 pelo método SetE1CRC4Option, e indicando a opção adequada de sincronismo pelo método SetCardSyncMode.

Antes de criar threads ISDN PRI é também necessário configurar o modo de utilização do protocolo, por meio do método ConfigISDNThread. Estas configurações devem ser feitas por span.

As configurações necessárias são:

- **ISDN_SET_LOCALTYPE**: indica o tipo do ambiente, podendo ser do tipo *network* (NET) ou do tipo *user* (CPE). Normalmente, a central pública é configurada como NET e o PABX como CPE. O ambiente do tipo CPE funciona apenas com o tipo NET e vice-versa;

- **ISDN_SET_SWITCHTYPE**: indica o padrão do protocolo ISDN PRI. A implementação do protocolo ISDN PRI feita na VoicerLib é baseada no padrão europeu, portanto, deve-se configurar como ISDN_EUROISDN_E1;

- **ISDN_SET_MIN_DIGITS**: indica o número mínimo de dígitos que serão aceitos em chamadas entrantes;

- **ISDN_SET_MAX_DIGITS**: indica o número máximo de dígitos que serão aceitos em chamadas entrantes;

- **ISDN_SET_AUTO_SIGNALING**: quando habilitado, indica que

Copyright © 2010 Digivoice Eletrônica

a VoicerLib irá tratar a sinalização de chamadas entrantes automaticamente, sem qualquer interferência da aplicação. Quando desabilitado, a aplicação poderá interagir no processo de aceitação de chamadas entrantes por meio da leitura dos dígitos recebidos. Veja mais detalhes sobre esta opção no tópico Recebendo chamadas;

- **ISDN_SET_EXCLUSIVE_CHANNEL**: quando habilitado, a VoicerLib somente irá estabelecer a chamada na porta indicada ao fazer uma chamada sainte. Caso a porta indicada já esteja sendo utilizada, a chamada sainte não será completada. Quando desabilitada, a VoicerLib tentará estabelecer a chamada na porta indicada. Se o PABX ou a Central Pública indicar que deseja estabelecer a chamada em outra porta, a VoicerLib aceitará. Neste caso, a VoicerLib indica a troca de portas por meio do evento OnChannel.

- **ISDN_SET_GENERATE_TONE**: quando habilitado, a VoicerLib gera tom de chamando para o PABX ou a central pública quando recebe uma chamada.

Uma vez feita a configuração das threads ISDN PRI, é possível criá-las por meio do método CreateISDNThread. Deve ser criada uma thread para cada span. Caso seja necessário, é possível destruir uma thread ISDN PRI por meio do método DestroyISDNThread.

Caso a aplicação já tenha criado alguma thread ISDN PRI e for necessário alterar qualquer configuração, é possível desabilitar a thread desejada pelo método DisableISDNThread, reconfigurá-la usando o método ConfigISDNThread e habilitá-la novamente pelo método EnableISDNThread. Este processo faz com que todas as portas referentes ao span selecionado sejam resetadas.

Ao se executar o método CreateISDNThread para criar thread ISDN, a VoicerLib habilita a criação de um arquivo de texto no diretório *"c:/log"*, que contém toda a sinalização ISDN transmitida e recebida pelas placas Digivoice com interface E1.

Esse arquivo é nomeado como "isdn-<placa>_.log". Por

Copyright © 2010 Digivoice Eletrônica

exemplo, a sinalização referente à placa 1 e ao span 2 será gravada no arquivo *"c:/log/isdn-1_2.log"*.

Obs.:O diretório "c:/log" não é criado automaticamente e deve ser criado manualmente. Se ele existir, o arquivo será criado. Caso contrário, nada acontecerá, ou seja, a sinalização não será gravada em log.

Não é aconselhável deixar este diretório criado por um período longo de tempo, uma vez que o tamanho do arquivo pode crescer substancialmente. Isso pode causar problemas de sobrecarga ao disco rígido e, portanto, compromer o funcionamento da máquina e da aplicação.

Copyright © 2010 Digivoice Eletrônica

Funções de Controle da Thread ISDN PRI

Em algumas situações pode ser necessário parar a thread ISDN PRI para efetuar algum controle manualmente. Para isso foram criados os métodos EnableISDNThread e DisableISDNThread. Com eles é possível desabilitar a thread ou reabilitá-la sem ter que destrui-la ou recriá-la, pois são processos com maior consumo de processamento. Nos dois métodos, são passados como parâmetros a placa e o span.

Ao chamar o método de criação da thread CreateISDNThread não é necessário habilitá-la, pois ao criar, ela estará automáticamente habilitada. O método EnableISDNThread só é útil após o uso de um DisableISDNThread.

Também é possível saber se a thread ISDN PRI está habilitada ou não através do método GetISDNThreadStatus. Este método recebe a placa e o span como parâmetro e devolve os seguintes valores:

DG_E1_THREAD_ENABLED (1) - Thread habilitada
DG_E1_THREAD_DISABLED (0) - Thread desabilitada

Copyright © 2010 Digivoice Eletrônica

Efetuando Chamadas

Após iniciar o driver pelo método StartVoicerLib e criar as threads ISDN PRI, é possível efetuar chamadas.
Para efetuar chamadas, pode-se configurar certas particularidades, são elas:

- Tipo de chamada utilizada: indica particularidades da chamada, tal como o codec utilizado, por meio do método SetBearerCapability;
- Tipo do número local: indica particularidades do ramal local, tal como o tipo de número, por meio do método SetCallingNumber;
- Tipo do número remoto: indica particularidades do ramal remoto, tal como o tipo de número, por meio do método SetCalledNumber;
- Envio de identificação do número (BINA): especifica a identificação enviada na chamada sainte pelo método SetPortID;
- Envio do nome da identificação do número: especifica o nome da identificação do número enviado na chamada sainte pelo método SetNameID;
- Envio de mensagem de texto: especifica uma mensagem de texto qualquer enviada na chamada sainte pelo método SetText.

Depois das configurações realizadas, as informações ficam armazenadas em memória pela VoicerLib e serão utilizadas em todas as chamadas saintes daquela porta.

Para inicar uma chamada, é necessário alocar uma porta utilizando o método PickUp. Se este método retornar DG_ERROR_PORT_ALREADY_IN_USE, indica que a porta já está sendo utilizada por outra chamada. Se ele retornar DG_EXIT_SUCCESS, a porta foi alocada e, após ocorrer o evento OnDialToneDetected, é possível efetuar a chamada utilizando o método Dial. A VoicerLib irá utilizar as últimas configurações de chamada feitas antes do método Dial ser utilizado, tal como SetPortID, SetNameID, SetBearerCapability...

Quando o número discado estiver chamando, a VoicerLib irá indicar para a aplicação por meio do evento OnCalling a cada 5

Copyright © 2010 Digivoice Eletrônica

segundos. Quando o número discado atender a chamada, a VoicerLib irá indicar para a aplicação por meio do evento OnAnswerDetected.

É obrigatório chamar o método ConfigISDNThread antes a chamada do CreateISDNThread.

Copyright © 2010 Digivoice Eletrônica

Recebendo Chamadas

Após iniciar o driver pelo método StartVoicerLib e criar as threads ISDN PRI, é possível receber chamadas.

Neste caso, há dois tipos de recebimento de chamadas entrantes. A aplicação escolhe aquela que irá tratar por meio da configuração feita pelo método ConfigISDNThread e pelo parâmetro ISDN_SET_AUTO_SIGNALING. Caso a aplicação não especifique o modo de recebimento, a VoicerLib habilita por padrão o modo automático.

- Modo automático: Ao receber uma chamada entrante, a VoicerLib trata a sinalização sem qualquer tipo de interferência da aplicação. Sendo assim, se o número de dígitos da chamada entrante estiver no intervalo entre o mínimo e o máximo configurado pela aplicação, a VoicerLib recebe a chamada independentemente do número recebido. Este recebimento é indicado pelo evento OnRingDetected a cada 5 segundos. Para atender a chamada, a aplicação deve utilizar o método PickUp.

- **Modo manual**: Ao receber uma chamada entrante, a VoicerLib identifica o número recebido, o armazena em memória e informa a aplicação por meio do evento OnE1StateChange. A aplicação deve obter o número utilizando o método GetE1Number e verificá-lo.
Caso necessite de mais dígitos, a aplicação deve informar a VoicerLib usando o método SendISDNCommand com o parâmetro C_WAITFORDIGITS. Neste caso, a VoicerLib espera por mais dígitos e, quando eles chegarem, ela informa a aplicação com o mesmo evento, OnE1StateChange. Este ciclo se repete até que a aplicação indique para a VoicerLib que o número discado está correto, por meio do método SendISDNCommand com o parâmetro C_CALL_PROCEEDING.
A seguir, a aplicação pode rotear a chamada para outra porta (independentemente se é uma porta E1, FXS, FXO ou GSM). Quando esta porta estiver chamando, a aplicação deve avisar a VoicerLib que a chamada da porta cujo protocolo é o ISDN PRI

Copyright © 2010 Digivoice Eletrônica

está chamando, por meio do método SendISDNCommand e o parâmetro C_RING. Quando o número discado tiver atendido a chamada, a aplicação deve atender a porta com protocolo ISDN PRI usando o método PickUp.

Independentemente do modo de recebimento de chamadas configurado, algumas informações extras podem ser recebidas pela VoicerLib, tais como uma mensagem de texto e a identificação do número de A (BINA). Algumas dessas informações podem ser obtidas utilizando um método específico após a indicação de um evento por parte da VoicerLib.

- OnCallerID: Indica o recebimento da identificação do número de A, que pode ser obtido pelo método GetCallerId;
- OnNameID: Indica o recebimento do nome da identificação do número de A, que pode ser obtido pelo método GetNameId;
- OnText: Indica o recebimento de uma mensagem de texto, que pode ser obtido pelo método GetText;
- OnReverseCharging: Indica o pedido de inversão de cobrança da chamada (chamada à cobrar);

Copyright © 2010 Digivoice Eletrônica

Finalizando Chamadas

A aplicação pode indicar o motivo do desligamento utilizando o método SetHangUpCause e pode finalizar uma chamada utilizando o método HangUp. Se isso acontecer, a VoicerLib envia um pedido de desligamento para o lado remoto.

Quando a VoicerLib tiver a confirmação de que o desligamento foi aceito, ela reseta as informações da chamada e confirma para a aplicação que a porta está livre por meio do evento OnE1StateChange com o parâmetro C_E1_IDLE.
Somente com esta confirmação que a aplicação pode tentar utilizar a porta indicada para uma nova chamada.

Se em algum momento a chamada for desligada por outro agente diferente da aplicação, a VoicerLib envia o evento OnBusyDetected para a aplicação. Lembramos que este evento pode ocorrer inclusive durante o estabelecimento e o recebimento de chamadas. Ao receber esse evento, a aplicação pode obter a causa do desligamento por meio do método GetBusyCause. A seguir, a aplicação deve desligar a chamada pelo método HangUp. Quando a porta estiver livre e pronta para novas chamadas, a VoicerLib indica isso para a aplicação por meio do evento OnE1StateChange com o parâmetro C_E1_IDLE.

Copyright © 2010 Digivoice Eletrônica

Protocolo R2D MFC

Inicialização Protocolo R2D MFC

Para efetuar ou receber chamadas, a VoicerLib já traz embutida uma thread que executa o protocolo RD2 MFC 5C, a qual permite interagir com as centrais públicas, PABX com troncos E1, etc...

Após iniciar o driver através do método StartVoicerLib , é necessário iniciar uma thread individual para cada porta. Esta thread inicia internamente todo o tratamento de troca de sinalização com a "outra ponta" da ligação. Para iniciar esta thread, é utilizado o método CreateE1Thread, passando como parâmetro a porta desejada. Ao iniciar a thread, a porta é automaticamente colocada como Livre.

Também é necessário configurar corretamente as opções de sincronismo e modo de operação das placas E1. É recomendado utilizar o VBoxConfig (configurador da placa E1 que é instalado com a voicerlib) para configurar todas estas opções.

Copyright © 2010 Digivoice Eletrônica

Efetuando Chamadas

Sempre que for efetuar ligações em troncos E1 R2D MFC, é necessário que a porta do E1 tenha capacidade de enviar dados como sua identificação, categoria de assinante (grupo II), etc...

A identificação de A da porta (BINA a ser fornecido ao outro terminal) é configurada através do método SetPortId passando como parâmetro a porta e uma string com o número desejado (ex. setportid(porta,"1121916363")). Cada porta tem uma identificação própria na VoicerLib.

Todas os parâmetros de configuração relacionados ao protocolo R2D MFC são configurados, sempre após chamar o método CreateE1Thread (início da thread), através do método ConfigE1Thread . O primeiro parâmetro é a porta, o segundo indica o dado a configurar e o terceiro o valor deste dado. O dado a configurar é indicado pelas constantes mostradas abaixo:

– E1CFG_MAXDIGITS_RX - Número máximo de dígitos a receber
- E1CFG_SEND_ID_AFTERDIGIT - Envio de solicitação de identificação após o dígito n
- E1CFG_GROUP_B - Grupo B
- E1CFG_GROUP_II - Categoria do Assinante

É obrigatório chamar o método ConfigE1Thread após a chamada do CreateE1Thread.

Estando as informações sobre a porta corretamente configuradas, efetuar uma ligação. Para ocupação utilize o método Pickup. Logo que vier a confirmação de ocupação, indicando que a porta está livre para discar, será gerado o evento OnDialToneDetected. No protocolo R2D MFC, não é gerado um tom de discagem como se conhece na telefonia convencional mas o evento de tom de discagem é gerado para compatibilizar as aplicações e informar ao programa a hora certa de iniciar a discagem.

Copyright © 2010 Digivoice Eletrônica

No evento OnDialToneDetected, é necessário chamar o método Dial indicando o número a discar. O primeiro parâmetro do método é a porta, o segundo indica o número e o terceiro, que seria a pausa após a discagem, é desprezado no caso de discagens com a thread R2D MFC criada.

A partir daí, ocorre toda uma troca de sinalização entre os terminais. Essa troca de sinalização pode ser monitorada através do evento OnE1StateChange. Este evento passa para a aplicação o estado da chamada na variável Status, que pode ter os seguintes valores:

- C_GRUPO_B - Solicita grupo B para aplicação
- C_NOTCOMPLETED - Ligação não completada
- C_B_ENDCALL - Assinante B desligou
- C_ANSWERED - Assinante B atendeu ou retornou depois de C_B_ENDCALL
- C_CONGESTION - Congestionamento
- C_SEIZURE - Ocupação
- C_GROUP_II - Grupo II recebido
- C_NUMBER_RECEIVED - Número recebido durante ligação entrante
- C_UNAVAILABLE - Porta não disponível
- C_GROUP_I - Recebeu grupo I
- C_ID_RECEIVED - Recebeu identificação de assinante A
- C_E1_IDLE - Porta R2 foi para o estado LIVRE
- C_E1_SEIZURE_ACK - Porta R2 recebeu confirmação de ocupação
- C_SEND_GROUP_B - Recebeu grupo B da aplicação
- C_SEND_BACKWARD_SIGNAL - Recebeu comando de envio de sinal "para trás" da aplicação

Mais uma vez simulando uma ligação analógica convencional, a VoicerLib gerará os seguintes eventos a partir da discagem:

- OnAfterDial - Término da discagem
- OnBusyDetected - Linha ocupada, porta bloqueada ou timeout de atendimento
- OnAnswerDetected - Destino atendeu

Copyright © 2010 Digivoice Eletrônica

- OnCalling - Ligação chamando. Este evento é gerado a cada 5 segundos.

Repare que estes eventos poderão ocorrer em situações iguais às indicadas pelos Status de OnE1StateChange. Por exemplo, o evento OnBusyDetected também ocorrerá quando for detectado uma situação de porta não disponível (C_UNAVAILABLE).

A qualquer momento durante a chamada, é possível chamar o método HangUp para finalizá-la.

Quando se faz ligações com as placas analógicas, não há como perceber se o assinante B desligou já que o telefone fica mudo até que passe os 90 segundos e venha o tom de ocupado da central pública. Na placa E1 é possível saber quando o assinante B desligou, basta monitorar o status C_B_ENDCALL dentro do evento OnE1StateChange. Isso é muito útil em discadores que reproduzem mensagens automáticas, por exemplo. Caso o assinante B retorne à ligação, é gerado o evento OnE1StateChange com valor C_ANSWERED. Este evento também acontece quando ocorre o primeiro atendimento.

Copyright © 2010 Digivoice Eletrônica

Recebendo Chamadas

O recebimento de chamadas através de uma porta digital também é facilmente manipulado a partir da chamada do método CreateE1Thread.

Para ligações entrantes é necessário chamar o método ConfigE1Thread para configurar:

- E1CFG_MAXDIGITS_RX - Número máximo de dígitos a receber
- E1CFG_SEND_ID_AFTERDIGIT - Envio de solicitação de identificação após o dígito n (Valor 0 não pede ID)
- E1CFG_GROUP_B - Grupo B
- E1CFG_GROUP_II - Categoria do Assinante

O E1CFG_MAXDIGITS_RX deve ser informado porque o protocolo R2D exige a quantidade de dígitos que deverão ser detectados. Já o E1CFG_SEND_ID_AFTERDIGIT indicará para a thread R2D MFC o momento da solicitação da identificação do chamador. O Grupo B e o Grupo II variam conforme a aplicação e vêem com valor padrão 1.

Durante o recebimento de uma chamada, ocorre toda a troca de sinalização de linha R2D e de registro MFC 5C, conforme explicado no começo deste capítulo. Vários destes eventos podem ser monitorados através do evento OnE1StateChange. Para simplificar a detecção da chegada de uma ligação, a VoicerLib também gera um evento OnRingDetected logo após da troca de sinalização, permitindo que a aplicação atenda a chamada.

O atendimento de uma ligação entrante se dará a partir da chamada do método PickUp e sua finalização através do método HangUp.

Caso o chamador desligue, será gerado o evento OnBusyDetected para a aplicação.

A identificação de chamadas (BINA) é feita pelo método GetCallerID. O evento OnCallerID sinaliza quando esta informação

Copyright © 2010 Digivoice Eletrônica

já está disponível.

Funções de Controle da Thread R2

Em algumas situações pode ser necessário parar a thread R2D MFC para efetuar algum controle manualmente. Para isso foram criados os métodos EnableE1Thread e DisableE1Thread. Com eles é possível desabilitar a thread ou reabilitá-la sem ter que destruí-la ou recriá-la, pois são processos com maior consumo de processamento. Nos dois métodos o único parâmetro é a porta.

Ao chamar o método de criação da thread CreateE1Thread não é necessário habilitá-la, pois ao criar, ela estará automáticamente habilitada. O método EnableE1Thread só é útil após o uso de um DisableE1Thread

Também é possível saber se a thread R2D MFC está habilitada ou não através do método GetE1ThreadStatus . Este método recebe a porta como parâmetro e devolve os seguintes valores:

DG_E1_THREAD_ENABLED (1) - Thread habilitada
DG_E1_THREAD_DISABLED (0) - Thread desabilitada

Copyright © 2010 Digivoice Eletrônica

Depurando aplicativos para placas E1 com R2MFC

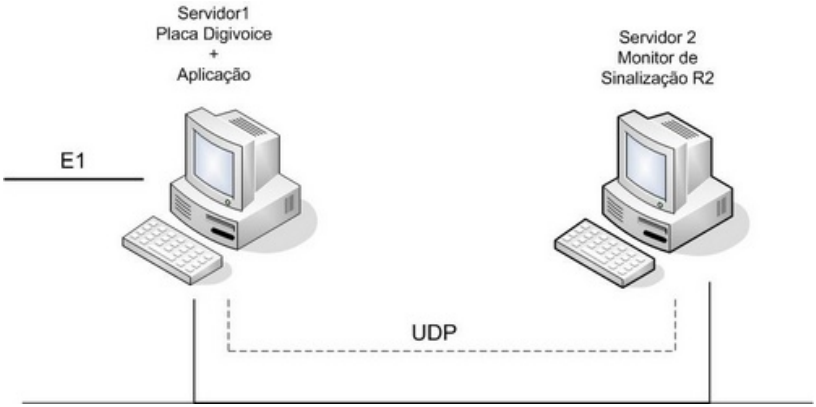
Liberado aplicativo para monitoração remota de sinalização R2MFC.

Sistema Operacional: Windows
Protocolo de Rede: UDP

Para habilitar depuração, chame o método EnableDebug na aplicação sendo desenvolvida.

Para desabilitar depuração chame o método DisableDebug na aplicação sendo desenvolvida.

Obs. É necessário que a porta 623 UDP esteja habilitada no firewall do Windows.



Copyright © 2010 Digivoice Eletrônica

Funções Especiais

Introdução Funções Especiais

As funções especiais são destinadas a executar tarefas comuns em telefonia de maneira mais simples para o programador.

Todas estas tarefas são plenamente realizáveis através das funções primitivas (PickUp, Dial, etc), porém procuramos reunir as tarefas mais comuns em grupos específicos:

- Funções de Idle – Atendimento Automático, Bina, Integração com PABX
- Funções de Prompt – Entrada de dados com ou sem conferência e confirmação (Senhas, etc...)
- Funções de Menu – Reprodução de menu de opções com consistência
- Funções de Discagem e Transferência – Discagem através de linha direta ou ramal, com opções para supervisão de ocupado, etc...

Os métodos e eventos apresentados a seguir são sempre indexados pelo parâmetro Port o que significa que as configurações são totalmente independentes por porta.

Para o pleno entendimento destas novas funcionalidades é muito importante estudar atentamente os exemplos fornecidos em Delphi7, Visual Basic 6 e VB.NET que encontram-se no site da DigiVoice www.digivoice.com.br, na seção de suporte.

Copyright © 2010 Digivoice Eletrônica

Funções de Idle

As funções de Idle permitem ao programador simplificar os processos de espera de ligações:

- Atendimento Automático após n toques
- Detecção de Identificação do Assinante A (Bina) através do evento OnCallerID
- Detecção de Dígitos após o atendimento, facilitando integrações com diversos modelos de Pabx.

As funções de Idle utilizam os métodos: IdleSettings, IdleStart e IdleAbort (ou suas correspondentes dg_XXXX disponíveis na API).

Atendimento Automático

IMPORTANTE: Os métodos de IdleXXXX são exemplificados em um programa específico encontrado no site da DigiVoice www.digivoice.com.br, na seção de suporte. Acompanhe e teste o exemplo.

A VoicerLib permite ao programador, a partir de simples parâmetros no método IdleSettings, configurar um atendimento automático a partir do enésimo toque e ainda esperar um tempo pré-definido antes de chamar o evento OnAfterPickup.

Os parâmetros do IdleSettings que interferem no funcionamento deste método são o segundo, terceiro e quarto, sendo:

- 2º Parâmetro (AutoPickUp) – Campo booleano (true/false) que indica se atende automático ou não
- 3º Parâmetro (RingCount) – Número de toques para o atendimento
- 4º Parâmetro (PauseAfterPickUp) – Pausa após o atendimento. Esta pausa pode ser utilizada para dar um tempo antes de iniciar a reprodução da mensagem de boas vindas, o que é útil em atendedores com bloqueio DDC (Chamada a cobrar).

Copyright © 2010 Digivoice Eletrônica

Monitoração de Dígitos

A monitoração de dígitos pode ser feita antes do atendimento, como nos casos de identificação de chamadas direto do tronco ou depois do atendimento, particularmente útil em integração do tipo inband com centrais PABX.

Estas configurações também são feitas a partir do método IdleSettings do 5º. ao último parâmetro:

- 5º Parâmetro (WatchTrunkBefore) – Indica que deve ser acionada a monitoração de dígitos antes do Ring. Deve ser utilizado principalmente em casos de identificação de chamadas.
- 6º Parâmetro (WatchTrunkAfter) – Indica que deve ser acionada a monitoração de dígitos depois do atendimento.
- 7º Parâmetro (Format) – O formato indica como a espera de dígitos será tratada:

wtDTMF/wtMFP – Deve ser utilizado somente com o WatchTrunkBefore ligado para tratamento de Bina DTMF (wtDTMF) ou Bina MFP (wtMFP).Neste caso será gerado o evento OnCallerID contendo o número identificado.
A detecção de BINA no formato FSK não depende do método IdleSettings e sim do método EnableFSKDetection.

wtCustom – Este formato traz os dígitos exatamente como foram detectados, sem nenhum tratamento. O resultado é dado no evento OnDigitsReceived. Funciona praticamente igual a um GetDigits.

- 8º Parâmetro (TimeOut) – É o tempo máximo que o sistema esperará pelos dígitos a partir do primeiro detectado. Funciona como parâmetro InterdigitTimeout do método GetDigits e só tem utilidade quando o formato for wtCustom.
- 9º Parâmetro (Max) – Indica qual o número máximo de dígitos que o sistema deverá esperar. Se não for utilizado, deve ser setado como 0 (zero).
- 10º Parâmetro (TermDigits) – Indica qual o dígito terminador. Deixar vazio se não for utilizado.

Copyright © 2010 Digivoice Eletrônica

Funções de Prompt

As funções de Prompt, disponíveis somente no ActiveX, facilitam os processos de entrada de dados e conferência em uma aplicação de URA. Como exemplo de utilização, observe a frase abaixo:

*"Digite sua senha ..... Você digitou 123.... Tecle # para confirmar ou * para cancelar"*
"Disque o ramal desejado...."

Para que os dígitos sejam reproduzidos corretamente, você deverá configurar corretamente a propriedade StockSigsPath apontando para a pasta onde os arquivos wave padrão estão armazenados. Esta pasta está localizada no diretório de instalação da Voicerlib

Basicamente as funções de prompt permitem reproduzir uma mensagem inicial (ou lista de mensagens), esperar por dígitos específicos (maxdigits, termdigits, etc...), reproduzir opcionalmente uma mensagem de conferência com as informações digitadas e ainda confirmar a digitação, permitindo a reentrada dos dados. Tudo isto pode ser feito através de parâmetros passados aos métodos PromptSettings e PromptStart. Exemplos destas funções encontram-se no site da DigiVoice www.digivoice.com.br, na seção de suporte, tanto em VB como em Delphi.

Ao chamar o método PromptStart a reprodução da mensagem é iniciada e o fluxo de execução segue normalmente, da maneira análoga ao método PlayFile, por exemplo. Quando as condições forem cumpridas de acordo com o configurado, o evento OnPrompt será acionado e receberá a variável Status contendo o que aconteceu na execução da função e o parâmetro Value contendo o dado digitado pelo usuário. Os detalhes do significado e funcionalidade de cada parâmetro deve ser consultado no Guia de Referência deste manual.

Copyright © 2010 Digivoice Eletrônica

Funções de Menu

As funções de menu, disponíveis somente no ActiveX, automatizam o processo de atendimento com opções de menu. Ex.:

"...Para vendas disque 3, expedição 4 ..."
".... Para saldo disque 1, para falar com o atendente 3..."

O método MenuErrorSettings configura as respostas às situações de erro: frase de erro ("...Opção Inválida!..."), número de tentativas em caso de erro e frase para quando o usuário não discar nada.

O método MenuStart inicia a reprodução da frase de menu sendo passado como parâmetro a frase de opções, os dígitos válidos e o tempo máximo para digitação após a frase. Também deve ser informado se o tipo da discagem (pulso ou tom) e se será monitorada.

Assim como o prompt, após a chamada do método MenuStart, o fluxo de execução segue normalmente. Após o usuário interagir com o menu, o evento OnMenu será chamado, recebendo o Status e e a opção selecionada (OptionSelected).

Para interromper a execução do menu basta chamar o método MenuAbort. Os detalhes do significado e funcionalidade de cada parâmetro deve ser consultado no Guia de Referência deste manual. Exemplos dessas funções são fornecidos no site da DigiVoice www.digivoice.com.br, na seção de suporte, tanto em VB quanto em Delphi.

Copyright © 2010 Digivoice Eletrônica

Funções de Discagem e Transferência

Os métodos e eventos que compõem esta funcionalidade automatizam todo o processo de discagem, transferência, supervisão e atendimento através da chamada de um único método (MakeCall).

Para acompanhar a explicação, observe atentamente o exemplo de Transferência encontrado no site da DigiVoice www.digivoice.com.br na seção de suporte, tanto em VB quanto em Delphi.

Com este grupo de métodos será possível efetuar de maneira simples:

- Discagem externa com supervisão de atendimento, verificando se o destino atendeu ou não ou ainda se ocorreu ocupado.
- Transferência entre ramais de um pabx com ou sem supervisão.
- Frases automáticas em caso de ocupado e não atender.

Toda a configuração é feita através dos métodos SetCall*xxxx*.

Você pode programar um flash no início da discagem através do método SetCallStartFlash, útil em casos de transferência. Em termos de flash também é possível programar um flash específico para retomada em caso de ocupado (SetCallBusyReturnFlash) e outro em caso de não atendimento (SetCallNoAnswerReturnFlash). Em todos os casos é possível definir dígitos e pausa após o flash e também a pausa após dígito que são suficientes para adaptação em qualquer central PABX. O tempo de flash é uma configuração geral para todos os casos e é definida pelo método SetCallFlashTime.

É possível definir frases a serem reproduzidas quando se detecta o ocupado (SetCallBusyPhrase) ou quando não atende (SetCallNoAnswerPhrase). O sistema utiliza o método SetCallNoAnswerRingCount para determinar com quantos toques será considerado um "não atendimento".

Copyright © 2010 Digivoice Eletrônica

Em caso de atendimento é possível programar uma pausa e uma frase a ser reproduzida para o usuário. Isto é feito pelo método SetCallAfterAnswer.

O início da discagem é feito através da chamada ao método MakeCall, onde são definidos o tipo de discagem (externa ou com flash), a frase a ser reproduzida inicialmente, o número que será discado e se a discagem será feita com ou sem supervisão. Se for feita sem supervisão, tão logo o número é discado a execução do método é finalizada e o evento OnAfterMakeCall é chamado.

O tipo de discagem define uma série de comportamentos importantes:

Externa – Inicia a discagem com um pickup, pois entende que o telefone está desligado. Utiliza as definições de dígito e pausa após o pickup definidos pelo método SetCallAfterPickUp.
Com Flash – Executa um flash antes de iniciar a discagem, pois entende que a linha está conectada a um PABX e o que vai ser feito é uma transferência entre ramais.

Além das configurações anteriores, é possível determinar se o tom de discagem será esperado obrigatoriamente (SetCallWaitForDialTone) e o tempo que a supervisão será iniciada após a discagem (SetCallPauseBeforeAnalysis). Este último é útil em supervisões de transferência para casos onde o PABX gera ruídos durante o flash que podem induzir a placa a deteções erradas de atendimento. Definindo um tempo com este método a supervisão só iniciará a 'n' milisegundos após a discagem, o que permitirá ignorar estes ruídos.

Assim como todos os outros métodos da VoicerLib, o MakeCall é assíncrono, ou seja, ao ser executado inicia o processo todo mas o fluxo de execução do programa continua.

IMPORTANTE: O MakeCall não pode ser chamado com o método IdleStart ativo. Caso isso ocorra, retornará erro código 1.

A qualquer momento é possível saber se um MakeCall está em curso através da monitoração do evento OnCallStateChange.

Copyright © 2010 Digivoice Eletrônica

Por fim, o evento OnAfterMakeCall é chamado informando o tipo de ocorrência detectada durante o processo (ocupado, atendimento, etc...) através dos seguintes valores assumidos na variável Status:

- mkOK - Discou com sucesso
- mkNoDialTone – Sem tom de discagem
- mkDelivered – Ligação entregue sem supervisão
- mkNoAnswer – Ligação não foi atendida
- mkBusy – Ligação ocupada
- mkAnswered - Ligação atendida
- mkAborted – Ligação cancelada pelo AbortCall
- mkDialToneAfterDial - Indica recebimento de tom de linha depois da discagem. Normalmente esta situação indica algum problema com o ambiente (PABX, linha, etc...)
- mkFaxDetected - Indica detecção de fax

(*) No caso de receber um mkNoDialTone depois de efetuar o flash para transferência no PABX, você deve retomar a ligação antes de desligá-la para evitar de prender a linha no PABX. Uma situação de não receber tom de discagem para transferência entre ramais só pode ocorrer se as configurações de flash estiverem erradas ou no caso de sobrecarga de processamento do PABX (ocorre principalmente em discadores automáticos).

Copyright © 2010 Digivoice Eletrônica

Reproduzindo Data

A VoicerLib permite ao programador implementar sistemas que reproduzam datas através do método PlayDate. Veja o exemplo:

```
voicerlib.PlayDate Porta, Date, "d/m/y", "#", 0
```

O primeiro parâmetro faz referência a porta da placa. O segundo parâmetro é uma string contendo a data a ser reproduzida (é possível utilizar uma variável aqui ou mesmo no método Date, que retorna a data corrente). .

O terceiro parâmetro é a máscara de formatação da data. Esta máscara determina como a data será falada. Pode assumir os seguintes valores:

d/m/y – Ex.: "25 de setembro de 2001"
d/m – Ex.: "25 de setembro"
m/d – Ex.: "Setembro 25"
m/d/y – Ex.: "Setembro 25 2001"

O quarto parâmetro é o TermDigits, que indica quais dígitos podem interromper a mensagem.

O último parâmetro é uma pausa a ser respeitada antes da reprodução da mensagem, expressa em milisegundos, ou seja, caso queira esperar 1 segundo utilize o valor 1000.

Mesmo utilizando a pausa, o método possui retorno imediato após sua execução. Qualquer tratamento de seu resultado deverá ser feito no evento OnPlayStop ou OnDigitsReceived.

As frases-padrão utilizadas para reproduzir esta mensagem encontram-se na pasta \%PROGRAM FILES%\VoicerLib4\StockSigs. Os arquivos contidos nesta pasta deverão ser distribuídos junto da aplicação desenvolvida.

Copyright © 2010 Digivoice Eletrônica

Reproduzindo Números Cardinais

A VoicerLib permite ao programador implementar sistemas que reproduzam números inteiros ou fracionários por extenso (ex.:Dez vírgula trinta e cinco) através do método PlayCardinal.

Veja o exemplo:

```
voicerlib.PlayCardinal  Porta, "10,35", "#", 0
```

O primeiro parâmetro refere-se a porta da placa. O segundo parâmetro é uma string contendo o número a ser reproduzido (é possível utilizar uma variável aqui, é claro).

É necessário passar o valor formatado de acordo com o mostrado anteriormente, sem pontos separadores nos milhares e com vírgula como separador decimal (nnnnnnn,nn). Qualquer coisa diferente disso fará com que o método não reproduza o valor corretamente.

O terceiro parâmetro é o TermDigits, que indica quais dígitos podem interromper a mensagem.

O último parâmetro é uma pausa a ser respeitada antes da reprodução da mensagem, expressa em milisegundos, ou seja, caso queira esperar 1 segundo utilize o valor 1000.

Mesmo utilizando a pausa, o método possui retorno imediato após sua execução. Qualquer tratamento de seu resultado deverá ser feito no evento OnPlayStop ou OnDigitsReceived.

As frases-padrão utilizadas para reproduzir esta mensagem encontram-se na pasta \%PROGRAM FILES%\VoicerLib4\StockSigs. Os arquivos contidos nesta pasta deverão ser distribuídos junto da aplicação desenvolvida.

Copyright © 2010 Digivoice Eletrônica

Reproduzindo Números Dígito a Dígito

A VoicerLib permite ao programador implementar sistemas que reproduzam números através do método PlayNumber. Com isso é possível "soletrar" os dígitos de um dado qualquer (conta, cartão, etc...). Também permite falar "/" barra, "-" traço, "." ponto, "," – vírgula.

Veja o exemplo:

```
voicerlib.PlayNumber, "456790-23", "#", 0
```

O primeiro parâmetro refere-se a porta da placa. O segundo parâmetro é uma string contendo o número a ser reproduzido (é possível utilizar uma variável aqui, é claro).

O terceiro parâmetro é o TermDigits, que indica quais dígitos podem interromper a mensagem.

O último parâmetro é uma pausa a ser respeitada antes da reprodução da mensagem, expressa em milisegundos, ou seja, caso queira esperar 1 segundo utilize o valor 1000.

Mesmo utilizando a pausa, o método possui retorno imediato após sua execução. Qualquer tratamento de seu resultado deverá ser feito no evento OnPlayStop ou OnDigitsReceived.

As frases-padrão utilizadas para reproduzir esta mensagem encontram-se na pasta \%PROGRAM FILES%\VoicerLib4\StockSigs. Os arquivos contidos nesta pasta deverão ser distribuídos junto da aplicação desenvolvida.

Copyright © 2010 Digivoice Eletrônica

Reproduzindo Valores por Extenso

A VoicerLib permite ao programador implementar sistemas que falem valores monetários por extenso (ex.: Um mil e quinhentos reais e trinta e dois centavos) através do método PlayCurrency.

Veja o exemplo:

```
voicerlib.PlayCurrency  Porta, "1234,33", "#", 0
```

O primeiro parâmetro refere-se a porta da placa. O segundo parâmetro é uma string contendo o valor a ser reproduzido (é possível utilizar uma variável aqui, é claro).

É necessário passar o valor formatado de acordo com o mostrado anteriormente, sem pontos separadores nos milhares e com vírgula como separador décimal (nnnnnnn,nn). Qualquer coisa diferente disso fará com que o método não reproduza o valor corretamente.

O terceiro parâmetro é o TermDigits, que indica quais dígitos podem interromper a mensagem.

O último parâmetro é uma pausa a ser respeitada antes da reprodução da mensagem, expressa em milisegundos, ou seja, caso queira esperar 1 segundo utilize o valor 1000.

Mesmo utilizando a pausa, o método possui retorno imediato após sua execução. Qualquer tratamento de seu resultado deverá ser feito no evento OnPlayStop ou OnDigitsReceived.

Copyright © 2010 Digivoice Eletrônica

Reproduzindo Lista de Mensagens

Uma das mais importantes características da VoicerLib é a possibilidade de montar uma lista de mensagens que serão reproduzidas como se fossem uma só, gerando apenas um evento OnPlayStart no início e um OnPlayStop no final de todas.

Com isso o programador poderá montar frases ou sequências de mensagens de maneira muito simples.

A lista de mensagens permite que sejam inseridas mensagens baseadas em arquivos, data, hora, valor ou número cardinal.

Vamos usar como exemplo a seguinte frase: "Olá, hoje é dia 25 de setembro de 2010 às 12:32:45". Temos 4 partes distintas nesta frase:

1º - "Bom Dia, hoje é dia " – Arquivo pré-gravado do usuário. Verificar formato correto de gravação.
2º - "25 de setembro de 2020" – Data do Sistema (PlayDate)
3º - "às" – Arquivo pré-gravado do usuário
4º - "12:32:45" – Hora do Sistema (PlayTime)

Adicionando Mensagens a Lista

Para adicionar uma ou mais mensagens a uma lista, utilize o método PlayListAdd que tem o seguinte formato:

```
PlayListAdd Porta, Tipo, String, Máscara, PausaAntes
```

O parâmetro Porta é referente a porta a ser utilizada. A VoicerLib permite manter uma lista independente por porta.

O parâmetro Tipo indica que tipo de mensagem será reproduzida. Pode assumir os seguintes valores (representados por constantes):

- ptCardinal – Reproduzir um número cardinal (Semelhante ao método PlayCardinal)

Copyright © 2010 Digivoice Eletrônica

- ptFile – Reproduzir um arquivo de áudio nos formatos disponíveis na voicerlib (Semelhante ao método PlayFile)
- ptDate – Reproduzir uma data (Semelhante ao método PlayDate)
- ptTime – Reproduzir hora (Semelhante ao método PlayTime)
- ptCurrency – Reproduzir um valor monetário (Semelhante ao método PlayCurrency)
- ptNumber – Reproduz números dígito a dígito (Semelhante ao método PlayNumber)

O terceiro parâmetro (String) deve assumir a formatação exigida pelos tipos anteriores, ou seja se for indicado que é do tipo ptFile, este parâmetro deve receber uma string com o nome do arquivo a ser reproduzido. Para saber os formatos corretos para cada tipo, consulte os métodos independentes relacionados (PlayTime, PlayDate, etc...) no começo desta seção.

A Máscara (4º parâmetro) é relacionada apenas ao tipo ptDate e segue a mesma padronização indicada no método PlayDate explicado anteriormente.

O último parâmetro é uma pausa a ser respeitada antes da reprodução da mensagem, expressa em milisegundos, ou seja, caso queira esperar 1 segundo utilize o valor 1000. Cada mensagem da lista pode ter um valor diferente, ou seja, é possível dar pausas diferenciadas entre cada item da lista na hora da reprodução. Seguindo o exemplo de frase do começo desta seção, teríamos a seguinte codificação:

```
voicerlib.PlayListAdd Port, ptFile, "Principal.wav","",0
voicerlib.PlayListAdd Port, ptDate, Date, "d/m/y", 0
voicerlib.PlayListAdd Port, ptFile, "as.wav", "", 0
voicerlib.PlayListAdd Port, ptTime, Time, "", 0
```

Apagando o conteúdo de uma lista

A lista de mensagens permanece com seu conteúdo mesmo depois da reprodução, portanto, é necessário apagar seu conteúdo antes de adicionar novos valores. Para isso deve ser utilizado o método PlayListClear passando como parâmetro a porta da placa:

Copyright © 2010 Digivoice Eletrônica

```
'Apaga a lista antes de usá-la
voicerlib.PlayListClear Port
voicerlib.PlayListAdd Port, ptFile,
"frase_boasvindas.wav","",0
(…)
```

Verificando o tamanho da lista

Para saber quantos elementos existem dentro da lista, utilize o método PlayListGetCount, onde o único parâmetro é a porta e o valor de retorno é a quantidade de elementos.

```
Dim i as integer

  i = voicerlib.PlayListGetCount(Port)
```

Removendo um ítem específico da lista

Também é possível remover um determinado elemento da lista através do método PlayListRemoveItem. Este método tem dois parâmetros: a porta e o índice do item a ser excluído (0 até n-1).

```
'Remove o primeiro elemento da lista
voicerlib.PlayListRemoveItem(Port,0)
```

Reproduzindo a lista de mensagens

Após inserir os elementos da lista é possível reproduzi-la através do método PlayList, que tem o seguinte formato:

```
voicerlib.PlayList Port,TermDigits
```

O primeiro parâmetro indica a porta da placa. Lembre-se que cada porta tem uma lista independente.

O parâmetro TermDigits é uma string que permite configurar quais dígitos poderão interromper a mensagem, exatamente como funciona nos outros métodos Playxxx.

O método PlayList também é assíncrono, ou seja, ao executar o comando o fluxo de execução do aplicativo segue imediatamente.

Copyright © 2010 Digivoice Eletrônica

O final da reprodução deverá ser monitorado através do evento OnPlayStop.

O método PlayList, apesar de reproduzir várias mensagens encadeadas, funciona da mesma forma que o PlayFile, ou seja, gera apenas um OnPlayStart no início e um OnPlayStop no final do último item.

Caso o parâmetro TermDigits seja utilizado e a lista de mensagens seja interrrompida por dígito, toda a lista será interrompida e serão gerados os eventos OnPlayStop e OnDigitsReceived, sendo que este último receberá o valor edDigitOverMessage na variável Status.

Na dúvida, leia novamente a explicação sobre o método PlayFile. Tudo o que se refere a TermDigits aplica-se ao PlayList também.

Copyright © 2010 Digivoice Eletrônica

Distribuindo uma Aplicação

Visando facilitar a distribuição de todos os arquivos necessários para a execução de uma aplicação construída utilizando a VoicerLib, as placas adquiridas sempre são acompanhadas de um software chamado VoicerLib.

A melhor forma de distribuir uma aplicação desenvolvida com a Voicerlib, é fazendo uso da biblioteca que acompanha a placa, instalando-a na máquina de destino. O procedimento de instalação, tanto físico como de software é abordado pelo Manual do Kit Integrador (disponível no diretório de instalação ou no site da digivoice www.digivoice.com.br).

Com a VoicerLib instalada, não é necessário mais nenhum arquivo adicional para que uma aplicação funcione. Obviamente os requisitos de cada linguagem de programação deve ser atendidos pelo desenvolvedor (runtimes, dlls, etc...).

As versões mais atualizadas sempre estarão disponíveis no site da DigiVoice www.digivoice.com.br.

Copyright © 2010 Digivoice Eletrônica

Mensagens de Erro

Todos os métodos e funções retornam valores indicando sucesso ou problema na sua execução. Nos métodos e nas funções são indicados os nomes das constantes que representam estes códigos e abaixo são enumerados as constantes com seus respectivos valores, em hexadecima e decimal.

Copyright © 2010 Digivoice Eletrônica

Constante	Hexa	Decimal	Descrição
DG_EXIT_SUCCESS	0	0	Executado com sucesso
DG_ERROR_MEMORY_ALLOCATION	400h	1024	Falha de alocação de memória. Pode ser por falta de memória, mas também por erro no device driver, direitos de usuário (Linux), etc.
DG_ERROR_MAXCARDS	401h	1025	Falha na recuperação do número de placas instaladas
DG_ERROR_FIRMWARE_NOT_FOUND	402h	1026	Arquivo de firmware não foi encontrado. Pode ser causado por problemas de instalação ou no caso de uso com ActiveX a propriedade ConfigPath pode estar com valores errados
DG_ERROR_FIRMWARE_IO_TIMEOUT	403h	1027	Erro na carga do firmware. Pode ser causado por slot defeituoso. Contate a assistência técnica da Digivoice
DG_ERROR_READING_PORTCOUNT	404h	1028	Falha na leitura do número de portas por placa. Pode ser causado por falha de hardware ou erro durante a instalação
DG_ERROR_LOADING_DEVICEDRIVER	405h	1029	Erro na carga do device driver. As mensagens de debug podem esclarecer a causa
DG_ERROR_CREATING_EVENT	406h	1030	Erro na instalação de eventos de controle da VoicerLib. Só pode ser causado por falta de memória
DG_ERROR_DRIVER_CLOSED	450h	1104	O método foi chamado sem que a Voicerlib fosse iniciada

Copyright © 2010 Digivoice Eletrônica

Constante	Hexa	Decimal	Descrição
DG_ERROR_CARD_OUT_OF_RANGE	451h	1105	Parâmetro placa foi passado com valor maior do que o número de placas instaladas
DG_ERROR_PORT_OUT_OF_RANGE	452h	1106	Parâmetro porta passado com valor maior que o número total de portas instaladas (somadas todas as placas).
DG_ERROR_PARAM_OUTOFRANGE	453h	1107	Um dos parâmetros do método foi passado fora do intervalo especificado. Consulte a referência do método.
DG_ERROR_DRIVER_ALREADY_OPEN	454h	1108	Pode ocorrer se o StartVoicerLib for chamado duas vezes seguidas
DG_ERROR_CONFIGFILE_NOT_FOUND	455h	1109	O arquivo de configuração passado por parâmetro não foi encontrado. Não é necessário passar o caminho completo do arquivo.
DG_FEATURE_NOT_SUPPORTED	456h	1110	Este método não é suportado pela placa utilizada.
DG_ERROR_RESOURCE_UNAVAILABLE	457h	1111	Normalmente associado à métodos de chat, este erro ocorre quando existem "salas" disponíveis, porém não foi possível alocá-las por falta de memória
DG_ERROR_RESOURCE_FULL	458h	1112	Normalmente associada à métodos de chat, este erro ocorre quando não existem mais "salas" disponíveis.
DG_ERROR_INVALIDPARAM	459h	1113	Foi passado um parâmetro com valor inválido.

Copyright © 2010 Digivoice Eletrônica

Constante	Hexa	Decimal	Descrição
DG_ERROR_AUTO_SIGNALING_ON	45ah	1114	No protocolo ISDN PRI, a sinalização está no modo automático e este comando não aceita esta configuração
DG_ERROR_AUTO_SIGNALINF_OFF	45bh	1115	No protocolo ISDN PRI, a sinalização está no modo manual e este comando não aceita esta configuração
DG_ERROR_PORT_ALREADY_IN_USE	45ch	1116	No protocolo ISDN PRI, a porta já está sendo usada por outra chamada
DG_ERROR_PORT_IS_NOT_IN_USE	45dh	1117	No protocolo ISDN PRI, a porta precisa estar atendida
DG_WARNING_OLDFILEFORMAT	460h	1120	O arquivo de configuração está em um formato obsoleto
DG_ERROR_COULD_NOT_CREATE_THREAD	500h	1280	Falha na inicialização interna das threads de controle. Este erro normalmente é causado por falta de memória livre
DG_ERROR_THREAD_NOT_RUNNING	501h	1281	Este erro ocorre quando se tenta chamar um método de manipulação de threads de controle sem que o mesmo tenha sido iniciado (ex. chamada do EnableCallProgress sem ter chamado CreateCallProgress)
DG_ERROR_FIFO_UNAVAILABLE	502h	1282	Fifo de comunicação entre threads não está disponível. Falta de memória pode ser a causa mais comum

Copyright © 2010 Digivoice Eletrônica

Constante	Hexa	Decimal	Descrição
DG_ERROR_THREAD_ALREADY_RUNNING	503h	1283	Este erro é causado caso se chame um método de criação de Callprogress, ThreadE1, etc... mais de uma vez
DG_ERROR_THREAD_ENABLED	504h	1284	A thread ISDN PRI está habilitada. Para configurar é necessário desabilitar a thread ISDN PRI (DisableISDNThread), configurá-la e depois habilitá-la novamente (EnableISDNThread)
DG_ERROR_THREAD_DISABLED	505h	1285	A thread ISDN PRI está desabilitada e para este comando ela precisa estar habilitada
DG_ERROR_NOT_PLAYING	550h	1360	Tentativa de interromper a reprodução sendo que não há nenhuma em curso
DG_ERROR_ALREADY_PLAYING	551h	1361	Tentativa de iniciar uma reprodução quando já existe uma em curso
DG_ERROR_PLAY_OPENFILE	552h	1362	Não foi possível localizar ou abrir um arquivo para reprodução
DG_ERROR_PLAY_EMPTYLIST	553h	1363	Tentativa de reprodução de lista de mensagens através do PlayList, MenuStart ou PromptStart sem haver uma lista criada

Copyright © 2010 Digivoice Eletrônica

Constante	Hexa	Decimal	Descrição
DG_ERROR_AUDIO_FORMAT_UNSUPPORTED	554h	1364	Formato de áudio não suportado pela placa
DG_ERROR_PLAY_INVALID_FILENAME	555h	1365	Nome de arquivo inválido
DG_ERROR_PLAY_BUFFER_FULL	556h	1366	Buffer de reprodução via streaming cheio, significando que as amostras de áudio estão sendo enviadas em uma taxa maior do que a VoicerLib pode enviar ao hardware
DG_ERROR_REC_STOPPING	660h	1536	Foi chamado o comando StopRecordFile enquanto uma gravação estava sendo finalizada
DG_ERROR_REC_OPENFILE	661h	1537	Erro na criação do arquivo de gravação
DG_ERROR_REC_ALREADY_RECORDING	662h	1538	Já existe uma gravação em curso
DG_ERROR_REC_NOT_RECORDING	663h	1539	Chamada do StopRecordFile quando não há gravação em curso
DG_EXIT_FAILURE	700h	1792	Falha genérica, não especificada pelos códigos abaixo
DG_ERROR_CONV_SOURCE	701h	1793	Erro ao abrir o arquivo de origem da conversão
DG_ERROR_CONV_TARGET	702h	1794	Erro ao abrir o arquivo de destino da conversão
DG_ERROR_CONV	703h	1795	Erro indeterminado na conversão de arquivos

Copyright © 2010 Digivoice Eletrônica

Funções/Métodos

No Guia de Referência são apresentadas todas as funções disponiveis na API (DLL e Shared Object) como os métodos disponíveis no ActiveX para Windows.

Algumas funções estão disponíveis somente para o ActiveX e outras somente na API. Também há alguma diferenciação quanto à placa que suporta determinada função. Os ícones no início de cada tópico indicam este nível de compatibilidade.

Copyright © 2010 Digivoice Eletrônica

AbortCall

Cancela a discagem automática feita pelo método MakeCall.

Declarações:

ActiveX:
```
SHORT AbortCall(SHORT Port);
```

Descrição:

Durante o processo de discagem automática, iniciado pelo método MakeCall, é possível cancelar a discagem através da chamada deste método.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

AudioMonitor (dg_AudioMonitor)

Monitora em tempo real o áudio de uma porta específica na saída de áudio do PC.

Declarações:

ActiveX:
```
SHORT AudioMonitor(SHORT Port, SHORT Enable);
```

API:
```
short dg_AudioMonitor(short port, short enable);
```

Descrição:

Este método foi desenvolvido para permitir que se escute um canal de uma placa DigiVoice através do áudio do PC.
Só é possível a monitoração de um canal por vez. É necessário que a thread de InputBuffer (EnableInputBuffer) esteja ativada para este canal.

Parâmetros:

Port – Indica a porta da placa
Enable - Habilita (DG_ENABLE) ou desabilita (DG_DISABLE) a monitoração.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro fora do intervalo permitido (Enable)
DG_ERROR_RESOURCE_UNAVAILABLE - Erro ao criar ou destruir

Copyright © 2010 Digivoice Eletrônica

os controles de áudio, verique a porta.

CancelGetDigits (dg_CancelGetDigits)

Cancela a detecção de dígitos iniciada pelo GetDigits.

Declarações:

ActiveX:
```
SHORT CancelGetDigits(SHORT Port);
```

API:
```
short dg_CancelGetDigits(short port);
```

Descrição:

Quando o processo de recepção de dígitos acionado pelo método GetDigits é iniciado, a VoicerLib entra em um tratamento interno que pode durar o tempo dos timeouts definidos no GetDigits. É conveniente, caso a ligação seja encerrada em uma determinada porta enquanto o GetDigits esteja sendo executado, cancelar esta recepção chamando o CancelGetDigits.

Parâmetros:

Port – Indica a porta da placa no qual será cancelado o GetDigits

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instalados

Copyright © 2010 Digivoice Eletrônica

ChatAddPort (dg_ChatAddPort)

Adiciona porta a determinada sala de conferência.

Declarações:

ActiveX:
```
SHORT ChatAddPort(LONG Handle, SHORT Port);
```

API:
```
short dg_ChatAddPort(long Handle, short Port);
```

Descrição:

Este método permite adicionar determinada porta à sala de conferência especificada por Handle. Para que a porta possa falar e ouvir na conferência é necessário também habilitá-la usando ChatEnablePort.

Parâmetros:

Handle - Obtido como retorno do método CreateChatRoom
Port - Porta física da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_RESOURCE_FULL - Sala cheia
DG_ERROR_RESOURCE_UNAVAILABLE - Sala não disponível ou não criada

Copyright © 2010 Digivoice Eletrônica

ChatDisablePort (dg_ChatDisablePort)

Desabilita porta alocada em um recurso de conferência.

Declarações:

ActiveX:
```
SHORT ChatDisablePort(LONG Handle, SHORT Port);
```

API:
```
short dg_ChatDisablePort(long Handle, short Port);
```

Descrição:

Este método desabilita a porta da sala mas sem removê-la. Isto é particularmente útil para que determinada porta possa interagir com o sistema (por exemplo, ouvindo um menu) sem que as outras portas ouçam. Ao desabilitar uma porta, ela é reconectada no recurso de DSP.

Parâmetros:

Handle - Obtido como retorno do método CreateChatRoom
Port - Porta física da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_RESOURCE_UNAVAILABLE - Sala não disponível ou não criada

Copyright © 2010 Digivoice Eletrônica

ChatEnablePort (dg_ChatEnablePort)

Habilita porta alocada em um recurso de conferência a conversar com as outras.

Declarações:

ActiveX:
```
SHORT ChatEnablePort(LONG Handle, SHORT Port, SHORT
Direction);
```

API:
```
short dg_ChatEnablePort(long Handle, short Port,
short Direction);
```

Descrição:

Mesmo que a porta esteja alocada em uma determinada conferência é necessário que ela esteja habilitada para que possa se comunicar com as outras portas da mesma sala. O parâmetro Direction define como esta porta se comportará, podendo só ouvir, só falar ou falar e ouvir na sala.

Parâmetros:

Handle - Obtido como retorno do método CreateChatRoom
Port - Porta física da placa
Direction - Define o modo de comportamento da porta na sala
- CHAT_TALK - A porta só *falará* na sala
- CHAT_LISTEN - A porta só *escutará* na sala
- CHAT_TALK_LISTEN - A porta poderá *falar* e *ouvir* na sala

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro fora do intervalo

Copyright © 2010 Digivoice Eletrônica

permitido (Direction)
DG_ERROR_RESOURCE_UNAVAILABLE - Sala não disponível ou não criada

Copyright © 2010 Digivoice Eletrônica

ChatRemovePort (dg_ChatRemovePort)

Remove porta de determinada conferência.

Declarações:

ActiveX:
```
SHORT ChatRemovePort(LONG Handle, SHORT Port);
```

API:
```
short dg_ChatRemovePort(long Handle, short Port);
```

Descrição:

Este método desabilita e remove uma porta, de determinada sala de conferência.

Parâmetros:

Handle - Obtido como retorno do método CreateChatRoom
Port - Porta física da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_RESOURCE_UNAVAILABLE - Sala não disponível ou não criada

Copyright © 2010 Digivoice Eletrônica

CheckCode (dg_CheckCode)

Compara a string de segurança da memória da placa com a string a ser gravada.

Declarações:

ActiveX:
```
SHORT CheckCode(SHORT wCard, SHORT *wData);
```

API:
```
SHORT dg_CheckCode(short wCard, short *wData);
```

Parâmetros:

wCard - A placa que foi gravada
wData - Dado gravado

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro placa foi passado com valor maior do que o número de placas instaladas.
DG_EXIT_FAILURE - A string de segurança a ser gravada, difere da string atual da memória da placa.

Veja também: WriteCode

Copyright © 2010 Digivoice Eletrônica

ClearDigits (dg_ClearDigits)

Apaga o conteúdo do buffer de dígitos de cada porta.

Declarações:

ActiveX:
```
SHORT ClearDigits(SHORT Port);
```

API:
```
short dg_ClearDigits(short port);
```

Descrição:

É utilizado para apagar o buffer interno de dígitos detectados.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

ConnectAudioChannels (dg_ConnectAudioChannels)

Efetua uma conexão bi-direcional entre duas portas de qualquer placa.

Declarações:

ActiveX:
```
SHORT ConnectAudioChannels(SHORT Port1, SHORT
Port2);
```

API:
```
short dg_ConnectAudioChannels(short port1,short
port2);
```

Descrição:

Ao chamar este método, o áudio da porta Port1 é conectado em Port2 e vice-versa, permitindo que o áudio seja enviado nos dois sentidos. As portas podem pertencer à qualquer placa instalada.

Ao chamar este método, os inputBuffer da port1 e da port2 serão habilitados automaticamente.

Obs. Em caso de gravação deverão ser gerados dois arquivos de gravação (Port1 e Port2) utilizando o método RecordFile, que deverão ser "mixados" com um editor de áudio após o término das mesmas.

Parâmetros:

Port1 – Indica a primeira porta
Port2 – Indica a segunda porta

Valor de Retorno:

Copyright © 2010 Digivoice Eletrônica

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

ConfigCallProgress (dg_ConfigCallProgress)

Configura as opções de tratamento da thread para supervisão de linha.

Declarações:

ActiveX:
```
SHORT ConfigCallProgress(SHORT Port, SHORT Cmd, LONG
Value);
```

API:
```
short dg_ConfigCallProgress(short port, short
command, int value);
```

Descrição:

Se na chamada ao CreateCallProgress não for indicado um arquivo de configuração, o método ConfigCallProgress poderá ser chamado para modificar qualquer um das configurações de controle de supervisão de linha. Na tabela a seguir é mostrado os valores de Command e os limites de Value.

A VoicerLib já fornece um arquivo de configuração chamado cp_default.cfg contendo as configurações mais comuns. A utilização do ConfigCallProgress é indicada quando o desenvolvedor preferir manter seu próprio arquivo de configurações.

Copyright © 2010 Digivoice Eletrônica

Configurações de Tempos de detecção

Command	Descrição	Valor padrão
CPCFG_ANSWER_SENSITIVITY	AnswerSensitivity refere-se a quantos "áudios" deverão ser recebidos na sequência para considerar atendimento	1
CPCFG_ANSWER_SENSITIVITY_TIME	Tempo mínimo para que um sinal de áudio seja considerado atendimento.	200
CPCFG_GENERICTONETIMEOUT	GenericToneTimeout determina o tempo máximo que o sistema esperará para reconhecer o tom genérico	15000
CPCFG_GENERICTONETIME	GenericToneTime determina o tempo *mínimo* para que o sistema reconheça o áudio como tom genérico	500
CPCFG_LINETONETIMEOUT	LineToneTimeout determina o tempo *máximo* que o sistema esperará para reconhecer o tom de linha	15000
CPCFG_LINETONETIME	LineToneTime determina o tempo *mínimo* para que o sistema reconheça o áudio como tom de linha	1000
CPCFG_FAXTONETIMEOUT	FaxToneTimeout determina o tempo *máximo* que o sistema esperará para reconhecer o tom de fax	15000
CPCFG_FAXTONETIME	FaxToneTime determina o tempo *mínimo* para que o sistema reconheça o áudio como tom de linha	500
CPCFG_CALLPROGRESSTIMEOUT	CallProgressTimeout é o tempo de espera após a discagem para se considerar que houve um atendimento por timeout	1500

Copyright © 2010 Digivoice Eletrônica

CPCFG_BUSYSENSIBILITY	determina quantos tons de ocupado devem ser detectados para que seja gerado um evento para a aplicação. Este parâmetro é aplicado somente no CP_ENABLE_BUSY_OR_FAX	1
CPCFG_BUSYMINTIME	BusyMinTime determina o tempo mínimo do tom de ocupado	100
CPCFG_BUSYMAXTIME	BusyMaxTime determina o tempo máximo do tom de ocupado	500
CPCFG_CALLINGMINTONETIME	Determina o tempo mínimo para se considerar um tom de chamada (após a discagem). Será testado como intervalo junto com CallingMaxToneTime	600
CPCFG_CALLINGMAXTONETIME	Determina o tempo máximo para se considerar um tom de chamada (após a discagem)	2000
CPCFG_CALLINGMINSILTIME	O tempo mínimo para se considerar silêncio. Será testado como intervalo junto com CallingMaxSilTime	700
CPCFG_CALLINGMAXSILTIME	Determina o tempo máximo para se considerar silêncio	5000
CPCFG_TONEINTERRUPTIONMINTIME	Determina o tempo mínimo do intervalo entre cada tom de linha. O detecção do tom de chamando depende também dos tempos do intervalo entre eles.	20
CPCFG_TONEINTERRUPTIONMAXTIME	Determina o tempo mínimo do intervalo entre cada tom de linha. O detecção do tom de chamando depende também dos tempos do intervalo entre eles.	380
CPCFG_LINETONEMINTIME	Determina o tempo mínimo para se considerar o tom de linha	2500
CPCFG_LINETONEMAXTIME	Determina o tempo máximo para se considerar o tom de linha	2700

Copyright © 2010 Digivoice Eletrônica

Configuração das frequências

Command	Descrição	Valor padrão
CPCFG_LINEFREQ	Índice da frequência do tom de linha	22h
CPCFG_CALLINGFREQ	Índice da frequência	22h
CPCFG_BUSYFREQ	Índice da frequência	22h
CPCFG_GENERICFREQ	Índice da frequência	25h
CPCFG_GENERICFREQ2	Índice da frequência	26h
CPCFG_GENERICFREQ3	Índice da frequência	27h
CPCFG_GENERICFREQ4	Índice da frequência	28h
CPCFG_GENERICFREQ5	Índice da frequência	29h
CPCFG_SILENCE	Índice da frequência	20h
CPCFG_AUDIO	Índice da frequência	21h
CPCFG_FAX1	Índice da frequência	23h
CPCFG_FAX2	Índice da frequência	24h

Na tabela de configurações de frequências não deve ser informado o valor da frequência e sim o valor do índice utilizado pela VoicerLib. Verifique no tópico Supervisão de Linha no Guia do Programador.

Parâmetros:

Port – Indica a porta da placa que será iniciado o controle
Command - Indica qual a informação será configurada de acordo com as constantes da tabela mostrada anteriomente
Value - Valor do dado indicado por Command

Valores de Retorno:

DG_EXIT_SUCCESS - Thread iniciada com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa
DG_ERROR_THREAD_NOT_RUNNING - A thread não foi inicializada ainda
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Command ou Value fora do especificado

Copyright © 2010 Digivoice Eletrônica

ConfigCustomCAS (dg_ConfigCustomCAS)

Configura as opções de tratamento da thread CAS

Declarações:

ActiveX:

```
SHORT ConfigCustomCAS(SHORT Port, SHORT Param, SHORT
ParamValue);
```

API:

```
short dg_ConfigCustomCAS(short port, short command,
int value);
```

Descrição:

Sempre após chamar o CreateCustomCAS sem utilizar um arquivo de configuração, é necessário configurar algumas informações que indicará o comportamento da sinalização R2D em relação ao ramal CAS. Na tabela abaixo é mostrado os valores de Command e os limites de Value.

Copyright © 2010 Digivoice Eletrônica

Command	Descrição
CASCFG_RING	Sinalização de RING
CASCFG_CALLER_HANGUP	Sinalização indicando que B desligou
CASCFG_IDLE	Sinalização indicando porta livre
CASCFG_ANSWER	Sinalização indicando que o equipamento atendeu uma chamada
CASCFG_PICKUP	Indicação que o fone foi "retirado do gancho" para discar
CASCFG_DROP_DELAY_BEFORE	Tempo de intervalo entre um PickUp e um Hangup
CASCFG_DROP	Comando para desligar
CASCFG_FLASH1_CMD	Comando de primeiro flash
CASCFG_FLASH1_DELAY	Pausa após o primeiro flash
CASCFG_FLASH1_CMD	Comando do segundo flash
CASCFG_FLASH2_DELAY	Pausa após o segundo flash

Parâmetros:

Port – Porta que será iniciado o controle
Command - Indica qual a informação será configurada de acordo com as constantes da tabela mostrada anteriormente
Value - Valor do dado indicado por Command

Valores de Retorno:

DG_EXIT_SUCCESS - Thread iniciada com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa
DG_ERROR_THREAD_NOT_RUNNING - A thread E1 não foi

Copyright © 2010 Digivoice Eletrônica

inicializada ainda
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Command ou Value fora do especificado

Copyright © 2010 Digivoice Eletrônica

ConfigE1Thread (dg_ConfigE1Thread)

Configura as opções de tratamento da thread em relação ao protocolo R2.

Declarações:

ActiveX:

```
SHORT ConfigE1Thread(SHORT Port, SHORT Cmd, LONG
Value);
```

API:

```
short dg_ConfigE1Thread(short port, short command,
int value);
```

Descrição:

Sempre após chamar o CreateE1Thread, é necessário configurar algumas informações que indicará o comportamento da sinalização R2D durante a troca de sinalização. Na tabela a seguir é mostrado os valores de Command e os limites de Value.

Copyright © 2010 Digivoice Eletrônica

Command	Descrição	Value
E1CFG_MAXDIGITS_RX	Número total de dígitos a receber. Zero indica que a thread R2 não fará tratamento automático do recebimento de dígitos.	> 0
E1CFG_SEND_ID_AFTER DIGIT	Configura a partir de qual dígito será solicitada a identificação. Zero indica que não enviará identificação	>= 0
E1CFG_GROUP_B	Informa o grupo B	>= 0
E1CFG_GROUP_II	Informa o grupo II - Categoria do Assinante	>= 0
SILENCE_THRESHOLD	Informa o limiar de silêncio durante a sinalização	-30
SILENCE_THRESHOLD_A FTER	Informa o limiar de silêncio a ser usado depois do atendimento	-28
E1CFG_SEIZURE_TIMEO UT	Timeout de ocupação	7000ms
E1CFG_RETENTION_TIM EOUT	Timeout de retenção	9000ms
E1CFG_ANSWER_TIMEO UT	Timeout de atendimento	75000ms
E1CFG_BLOCKING_ON	Tempo de porta atendida para bloqueio	1000ms
E1CFG_BLOCKING_OFF	Tempo de porta desligada para bloqueio	2000ms
E1CFG_MFT_TIMEOUT	Timeout de MF para Trás	30000ms
E1CFG_MFF_TIMEOUT	Timeout de MF para Frente	30000ms
E1CFG_GENERATE_RING BACK	0 - Não gera tom de controle de chamada 1 - Gera tom de controle de chamada (default)	0 ou 1
E1CFG_R2_COUNTRY	br(brasil - default), ar(argentina) e mx(méxico)	1 à 3
E1CFG_SIGTYPE	R2_TYPE_MFC = 1 (default), R2_TYPE_CB_FXO = 2, R2_TYPE_CB_FXS = 3 Tipo de sinalização da thread E1. Obs. R2_TYPE_CB_XXX diz respeito a sinalização para o equipamento Digivoice Channel Bank CB3000	1 à 3

Copyright © 2010 Digivoice Eletrônica

Os parâmetros de timeouts já estão configurados de acordo com a norma do protocolo R2D e só deverá ser alterada se for realmente necessário. Os quatro primeiros parâmetros da lista anterior deverão ser configurados sempre, de acordo com o ambiente de produção.

Parâmetros:

Port – Porta que será iniciado o controle
Command - Indica qual a informação será configurada de acordo com as constantes da tabela anterior
Value - Valor do dado indicado por Command

Valores de Retorno:

DG_EXIT_SUCCESS - Thread iniciada com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa
DG_ERROR_THREAD_NOT_RUNNING - A thread E1 não foi inicializada ainda
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Command ou Value fora do especificado

Veja Também: CreateE1Thread, DestroyE1Thread

Copyright © 2010 Digivoice Eletrônica

ConfigGSMThread (dg_ConfigGSMThread)

Configura a thread GSM.

Declarações:

ActiveX:
```
SHORT ConfigGSMThread(SHORT Port, SHORT command,
LONG Value);
```

API:
```
short dg_ConfigGSMThread(short port, short command,
int value);
```

Descrição:

Não é necessário iniciar a thread GSM para configurá-la, ou seja, os valores configurados já seram assumidos ao criar a thread.

Porém se a thread estiver iniciada, e seja necessário configura-la, é necessário desabilitar a thread GSM com o método DisableGSMThread e após a configuração ter sido concluída, habilitar a thread (EnableGSMThread), dessa forma a thread será reiniciada com os valores configurados.

O evento OnGSMReady deverá ser aguardado antes do envio de qualquer outro comando para o módulo.

Parâmetros:

Port – Porta que será iniciado o controle
Command - Indica qual a informação será configurada
- GSMCFG_DIGIT_TIMEOUT - Configuração de timeout de respostas de mensagens
- GSMCFG_ID_RESTRICTION - Restringe o BINA (identificador de

Copyright © 2010 Digivoice Eletrônica

A) nas ligações de saída
- GSMCFG_CALL_WAITING_ENABLE - Habilita/Desabilita o recebimento da notificação de segunda chamada
- GSMCFG_RETRY_TIMEOUT - Tempo entre tentativas de inicialização dos módulos GSM em caso de erro
- GSMCFG_ANSWER_TIMEOUT - Tempo máximo após o envio de um comando para o recebimento de resposta
Value - Se o command for GSMCFG_DIGIT_TIMEOUT é o tempo para considerar timeout em milisegundos (padrão 5000 ms);
Se for GSMCFG_ID_RESTRICTION se o valor for 0 não restringe o BINA e se o valor for 1 restringe o BINA;
Se for GSMCFG_CALL_WAITING_ENABLE se o valor for 0 habilita o recebimento do aviso de segunda chamada e se o valor for 1 desabilita esse recurso;
Se for GSMCFG_RETRY_TIMEOUT é o tempo para considerar timeout em milisegundos em caso de erro;
Se for GSMCFG_ANSWER_TIMEOUT é o tempo em milisegundos para o recebimento de resposta após o envio de um comando.

Valores de Retorno:

DG_EXIT_SUCCESS - Thread iniciada com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa
DG_ERROR_THREAD_NOT_RUNNING - A thread GSM não foi inicializada ainda
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Command ou Value fora do especificado

Veja Também: CreateGSMThread, DestroyGSMThread.

Copyright © 2010 Digivoice Eletrônica

ConfigISDNThread (dg_ConfigISDNThread)

Configura as opções de tratamento da thread em relação ao protocolo ISDN PRI.

Declarações:

ActiveX:
```
SHORT ConfigISDNThread(SHORT Card, SHORT Framer,
SHORT Command, LONG Value);
```

API:
```
short dg_ConfigISDNThread(short card, short framer,
short command, int value);
```

Descrição:

Sempre antes de chamar o método CreateISDNThread, é necessário configurar algumas informações que indicará o comportamento da protocolo ISDN PRI durante a sinalização. Se for necessário configurar a thread depois que o método CreateISDNThread foi chamado, basta somente desabilitar a thread com o DisableISDNThread, fazer as configurações necessárias com o ConfigISDNThread e depois habilitar a thread com EnableISDNThread.

Parâmetros:

Card – Placa que será configurada
Framer - Span que será configurado
Command - Indica qual a informação será configurada de acordo com as constantes da tabela
Value - Valor do dado indicado por Command

Copyright © 2010 Digivoice Eletrônica

Command	Value
ISDN_SET_LOCALTYPE	ISDN_NET(0) ou ISDN_CPE(1) - padrão 1
ISDN_SET_SWITCHTYPE	ISDN_EUROISDN_E1 (5)
ISDN_SET_MIN_DIGITS	Número maior ou igual a 3 e menor ou igual a 22 - padrão 4
ISDN_SET_MAX_DIGITS	Número maior ou igual a 3 e menor ou igual a 22 - padrão 4
ISDN_AUTO_SIGNALING	DG_DISABLE (0) ou DG_ENABLE (1) - padrão 1
ISDN_SET_EXCLUSIVE_CHANNEL	DG_DISABLE (0) ou DG_ENABLE (1) - padrão 1
ISDN_SET_GENERATE_TONE	DG_DISABLE (0) ou DG_ENABLE (1) - padrão 0

Valores de Retorno:

DG_EXIT_SUCCESS - Configuração feita com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro placa foi passado com valor zero ou maior do que o número de placas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Framer fora do valor especificado
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1.
DG_ERROR_THREAD_ENABLED - A thread ISDN está habilitada. Para configurar é necessário desabilitar a thread ISDN (DisableISDNThread), configurá-la e depois habilitá-la novamente (EnableISDNThread)

Veja Também: CreateISDNThread, DestroyISDNThread, EnableISDNThread, DisableISDNThread.

Copyright © 2010 Digivoice Eletrônica

CreateCallProgress (dg_CreateCallProgress)

Inicia o tratamento automático de supervisão de linha

Declarações:

ActiveX:
```
SHORT CreateCallProgress(SHORT Port , LPCTSTR
ConfigFile);
```

API:
```
short  dg_CreateCallProgress(short port, char
*ConfigFile);
```

Descrição:

Este método cria a thread de controle de supervisão de linha (call progress), todas as configurações pertinentes à supervisão de linha são lidas do arquivo informado no parâmetro ConfigPath. Caso este arquivo não seja fornecido, as configurações também podem ser modificadas pelo método ConfigCallProgress. Os arquivos de configuração do callprogress ficam na mesma pasta do firmware, sendo que o arquivo padrão é o cp_default.cfg que deverá atender a maioria das instalações. Este arquivo contém diversos comentários, explicando cada um dos parâmetros configuráveis.

Este método deve ser chamado antes do call progress em si ser habilitado pelo método EnableCallProgress. Como é uma thread com baixo consumo de processamente quando está atuando, não há maiores problemas em criá-la no início da aplicação e só destruí-la no seu término.

Parâmetros:

Port – Porta que será iniciado o controle
ConfigFile- Arquivo de configuração padrão, sem o caminho.

Copyright © 2010 Digivoice Eletrônica

Valores de Retorno:

DG_EXIT_SUCCESS - Thread iniciada com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_THREAD_ALREADY_RUNNING - A thread já havia sido iniciada anteriormente
DG_ERROR_COULD_NOT_CREATE_THREAD - Falha na inicialização da thread
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_CONFIGFILE_NOT_FOUND - O arquivo de configuração passado por parâmetro não foi encontrado. Não é necessário passar o caminho completo do arquivo.
DG_WARNING_OLDFILEFORMAT - O arquivo de configuração está em um formato obsoleto

Copyright © 2010 Digivoice Eletrônica

CreateChatRoom (dg_CreateChatRoom)

Cria recurso para conferência ("cria sala de bate papo").

Declarações:

ActiveX:
```
LONG CreateChatRoom(SHORT Card, SHORT MaxPorts);
```

API:
```
long dg_CreateChatRoom(short card, short maxports);
```

Descrição:

A criação da conferência permite que um número de portas determinado em maxports conversem entre si. Ao criar esse recurso o valor de retorno é utilizado como um código identificador (handle) para cada sala de conferência.

A conferência é criada para determinada placa. As portas que participarão desta conferência serão determinadas pelo método ChatAddPort. Atualmente a voicerLib tem capacidade para 30 salas de conferência, mas os limites de portas por sala e capacidade de processamento das placas deverá ser considerado.

Parâmetros:

Card - Placa onde será criada a sala de conferência
MaxPorts – Máximo de canais permitido nesta sala de conferência

Valor de Retorno:

Caso o retorno seja sucesso, retornará um valor maior que zero que é o Handle válido a ser usado nas outras funções, em caso de erro retorna zero

Copyright © 2010 Digivoice Eletrônica

CreateCustomCAS (dg_CreateCustomCAS)

Inicia a thread de controle para ramal CAS (E1)

Declarações:

ActiveX:
```
SHORT CreateCustomCAS(SHORT Port , LPCTSTR
ConfigFile);
```

API:
```
short dg_CreateCustomCAS(short port, char
*ConfigFile);
```

Descrição:

Este método cria a thread de controle de ramal CAS (E1). Todas as configurações pertinentes são lidas do arquivo informado no parâmetro ConfigPath. Caso este arquivo não seja fornecido, as configurações também podem ser modificadas pelo método ConfigCustomCAS. Os arquivos de configuração ficam na mesma pasta do firmware. Este arquivo contém diversos comentários, explicando cada um dos parâmetros configuráveis.

O arquivo de configuração contém a ação para cada sinal R2D que o PABX enviar ao ramal CAS.

Parâmetros:

Port – Porta que será iniciado o controle
ConfigFile- Arquivo de configuração padrão, sem o caminho

Valores de Retorno:

DG_EXIT_SUCCESS - Thread iniciada com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado

Copyright © 2010 Digivoice Eletrônica

DG_ERROR_THREAD_ALREADY_RUNNING - A thread já havia sido iniciada anteriormente
DG_ERROR_COULD_NOT_CREATE_THREAD - Falha na inicialização da thread
DG_ERROR_CONFIGFILE_NOT_FOUND - Arquivo informado não foi encontrado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

CreateE1Thread (dg_CreateE1Thread)

Inicia o tratamento automático do protocolo R2D de troncos digitais

Declarações:

ActiveX:
```
SHORT CreateE1Thread(SHORT Port);
```

API:
```
short dg_CreateE1Thread(short port);
```

Descrição:

Antes de iniciar ou receber chamadas através do tronco E1 é necessário chamar este método, que cria uma thread de controle sinalização de linha R2D e de registro MFC 5C. Desta forma o programador não necessita conhecer este protocolo para efetuar ou receber ligações em cada porta.

Parâmetros:

Port – Porta que será iniciado o controle

Valores de Retorno:

DG_EXIT_SUCCESS - Thread iniciada com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_THREAD_ALREADY_RUNNING - A thread já havia sido iniciada anteriormente
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa
DG_ERROR_COULD_NOT_CREATE_THREAD - Falha na inicialização da thread
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do

Copyright © 2010 Digivoice Eletrônica

intervalo permitido, excedendo o número de portas instaladas

Veja Também: DestroyE1Thread, ConfigE1Thread

Copyright © 2010 Digivoice Eletrônica

CreateGSMThread (dg_CreateGSMThread)

Cria e inicializa os módulos da placa GSM, a thread GSM deverá ser criada para cada porta da placa.

Declarações:

ActiveX:
```
SHORT CreateGSMThread(SHORT Port);
```

API:
```
short dg_CreateThread(short port);
```

Descrição:

A criação das threads pode levar alguns instantes devido a conexão com a operadora de telefonia móvel e outras configurações, por isso foi criado um evento que sinaliza se a criação e inicialização das threads foram concluídas com sucesso ou não. O evento OnGSMReady deve ser aguardado antes do envio de qualquer outro comando para a porta inicializada.

Parâmetros:

Port – Porta que será iniciado o controle

Valores de Retorno:

DG_EXIT_SUCCESS - Thread iniciada com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_THREAD_ALREADY_RUNNING - A thread já havia sido iniciada anteriormente
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa

Copyright © 2010 Digivoice Eletrônica

DG_ERROR_COULD_NOT_CREATE_THREAD - Falha na inicialização da thread

Veja Também: DestroyGSMThread, ConfigGSMThread.

Copyright © 2010 Digivoice Eletrônica

CreateISDNThread (dg_CreateISDNThread)

Inicia o tratamento do protocolo ISDN PRI de troncos digitais.

Declarações:

ActiveX:
```
SHORT CreateISDNThread(SHORT Card, SHORT Framer);
```

API:
```
short dg_CreateISDNThread(short card, short framer);
```

Descrição:

Antes de iniciar ou receber chamadas através do tronco E1, que utiliza o protocolo ISDN PRI, é necessário chamar este método para criar uma thread de controle sinalização.
Obs. Lembrando que é necessário configurar a thread ISDN (ConfigISDNThread) antes de criá-la.

Parâmetros:

Card – Placa que será iniciado o controle
Framer - Span que será iniciado o controle

Valores de Retorno:

DG_EXIT_SUCCESS - Criou thread ISDN PRI com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro placa foi passado com valor zero ou maior do que o número de placas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Framer fora do valor especificado
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1.
DG_ERROR_THREAD_ALREADY_RUNNING - A thread ISDN já está

Copyright © 2010 Digivoice Eletrônica

iniciada
DG_ERROR_COULD_NOT_CREATE_THREAD - Falha na iniciação da thread ISDN

Veja Também: DestroyISDNThread, Inicialização do Protocolo ISDN PRI.

CreateLoggerControl (dg_CreateLoggerControl)

Inicia o tratamento automático de gravação em paralelo para sinalização R2 MFC.

Declarações:

ActiveX:
```
SHORT CreateLoggerControl(SHORT Port, SHORT Type);
```

API:
```
short dg_CreateLoggerControl(short port, short
type);
```

Descrição:

Este método simplifica a criação de aplicações de gravação em paralelo, fazendo todo o tratamento do protocolo indicado pelo parâmetro Type automaticamente e gerando os eventos indicativos de início e fim da conversação.

Esta thread de controle também cuida de todas as chamadas as funções de conferência entre canais (ChatAddPort, CreateChatRoom, etc...).

Parâmetros:

Copyright © 2010 Digivoice Eletrônica

Port – Porta que será iniciado o controle
Type - Tipo de protocolo utilizado. Valores possíveis:
LOGGER_RD2MF - Protocolo R2D (placas E1)

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa
DG_ERROR_THREAD_ALREADY_RUNNING - A thread já está criada
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa
DG_ERROR_COULD_NOT_CREATE_THREAD - Falha na inicialização da thread

Copyright © 2010 Digivoice Eletrônica

CreateLoggerCCS (dg_CreateLoggerCCS)

Inicia o tratamento automático de gravação em paralelo para a sinalização ISDN PRI.

Declarações:

ActiveX:
```
SHORT CreateLoggerCCS(SHORT Port, SHORT Type);
```

API:
```
short dg_CreateLoggerCCS(short port, short type);
```

Descrição:

Este método simplifica a criação de aplicações de gravação em paralelo, fazendo todo o tratamento do protocolo indicado pelo parâmetro Type automaticamente e gerando os eventos indicativos de início e fim da conversação.

Esta thread de controle também cuida de todas as chamadas as funções de conferência entre canais (ChatAddPort, CreateChatRoom, etc...).

Parâmetros:

Port – Porta que será iniciado o controle
Type - 0 - Desabilita arquivo de log
1 - Habilita arquivo de log
Obs. A pasta log deve estar criada se a opção for 1.

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do

Copyright © 2010 Digivoice Eletrônica

intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa
DG_ERROR_THREAD_ALREADY_RUNNING - A thread já está criada
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa
DG_ERROR_COULD_NOT_CREATE_THREAD - Falha na inicialização da thread

Copyright © 2010 Digivoice Eletrônica

DefinePortResource (dg_DefinePortResource)

Associa uma porta física à uma porta virtual

Declarações:

ActiveX:
```
SHORT DefinePortResource(SHORT Port, SHORT Card,
SHORT CardChannel);
```

API:
```
short dg_DefinePortResource(short port, short card,
short card_channel);
```

Descrição:

Este método permite associar uma porta física de uma determinada placa à um número de porta lógica. Para isso, o método DefinePortResource deve ser chamado passando como parâmetro a porta lógica, a placa e a porta física da placa.

Se por acaso já houver uma outra porta física associada à uma porta lógica e se tentar associar uma nova porta, a configuração anterior será sobreposta.

O valor de porta virtual é o que deverá ser utilizado a partir daí para se referenciar qualquer coisa daquela porta física.

Parâmetros:

Port – Indica o número da porta virtual
Card - Valor da placa (1 a n)
CardChannel - Valor da placa física da placa (depende do tipo de placa)

Copyright © 2010 Digivoice Eletrônica

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro card fora do intervalo de placas instaladas
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

DestroyCallProgress (dg_DestroyCallProgress)

Finaliza o tratamento automático de supervisão de linha

Declarações:

ActiveX:
```
SHORT DestroyCallProgress(SHORT Port);
```

API:
```
short dg_DestroyCallProgress(short port);
```

Descrição:

Este método deverá ser chamado para finalizar uma thread de tratamento de supervisão de linha, sempre quando não for mais necessária

Parâmetros:

Port – Porta que será iniciado o controle

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

DestroyChatRoom (dg_DestroyChatRoom)

Remove recurso de uma sala de conferência

Declarações:

ActiveX:
```
SHORT DestroyChatRoom(SHORT Card, LONG Handle);
```

API:
```
short dg_DestroyChatRoom(short card, unsigned int
Handle);
```

Descrição:

Utilize esse método para excluir uma conferência já existente, passando o identificado (Handle) que é código utilizado para identificar cada conferência.

Parâmetros:

Card - Placa onde será criada a sala de conferência
Handle - Obtido como retorno do método CreateChatRoom

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro card fora do intervalo de placas instaladas
DG_ERROR_RESOURCE_UNAVAILABLE - Recurso não disponível

Copyright © 2010 Digivoice Eletrônica

DestroyCustomCAS (dg_DestroyCustomCAS)

Finaliza o tratamento automático de ramal CAS

Declarações:

ActiveX:
```
SHORT DestroyCustomCAS(SHORT Port);
```

API:
```
short dg_DestroyCustomCAS(short port);
```

Descrição:

Este método deverá ser chamado para finalizar uma thread de tratamento de ramal CAS, sempre quando não for mais necessária

Parâmetros:

Port – Porta que será iniciado o controle

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

DestroyE1Thread (dgDestroyE1Thread)

Termina o tratamento automático do protocolo R2D de troncos digitais.

Declarações:

ActiveX:
```
SHORT DestroyE1Thread(SHORT Port);
```

API:
```
short dg_DestroyE1Thread(short port);
```

Descrição:

Quando a thread de controle R2 não for mais necessária, é preciso removê-la para evitar desperdício de memória.

Parâmetros:

Port – Porta que será iniciado o controle

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa

Veja Também: CreateE1Thread, ConfigE1Thread

Copyright © 2010 Digivoice Eletrônica

DestroyGSMThread (dg_DestroyGSMThread)

Destroi a thread GSM. Deve ser chamado antes do método ShutdwonVoicerlib.

Declarações:

ActiveX:
```
SHORT DestroyE1Thread(SHORT Port);
```

API:
```
short dg_DestroyE1Thread(short port);
```

Parâmetros:

Port – Porta que será iniciado o controle

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa

Veja Também: CreateGSMThread, ConfigGSMThread

Copyright © 2010 Digivoice Eletrônica

DestroyISDNThread (dg_DestroyISDNThread)

Termina o tratamento do protocolo ISDN PRI.

Declarações:

ActiveX:
```
SHORT DestroyISDNThread(SHORT Card, SHORT Framer);
```

API:
```
short dg_DestroyISDNThread(short card, short
framer);
```

Descrição:

Quando a thread de controle ISDN não for mais necessária, é preciso removê-la para evitar desperdício de memória.

Parâmetros:

Card – Placa que será finalizado o controle
Framer - Span que será finalizado o controle

Valores de Retorno:

DG_EXIT_SUCCESS - Destruiu thread ISDN PRI com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro placa foi passado com valor zero ou maior do que o número de placas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Framer fora do valor especificado
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1.

Copyright © 2010 Digivoice Eletrônica

Veja Também: CreateISDNThread.

DestroyLoggerControl (dg_DestroyLoggerControl)

Finaliza o tratamento automático de gravação em paralelo.

Declarações:

ActiveX:
```
SHORT DestroyLoggerControl(SHORT Port);
```

API:
```
short dg_DestroyLoggerControl(short port);
```

Descrição:

Este método deverá ser chamada para finalizar uma thread de tratamento iniciada através do método CreateLoggerControl.

Parâmetros:

Port – Porta que será iniciado o controle

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa

Copyright © 2010 Digivoice Eletrônica

DestroyLoggerCCS (dg_DestroyLoggerCCS)

Finaliza o tratamento automático de gravação em paralelo.

Declarações:

ActiveX:
```
SHORT DestroyLoggerCCS(SHORT Port);
```

API:
```
short dg_DestroyLoggerCCS(short port);
```

Descrição:

Este método deverá ser chamada para finalizar uma thread de tratamento iniciada através do método CreateLoggerCCS.

Parâmetros:

Port – Porta que será iniciado o controle

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa

Copyright © 2010 Digivoice Eletrônica

dg_SetEventCallback

API

Declarações:

API:

```
void dg_SetEventCallback(void (DCALLBACK *ptrFunc)
(void *), void *context_data);
```

Descrição:

Este método, exclusiva da API, indica à VoicerLib qual a função que fará o tratamento dos eventos da placa. Nela deve ser passado o ponteiro da função e a estrutura que guardará os dados dos eventos.

Para maiores detalhes e um exemplo de funcionamento, consulte o tópico Conceitos Básicos - API do Guia de Programação deste manual.

Parâmetros:

***ptrFunc**: Ponteiro da função que tratará os eventos
context_data: ponteiro para a estrutura que conterá os dados do evento

Copyright © 2010 Digivoice Eletrônica

Dial (dg_Dial)

Disca uma sequência de números ou pausa.

Declarações:

ActiveX:
```
SHORT Dial(SHORT Port, LPCTSTR Number, LONG
PauseAfterDial, SHORT DialType);
```

API:
```
short dg_Dial(short port, char *Number, long
pause_after_dial, short dialtype);
```

Descrição:

O método Dial permite enviar dígitos ou pausas para que seja efetuada a discagem. Pode ser enviado os dígitos de "0" a "9", "#", "*" e os símbolos de pausa "vírgula" "ponto-e-vírgula" e "ponto".

A pausa para cada símbolo deve ser atribuída através do método SetDialDelays.

O parâmetro DialType aceita apenas o valor dtTone(1) para discagens por tom.

Se o protocolo for R2D MFC, o dígito é enviado, sendo gerado um MFF ou MFT, conforme o andamento do protocolo.

Se o protocolo for ISDN PRI, os dígitos serão enviados apenas por sinalização.

Parâmetros:

Port – Indica a porta da placa
Number – String contendo os dígitos ou pausas para discagem.

Copyright © 2010 Digivoice Eletrônica

PauseAfterDial - É um tempo extra em milisegundos que é dado após a discagem. O evento OnAfterDial só será gerado após este tempo.
DialType - Pode assumir o valor dtTone(1), indicando se a discagem é por tom ou dtDirectMF(7) que gera o MF mesmo com a thread E1 habilitada

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro Port fora do intervalo permitido
DG_ERROR_PARAM_OUTOFRANGE - Parametro PauseAfterDial maior que 60000 ms
DG_FEATURE_NOT_SUPPORTED - Para o protocolo ISDN PRI, é necessário que a placa seja E1
DG_ERROR_COULD_NOT_CREATE_THREAD - Falha na iniciação da thread
DG_ERROR_PORT_ALREADY_IN_USE - Caso o protocolo seja ISDN, a porta esta ocupada (ISDN_SET_EXCLUSIVE_CHANNEL = DG_ENABLE) e/ou não tem outras portas disponíveis (ISDN_SET_EXCLUSIVE_CHANNEL = DG_DISABLE)
DG_ERROR_PORT_IS_NOT_IN_USE - Caso o protocolo seja ISDN, a porta não está atendida
DG_ERROR_FIFO_UNAVAILABLE - Fifo de comunicação entre threads não está disponível. Falta de memória pode ser a causa mais comum.
DG_ERROR_THREAD_DISABLED - A thread ISDN está desabilitada

Copyright © 2010 Digivoice Eletrônica

DisableAGC (dg_DisableAGC)

Desabilita o ajuste automático de ganho em um determinada porta

Declarações:

ActiveX:
```
SHORT DisableAGC(SHORT Port);
```

API:
```
short dg_DisableAgc(short port);
```

Descrição:

Este método desabilita o controle automático de ganho em uma gravação ou conferência.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

DisableAnswerDetection (dg_DisableAnswerDetection)

Desabilita a supervisão de atendimento

Declarações:

ActiveX:
```
SHORT DisableAnswerDetection(SHORT Port);
```

API:
```
short dg_DisableAnswerDetection(short port);
```

Descrição:

O método DisableAnswerDetection desabilita a supervisão de atendimento. A chamada deste método não tem efeito nas placas digitais pois, devido à característica do protocolo utilizado, a detecção sempre está habilitada.

Nas placas FXO, o atendimento é detectado por áudio ou por timeout. Logo ao primeiro evento OnAnswerDetected é preciso chamar este método pois, caso contrário, o evento continuará sendo gerado durante a conversação.

A VoicerLib 4 tem funções específicas para detecção de silêncio, não sendo mais necessário utilizar a detecção de atendimento para este fim.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

Copyright © 2010 Digivoice Eletrônica

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_THREAD_NOT_RUNNING - Thread de controle de callprogress não está em funcionamento
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

DisableAutoFramers (dg_DisableAutoFramers)

Desabilita os framers E1

Declarações:

ActiveX:
```
SHORT DisableAutoFramers(void);
```

API:
```
short dg_DisableAutoFramers(void);
```

Descrição:

Sempre quando for chamado o método de sincronismo SetCardSyncMode, a VoicerLib automaticamente reinicia os framers que fazem a comunicação E1.

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_ALREADY_OPEN - Driver já habilitado

Copyright © 2010 Digivoice Eletrônica

DisableCallProgress (dg_DisableCallProgress)

Desabilita a supervisão de linha, fax e ocupado.

Declarações:

ActiveX:
```
SHORT DisableCallProgress(SHORT Port);
```

API:
```
short dg_DisableCallProgress(short port);
```

Descrição:

O método DisableCallProgress desabilita a supervisão de linha sem finalizar a thread de controle de callprogress. A supervisão de linha permite monitorar o sinal de chamada, ocupado, fax e tom de discagem.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_THREAD_NOT_RUNNING - A thread de callprogress não foi criada (pelo método CreateCallProgress)
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

DisableDebug (dg_DisableDebug)

Desabilita o recurso de depuração da VoicerLib.

Declarações:

ActiveX:
```
SHORT DisableDebug(void);
```

API:
```
short dg_DisableDebug(void);
```

Descrição:

Este método desabilita o envio de informações de debug via TCP/IP

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso

Copyright © 2010 Digivoice Eletrônica

DisableDTMFFilter (dg_DisableDTMFFilter)

Desabilita filtro de DTMF e 425Hz

Declarações:

ActiveX:
```
SHORT DisableDTMFFilter(SHORT Port);
```

API:
```
short dg_DisableDTMFFilter(short port);
```

Descrição:

O método DisableDTMFFilter desliga as filtragens de DTMF de uma determinada porta do sistema.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

DisableE1Thread (dg_DisableE1Thread)

Desabilita o tratamento automático do protocolo R2D de troncos digitais.

Declarações:

ActiveX:
```
SHORT DisableE1Thread(SHORT Port);
```

API:
```
short dg_DisableE1Thread(short port);
```

Descrição:

O DisableE1Thread faz com que a thread pare de controlar o protocolo R2D de determinada porta, sem que a thread seja destruída.

Parâmetros:

Port – Porta que será iniciado o controle

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_THREAD_NOT_RUNNING - A thread E1 não foi inicializada ainda
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Veja Também: CreateE1Thread, ConfigE1Thread, DestroyE1Thread, EnableE1Thread.

Copyright © 2010 Digivoice Eletrônica

DisableEchoCancelation (dg_DisableEchoCancelation)

Desabilita cancelamento de eco em recursos de conferência.

Declarações:

ActiveX:
```
SHORT DisableEchoCancelation(SHORT Port);
```

API:
```
short dg_DisableEchoCancelation(short port);
```

Descrição:

O método DisableEchoCancelation desabilita o cancelamento de eco nos recursos de conferência.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

DisableGSMThread (dg_DisableGSMThread)

Desabilita a thread GSM.

Declarações:

ActiveX:
```
SHORT DisableGSMThread(SHORT Port);
```

API:
```
short dg_DisableGSMThread(short port);
```

Descrição:

Desabilita a comunicação da aplicação com os módulos GSM da porta especificada

Parâmetros:

Port – Porta que será iniciado o controle

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_THREAD_NOT_RUNNING - Thread de controle de GSM não está em funcionamento

Veja também: EnableGSMThread.

Copyright © 2010 Digivoice Eletrônica

DisableInputBuffer (dg_DisableInputBuffer)

Interrompe o envio de amostras da placa para a aplicação.

Declarações:

ActiveX:
```
SHORT DisableInputBuffer(SHORT Port);
```

API:
```
short dg_DisableInputBuffer(short port);
```

Descrição:

O método DisableInputBuffer interrompe o envio de amostras da placa para a aplicação, iniciada pelo EnableInputBuffer.

Este método deve ser chamado sempre após o StopRecordFile caso haja uma gravação em curso.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

DisableISDNThread (dg_DisableISDNThread)

Desabilita o tratamento do protocolo ISDN PRI de troncos digitais.

Declarações:

ActiveX:
```
SHORT DisableISDNThread(SHORT Card, SHORT Framer);
```

API:
```
short dg_DisableISDNThread(short card, short framer);
```

Descrição:

Este método faz com que a thread pare de controlar o protocolo ISDN de determinado span, sem que a thread seja destruída.

Parâmetros:

Card– Indica a placa que a thread será desabilitada
Framer - Indica o span específico da placa que a thread será desabilitada

Valor de Retorno:

DG_EXIT_SUCCESS - Desabilitou thread ISDN com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro placa foi passado com valor zero ou maior do que o número de placas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Framer fora do valor especificado
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1.
DG_ERROR_THREAD_NOT_RUNNING - Thread ISDN não está habilitada

Copyright © 2010 Digivoice Eletrônica

Veja Também: EnableISDNThread, ConfigISDNThread.

DisableMailBoxDetection (dg_DisableMailBoxDetection)

Desabilita a detecção de caixa postal (secretária eletrônica).

Declarações:

ActiveX:
```
SHORT DisableMailBoxDetected(SHORT Port);
```

API:
```
short dg_DisableMailDetected(short port);
```

Descrição:

Ao desabilitar a detecção de caixa postal (secretária eletrônica), a VoicerLib internamente desabilita também a detecção de silêncio (DisableSilenceDetection).

Obs. Após o evento OnMailBoxDetected acontecer, este método é chamado automaticamente pela VoicerLib.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

DisablePulseDetection (dg_DisablePulseDetection)

Desabilita a detecção de pulso.

Declarações:

ActiveX:
```
SHORT DisablePulseDetection(SHORT Port);
```

API:
```
short dg_DisablePulseDetection(short port);
```

Descrição:

Este método desabilita a detecção de pulso.

Parâmetros:

Port – Indica a porta da placa que gerou o evento

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

DisableSilenceDetection (dg_DisableSilenceDetection)

Desabilita a detecção de silêncio

Declarações:

ActiveX:
```
SHORT DisableSilenceDetection(SHORT Port);
```

API:
```
short dg_DisableSilenceDetection(short port);
```

Descrição:

O método DisableSilenceDetection desabilita a detecção de silêncio da VoicerLib que foi habilitada pelo EnableSilenceDetection.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

DisconnectAudioChannels (dg_DisconnectAudioChannels)

Efetua uma desconexão de duas portas conectadas de qualquer placa.

Declarações:

ActiveX:

```
SHORT DisconnectAudioChannels(SHORT Port1, SHORT
Port2);
```

API:

```
short dg_DisconnectAudioChannels(short port1,short
port2);
```

Descrição:

Ao chamar este método, as duas portas voltam a ter o comportamento inicial, de antes de serem conectadas pelo ConnectAudioChannels.

Parâmetros:

Port1 – Indica a primeira porta a ser desconectada
Port2 –Indica a segunda porta a ser desconectada

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

EnableAGC (dg_EnableAGC)

Habilita o ajuste automático de ganho em um determinada porta

Declarações:

ActiveX:
```
SHORT EnableAGC(SHORT Port);
```

API:
```
short dg_EnableAgc(short port);
```

Descrição:

O AGC (Automatic Gain Control) ou controle automático de ganho é muito útil quando existe uma diferença de ganho entre os interlocutores numa gravação ou conferência.

Ao habilitar este controle, o desenvolvedor pode melhorar substancialmente a qualidade da gravação ou a conversação em um conferência

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

EnableAnswerDetection (dg_EnableAnswerDetection)

Habilita a supervisão de atendimento.

Declarações:

ActiveX:
```
SHORT EnableAnswerDetection(SHORT Port);
```

API:
```
short dg_EnableAnswerDetection(short port);
```

Descrição:

O método EnableAnswerDetection habilita a supervisão de atendimento. A supervisão de linha permite monitorar quando uma ligação for atendida. É necessário que a thread de controle de callprogress tenha sido iniciada (CreateCallProgress) e que o método EnableCallProgress também tenha sido chamado para que a detecção de atendimento possa funcionar.

A chamada deste método não tem efeito nas placas E1 pois, devido à caracterítica do protocolo utilizado, a detecção sempre está habilitada.

Ao ser detectado o atendimento, o evento OnAnswerDetected é gerado indicando o tipo de detecção (por áudio ou por timeout) e poderá ser tratado pela aplicação. Nas placas analógicas esta detecção é gerada a partir da presença de áudio na linha. Nas placas digitais, o atendimento é um dado, portanto, o evento é gerado no atendimento, independente da existência de áudio.

A VoicerLib 4 tem funções específicas para detecção de silêncio, não sendo mais necessário utilizar a detecção de atendimento para este fim.

Copyright © 2010 Digivoice Eletrônica

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_THREAD_NOT_RUNNING - Thread de controle de callprogress não está em funcionamento

Copyright © 2010 Digivoice Eletrônica

EnableCallProgress (dg_EnableCallProgress)

Habilita a supervisão de linha, chamada, fax e ocupado.

Declarações:

ActiveX:
```
SHORT EnableCallProgress(SHORT Port, SHORT CPType);
```

API:
```
short dg_EnableCallProgress(short port, short
cptype);
```

Descrição:

O método EnableCallProgress habilita a supervisão de linha. A supervisão de linha permite monitorar o tom de discagem, ocupado, chamada e fax.

Quando o sistema é iniciado, a supervisão está desabilitada como padrão, portanto é necessário criar uma thread de controle de callprogress para cada porta através do método CreateCallProgress e chamar o EnableCallProgress quando se quiser monitorar a linha.

O parâmetro CPType indica o tipo de supervisão a ser iniciada. Consulte o "Guia do Programador" no tópico Supervisão de Linha para saber maiores detalhes de quando utilizar cada tipo de call progress.

Parâmetros:
Port – Indica a porta da placa.
CPType - Tipo de supervisão
- CP_ENABLE_GENERIC_TONE - Desenvolvida para detectar a presença de um tom qualquer pré-configurado ou o atendimento

Copyright © 2010 Digivoice Eletrônica

de chamada, se esta facilidade estiver habilitada na VoicerLib pelo método EnableAnswerDetection
- CP_ENABLE_LINETONE_OR_BUSY - Desenvolvida para a detecção rápida de tom de discar (tom de linha) ou ocupado antes do início de uma discagem ou após um Flash
- CP_ENABLE_BUSY_OR_FAX - Desenvolvida para a detecção rápida de tom de ocupado ou sinal de FAX após o atendimento de uma chamada
- CP_ENABLE_ALL - Desenvolvida para a detecção de tons de discar, de chamada, de ocupado, de FAX e atendimento entre uma discagem e o atendimento pelo assinante chamado

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro CPType fora do intervalo permitido
DG_ERROR_THREAD_NOT_RUNNING - Thread de controle não foi iniciada pelo CreateCallProgress.

Copyright © 2010 Digivoice Eletrônica

EnableDebug (dg_EnableDebug)

Habilita o recurso de depuração da VoicerLib em ambiente Windows.

Declarações:

ActiveX:
```
SHORT EnableDebug(SHORT UdpPort);
```

API:
```
short dg_EnableDebug(short UdpPort);
```

Descrição:

A VoicerLib tem um recurso de depuração que permite que seja enviado mensagens via TCP/IP (UDP) de todos os comandos de baixo nível enviados para ou recebidos da placa. Para isso é utilizado o recurso de broadcast do protocolo UDP, permitindo que as mensagens sejam monitoradas de outra aplicação, mesmo que seja em um computador remoto.

Este recurso é útil para fins de localização de problemas, porém não deve ser utilizado em ambientes de produção que não necessitem de depuração pois consome recursos de CPU.

Obs. Veja o tópico
Depurando aplicativos para placas E1 com R2MFC.

Parâmetros:

UdpPort - Informa qual a porta UDP deverá ser monitorada pela aplicação cliente

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso

Copyright © 2010 Digivoice Eletrônica

EnableDTMFFilter (dg_EnableDTMFFilter)

Habilita filtro de DTMF

Declarações:

ActiveX:
```
SHORT EnableDTMFFilter(SHORT Port);
```

API:
```
short dg_EnableDTMFFilter(short port);
```

Descrição:

O método EnableDTMFFilter habilita a filtragem de tons DTMF e 425Hz. Esta filtragem é necessária principalmente em situações de conferência para evitar que os dígitos detectados por uma porta sejam também detectados por outra na mesma conferência. O mesmo vale para o tom 425hz pois o tom de ocupado também não pode se propagar em todos os canais de conferência.

Em uma aplicação de conferência típica, vários canais estão compartilhando o mesmo recurso e tudo que se ouve ou se fala em uma porta é propagado pelas outras. Muitas vezes é necessário que o usuário interaja com o sistema através de dígitos do telefone. Para permitir isso sem que exista interferência das outras portas, essa filtragem se torna necessária.

Importante: Este recurso não permite a utilização simultânea da gravação em formato GSM.

Parâmetros:

Port – Indica a porta da placa

Copyright © 2010 Digivoice Eletrônica

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

EnableE1Thread (dg_EnableE1Thread)

Habilita o tratamento automático do protocolo R2D de troncos digitais.

Declarações:

ActiveX:
```
SHORT EnableE1Thread(SHORT Port);
```

API:
```
short dg_EnableE1Thread(short port);
```

Descrição:

Após o uso do DisableE1Thread é necessário chamar o EnableE1Thread para que a thread de controle R2 possa voltar a gerenciar o protocolo R2D.

Parâmetros:

Port – Porta que será iniciado o controle

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Veja Também: CreateE1Thread, ConfigE1Thread, DestroyE1Thread, DisableE1Thread

Copyright © 2010 Digivoice Eletrônica

EnableEchoCancelation (dg_EnableEchoCancelation)

Habilita cancelamento de eco em recursos de conferência.

Declarações:

ActiveX:
```
SHORT EnableEchoCancelation(SHORT Port);
```

API:
```
short dg_EnableEchoCancelation(short port, short
taps, short training);
```

Descrição:

O método EnableEchoCancelation habilita o cancelamento de eco nos recursos de conferência, evitando a degradação do sinal causada pelo efeito de eco no áudio.

ATENÇÃO: Nas placas E1, o cancelamento de eco só pode ser utilizado nas placas com 30 canais. Nas placas com 60 canais, o método retorna DG_FEATURE_NOT_SUPPORTED e o cancelamento de eco não é ativado. Pode-se forçar uma placa de 60 canais funcionar apenas com 30 e suportar cancelamento de eco chamando-se o método ForceSingleSpan antes de se iniciar a VoicerLib.

Obs. Caso utilize o ActiveX, os parâmetros taps e training serão:
taps = ECHO_TAPS_64 (ou ECHO_TAPS_128 quando o número de portas for menor ou igual a trinta na placa)
training = 800 milisegundos

Parâmetros:

Copyright © 2010 Digivoice Eletrônica

Port – Indica a porta da placa
taps (API) - Tamanho do cancelador de eco
training (API) - Tempo antes do treinamento do cancelador de eco

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa

Copyright © 2010 Digivoice Eletrônica

EnableFSKDetection (dg_EnableFSKDetection)

API | Active X

Habilita detecção de identificação de assinante no padrão FSK.

Declarações:

ActiveX:
```
SHORT EnableFSKDetection(SHORT Port, SHORT Enable,
SHORT Type);
```

API:
```
short dg_EnableFSKDetection(short port, short enable,
short type);
```

Parâmetros:

Port – Indica a porta da placa
Enable - 0 – disable, 1- enable
Type - 0 – V.23, 1 – Bell 202

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Enable ou Type diferente dos valores permitidos

Copyright © 2010 Digivoice Eletrônica

EnableGSMThread (dg_EnableGSMThread)

Habilita a thread GSM.

Declarações:

ActiveX:
```
SHORT EnableGSMThread(SHORT Port);
```

API:
```
short dg_EnableGSMThread(short port);
```

Descrição:

Habilita a comunicação da aplicação com os módulos GSM da porta especificada.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_THREAD_NOT_RUNNING - Thread de controle de GSM não está em funcionamento

Veja também: DisableGSMThread.

Copyright © 2010 Digivoice Eletrônica

EnableInputBuffer (dg_EnableInputBuffer)

Habilita o envio de amostras da placa para a aplicação.

Declarações:

ActiveX:

```
SHORT EnableInputBuffer(SHORT Port, SHORT
Agc_Enable);
```

API:

```
short dg_EnableInputBuffer(short port, short
agc_enable);
```

Descrição:

O método EnableInputBuffer inicia o envio de amostras da placa para a VoicerLib. Com isso é possível iniciar uma gravação em arquivo ou manipular as amostras diretamente na aplicação ou ainda ambas as situações simultanemente.

O que determina o que será feito com as amostras é a chamada do método RecordFile para gravá-las ou associando uma função callback com o método SetAudioInputCallback para tratá-la diretamente na aplicação (VoIP, por exemplo).

Recomenda-se chamar o EnableInputBuffer somente no início da aplicação e a cada gravação, utilizar o PauseInputBuffer quando necessário.

O parâmetro agc_enable permite habilitar o controle automatico de ganho que melhora a qualidade da gravação caso a diferença de áudio entre os interlocutores seja muito grande.

Parâmetros:

Copyright © 2010 Digivoice Eletrônica

Port – Indica a porta da placa
Agc_Enable - Habilita/desabilita controle de ganho. Pode receber DG_ENABLE_AGC(1) ou DG_DISABLE_AGC(0)

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Agc_enable diferente dos valores permitidos
DG_ERROR_THREAD_ALREADY_RUNNING - A thread de envio de amostras da placa já está em execução

Copyright © 2010 Digivoice Eletrônica

EnableISDNThread (dg_EnableISDNThread)

Habilita o tratamento do protocolo ISDN PRI.

Declarações:

ActiveX:
```
SHORT EnableISDNThread(SHORT Card, SHORT Framer);
```

API:
```
short dg_EnableISDNThread(short card, short framer);
```

Descrição:

Após o uso do DisableISDNThread é necessário chamar o EnableISDNThread para que a thread de controle possa voltar a gerenciar o protocolo ISDN PRI.

Parâmetros:

Card – Indica a placa
Framer - Indica o span da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Habilitou thread ISDN com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro placa foi passado com valor zero ou maior do que o número de placas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Framer fora do valor especificado
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1.
DG_ERROR_COULD_NOT_CREATE_THREAD - Falha na iniciação da thread ISDN

Copyright © 2010 Digivoice Eletrônica

DG_EXIT_FAILURE - Falha no envio do comando para a placa
DG_ERROR_THREAD_NOT_RUNNING - Thread ISDN não está em execução

Veja Também: DisableISDNThread , ConfigISDNThread.

EnableMailBoxDetection (dg_EnableMailBoxDetection)

A VoicerLib possui um algoritmo para a detecção de Caixa Postal (Secretária Eletrônica), que tenta reconhecer o padrão de um atendimento humano (que é um "Alô" seguido de silêncio) de um atendimento eletrônico (normalmente uma mensagem de áudio mais longa).

Para habilitar esta supervisão de atendimento, utilize este método. Caso seja detectado caixa postal (secretária eletrônica), a VoicerLib gerará o evento OnMailBoxDetected com o status 0 (áudio).

Declarações:

ActiveX:

```
SHORT EnableMailBoxDetected(SHORT Port, SHORT
SilenceTime, SHORT AudioTime, SHORT Threshold);
```

API:

```
short dg_EnableMailBoxDetected(short port, short
silencetime, short audiotime, short threshold);
```

Descrição:

Habilite a detecção depois do atendimento da porta.

Neste método a VoicerLib habilita internamente a supervisão de linha (EnableCallProgress) com CP_ENABLE_BUSY_OR_FAX e

Copyright © 2010 Digivoice Eletrônica

habilita detecção de silêncio (EnableSilenceDetection) com o tempo passado no parâmetro SilenceTime como tempo de silêncio e com o tempo passado no parâmetro AudioTime como tempo de áudio.

Obs. Sempre é necessário criar a thread de CallProgress para qualquer interface.

Parâmetros:

Port – Indica a porta da placa
AudioTime - Tempo mínimo de áudio para ser considerado um atendimento eletrônico
SilenceTime - Tempo de silêncio mínimo em que o recebedor da chamada deve ficar em silêncio para ser considerado atendimento humano. (Este tempo deve ser maior que 800 ms)
Threshold - Limiar de silêncio somente para a detecção de caixa postal, após essa detecção o valor antigo é restaurado.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

EnablePulseDetection (dg_EnablePulseDetection)

Habilita a detecção de pulso.

Declarações:

ActiveX:
```
SHORT EnablePulseDetection(SHORT Port, SHORT
Sensibility);
```

API:
```
short dg_EnablePulseDetection(short port, short
sensibility);
```

Descrição:

Este método habilita a detecção de pulso. Só é possível detectar pulso com precisão durante o silêncio.

O parâmetro sensibilidade permite alterar a sensibilidade de detecção de pulso, variando de -42dB até +12dB mas aconselha-se sempre passar zero, e variar o valor apenas caso haja algum problema na detecção de pulso.

Ao hablitar a detecção de pulso, os eventuais dígitos detectados são tratados da mesma forma que na detecção de MFs, ou seja, através do evento OnDigitDetected ou OnDigitsReceived

Parâmetros:

Port – Indica a porta da placa
Sensibility - Sensibilidade de detecção variando de -42dB até +12dB

Valor de Retorno:

Copyright © 2010 Digivoice Eletrônica

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro fora do intervalo permitido

Copyright © 2010 Digivoice Eletrônica

EnableSilenceDetection (dg_EnableSilenceDetection)

Habilita a detecção de silêncio.

Declarações:

ActiveX:

```
SHORT EnableSilenceDetection(SHORT Port, LONG
silence_time, SHORT audio_time);
```

API:

```
short dg_EnableSilenceDetection(short port, int
silence_time, short audio_time);
```

Descrição:

Este método habilita a detecção de silêncio. A VoicerLib detectará silêncio se perceber a ausência de audio por silence_time em milisegundos. O parâmetro audio_time permite modificar a sensibilidade com que a VoicerLib detectará áudio (inverso de silêncio) e é representado em milisegundos também. O padrão é passar zero no audio_time para que ao primeiro sinal de áudio, a VoicerLib já gere o evento indicando fim do silêncio.

O evento OnSilenceDetected ocorre quando se detecta o silêncio e quando se detecta o fim do silêncio. O parâmetro recebido SignalCode indica se foi detectado o silêncio recebendo o valor DG_SILENCE_DETECTED (1) ou fim de silêncio recebendo o valor DG_AUDIO_DETECTED (0).

Parâmetros:

Port – Indica a porta da placa
silence_time - Valor mínimo em milisegundos para considerar

Copyright © 2010 Digivoice Eletrônica

silêncio
audio_time - Valor mínimo em milisegundos para considerar presença de áudio ou zero para considerar no primeiro sinal de áudio (recomendado)

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

Flash (dg_Flash)

Executa um comando de flash para a central PABX.

Declarações:

ActiveX:
```
SHORT Flash(SHORT Port, SHORT MSec, SHORT
PauseAfterFlash);
```

API:
```
short dg_Flash(short port, short flash_count, long
msec, long pauseafterflash);
```

Descrição:

Com a placa FXO, as centrais PABX sempre necessitam do flash (desligar e ligar) para poder efetuar uma transferência ou outra função qualquer. O método Flash permite que seja encaminhado para a placa um comando flash com o tempo em milissegundos especificado e também um tempo de pausa após o flash. Esta pausa é útil em algumas centrais que demoram para comutar os ramais.

Nas placas E1, o flash se comporta de maneira diferente e depende de correta configuração de ramais CAS (veja em CreateCustomCAS). Nestes casos, ao executar o comando de flash, o sistema enviará comandos R2 ao PABX. Se não estiver configurado as threads de controle CAS, esta função não tem efeito nenhum.

Parâmetros:

Port – Indica a porta da placa
Milisseconds – Indica o tempo em milissegundos para o Flash (consulte a documentação do PABX para maiores detalhes)
PauseAfterFlash - Indica a pausa após flash em milissegundos.

Copyright © 2010 Digivoice Eletrônica

flash_count (API) - Indica a quantidade de flashes

Obs. Caso utilize o ActiveX, o parâmetros flash_count será 1.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_EXIT_FAILURE - Falha no envio do comando para a placa
DG_ERROR_COULD_NOT_CREATE_THREAD - Não foi possível iniciar a thread para executar esta operação

Copyright © 2010 Digivoice Eletrônica

ForceSingleSpan (dg_ForceSingleSpan)

Força uma placa de 60 canais a ser de 30 canais

Declarações:

ActiveX:
```
SHORT ForceSingleSpan(void);
```

API:
```
short dg_ForceSingleSpan(void);
```

Descrição:

Este método deve ser chamado somente no caso de se ter placas de 60 canais e se querer utilizar o cancelamento de eco nos primeiros 30 canais. É obrigatório chamá-lo antes de se iniciar a VoicerLib.

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_ALREADY_OPEN - Driver já habilitado

Copyright © 2010 Digivoice Eletrônica

GenerateMF (dg_GenerateMF)

Gerador de sinais multifrequênciais

Declarações:

ActiveX:
```
SHORT GenerateMF(SHORT Port, SHORT TypeMF, SHORT
Dig);
```

API:
```
short dg_GenerateMF(short port, short type, char
cDig);
```

Descrição:

Este método inicia a geração de sinais multifrequênciais conforme indicado no parâmetro TypeMF. Qualquer valor maior que zero inicia a geração do MF e ele só será suspenso quando for chamado o valor GENERATE_OFF.

Parâmetros:

Port – Indica a porta da placa
TypeMF – Tipo do MF a ser gerado:
GENERATE_OFF = 0
GENERATE_DTMF = 1
GENERATE_MFT = 2
GENERATE_MFF = 3
GENERATE_MF = 4
GENERATE_TONE1 = 5
GENERATE_TONE2 = 6
Dig – Dígito a ser gerado (1 a 15)

Valor de Retorno:

Copyright © 2010 Digivoice Eletrônica

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro fora do intervalo permitido. (Type ou Dig diferente dos valores anteriores)

Copyright © 2010 Digivoice Eletrônica

GetAbsolutePortNumber (dg_GetAbsolutePortNumber)

Retorna o número absoluto da porta

Declarações:

ActiveX:
```
SHORT GetAbsolutePortNumber(SHORT Card, SHORT
RelativePort);
```

API:
```
short dg_GetAbsolutePortNumber(short card, short
relativeport);
```

Descrição:

A porta absoluta é a numeração de 1 a n, dependendo dos tipos e da quantidade de placas instaladas. Deve ser passado o número da placa e a porta relativa. Ex.: placa 2, porta relativa 3 equivale a porta absoluta 63 em um sistema com placas digitais de 60 canais.

Parâmetros:

Card - Valor da placa (1 a n)
RelativePort – Indica a valor da porta relativa aquela placa

Valor de Retorno:

Caso o retorno seja sucesso, retornará o número da porta absoluta maior ou igual a um
Em caso de erro retorna zero

Copyright © 2010 Digivoice Eletrônica

GetAlarmStatus (dg_GetAlarmStatus)

Solicita estado dos alarmes

Declarações:

ActiveX:
```
SHORT GetAlarmStatus(SHORT Card);
```

API:
```
short dg_GetAlarmStatus(short card);
```

Descrição:

Os alarmes dos E1s das placas digitais são gerados automaticamente pela placa quando ocorrem. Este método permite solicitar manualmente o último estado dos alarmes. O evento OnE1Alarm será gerado indicando o código dos alarmes.

Parâmetros:

Card – Valor da placa (1 a n)

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro Card fora do intervalo permitido, excedendo o número de placas instaladas

Copyright © 2010 Digivoice Eletrônica

GetBusyCause (dg_GetBusyCause)

Retorna a causa do desligamento da chamada em caso de ocupado no protocolo ISDN PRI.

Declarações:

ActiveX:
```
SHORT GetBusyCause(SHORT Port);
```

API:
```
short dg_GetBusyCause(short port);
```

Descrição:

Este método pode ser executado para obter o motivo do desligamento da chamada somente após receber o evento OnBusyDetected.

Utilize o método ReturnCodeToStringCauseBusy para obter a descrição da causa de ocupado.

Parâmetros:

Port – Porta onde deve ser lida a sinalização

Valores de Retorno:

Causas Normais
- ISDN_CAUSE_UNALLOCATED_NUMBER (1)
- ISDN_CAUSE_NO_ROUTE_SPECIFIC_TRANSIT_NETWORK (2)
- ISDN_CAUSE_NO_ROUTE_DESTINATION (3)
- ISDN_CAUSE_SEND_SPECIAL_INFORMATION_TONE (4)
- ISDN_CAUSE_MISDIALLED_TRUNK_PREFIX (5)
- ISDN_CAUSE_CHANNEL_UNACCEPTABLE (6)
- ISDN_CAUSE_CALL_AWARDED (7)

Copyright © 2010 Digivoice Eletrônica

- ISDN_CAUSE_PREEMPTION (8)
- ISDN_CAUSE_PREEMPTION_CIRCUIT_RESERVED (9)
- ISDN_CAUSE_NORMAL_CALL_CLEARING (16)
- ISDN_CAUSE_USER_BUSY (17)
- ISDN_CAUSE_NO_USER_RESPONDING (18)
- ISDN_CAUSE_NO_ANSWER_FROM_USER (19)
- ISDN_CAUSE_SUBSCRIBER_ABSENT (20)
- ISDN_CAUSE_CALL_REJECTED (21)
- ISDN_CAUSE_NUMBER_CHANGED (22)
- ISDN_CAUSE_REDIRECTION_TO_NEW_DESTINATION (23)
- ISDN_CAUSE_EXCHANGE_ROUTING_ERROR (25)
- ISDN_CAUSE_NON_SELECTED_USER_CLEARING (26)
- ISDN_CAUSE_DESTINATION_OUT_OF_ORDER (27)
- ISDN_CAUSE_INVALID_NUMBER_FORMAT (28)
- ISDN_CAUSE_FACILITY_REJECTED (29)
- ISDN_CAUSE_STATUS_ENQUIRY (30)
- ISDN_CAUSE_NORMAL_UNSPECIFIED (31)

Recurso Indisponível
- ISDN_CAUSE_NO_CIRCUIT_CHANNEL_AVAILABLE (34)
- ISDN_CAUSE_NETWORK_OUT_OF_ORDER (38)
- ISDN_CAUSE_PERMANENT_FRAME_MODE_CONNECTION_OUT_SERVICE (39)
- ISDN_CAUSE_PERMANENT_FRAME_MODE_CONNECTION_OPERATIONAL (40)
- ISDN_CAUSE_TEMPORARY_FAILURE (41)
- ISDN_CAUSE_SWITCHING_EQUIPAMENT_CONGESTION (42)
- ISDN_CAUSE_ACCESS_INFORMATION_DISCARDED (43)
- ISDN_CAUSE_REQUEST_CIRCUIT_CHANNEL_NOT_AVAILABLE (44)
- ISDN_CAUSE_PRECEDENCE_CALL_BLOCKED (46)
- ISDN_CAUSE_RESOURCE_UNAVAILABLE_UNSPECIFIED (47)

Serviço Indisponível
- ISDN_CAUSE_QOS_NOT_AVAILABLE (49)
- ISDN_CAUSE_REQUESTED_FACILITY_NOT_SUBSCRIBED (50)

Copyright © 2010 Digivoice Eletrônica

- ISDN_CAUSE_OUTGOING_CALLS_BARRED_WITHIN_CUG (53)
- ISDN_CAUSE_INGOING_CALLS_BARRED_WITHIN_CUG (55)
- ISDN_CAUSE_BEARER_CAPABILITY_NOT_AUTHORIZED (57)
- ISDN_CAUSE_BEARER_CAPABILITY_NOT_PRESENTLY_AVAILABLE (58)
- ISDN_CAUSE_INCONSISTENCY_IN_DESIGNEATED_OUTGOING (62)
- ISDN_CAUSE_SERVICE_NOT_AVAILABLE_SPECIFIED (63)

Serviço Não Implementado
- ISDN_CAUSE_BEARER_CAPABILITY_NOT_IMPLEMENTED (65)
- ISDN_CAUSE_CHANNEL_TYPE_NOT_IMPLEMENTED (66)
- ISDN_CAUSE_REQUEST_FACILITY_NOT_IMPLEMENTED (69)
- ISDN_CAUSE_ONLY_RESTRICTED_DIGITAL_BEARER_AVAILABLE (70)
- ISDN_CAUSE_SERVICE_OPTION_NOT_IMPLEMENTED (79)

Mensagem ou Sinalização Inválida
- ISDN_CAUSE_INVALID_CALL_REFERENCE_VALUE (81)
- ISDN_CAUSE_IDENTIFIED_CHANNEL_NOT_EXIST (82)
- ISDN_CAUSE_SUSPENDED_CALL_EXIST_CALL_IDENTITY_NOT (83)
- ISDN_CAUSE_CALL_IDENTITY_IN_USE (84)
- ISDN_CAUSE_NO_CALL_SUSPENDED (85)
- ISDN_CAUSE_REQUESTED_CALL_IDENITY_HAS_BEEN_CLEARED (86)
- ISDN_CAUSE_USER_NOT_MEMBER_OF_CUG (87)
- ISDN_CAUSE_INCOMPATIBLE_DESTINATION (88)
- ISDN_CAUSE_NON_EXISTENT_CUG (90)
- ISDN_CAUSE_INVALID_TRANSIT_NETWORK_SELECTION (91)
- ISDN_CAUSE_INVALID_MESSAGE_UNSPECIFIED (95)

Erro de Protocolo
- ISDN_CAUSE_MANDATORY_INFORMATION_MISSING (96)
- ISDN_CAUSE_MESSAGE_TYPE_NON_EXISTENT (97)
-

Copyright © 2010 Digivoice Eletrônica

ISDN_CAUSE_MESSAGE_NOT_COMPATIBLE_EXISTENT_IMPLEMENTED (98)
- ISDN_CAUSE_INFORMATION_ELEMENT_NON_EXISTENT (99)
- ISDN_CAUSE_INVALID_INFORMATION_ELEMENT (100)
- ISDN_CAUSE_MESSAGE_NOT_COMPATIBLE_WITH_CALL_STATE (101)
- ISDN_CAUSE_RECOVERY_ON_TIMER_EXPIRY (102)
- ISDN_CAUSE_PARAMETER_NOT_EXISTENT_IMPLEMENTED (103)
- ISDN_CAUSE_MESSAGE_WITH_UNRECOGNIZED_PARAMETER (110)
- ISDN_CAUSE_PROTOCOL_ERROR_UNSPECIFIED (111)

Erro não especificado
- ISDN_CAUSE_INTERWORKING_UNSPECIFIED (127)

DG_ERROR_DRIVER_CLOSED: O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE: Parâmetro porta foi passado com valor zero ou maior do que o número de portas instaladas
DG_EXIT_FAILURE: Falha no envio do comando para a placa
DG_FEATURE_NOT_SUPPORTED: Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1
DG_ERROR_PORT_IS_NOT_IN_USE: A porta não está atendida

Veja Também: OnBusyDetected, ReturnCodeToStringCauseBusy.

Copyright © 2010 Digivoice Eletrônica

GetCallerID (dg_GetCallerID)

Recupera o número de A quando utilizado as threads de controle R2D MFC, ISDN PRI e Logger.

Declarações:

ActiveX:
```
BSTR GetCallerID(SHORT Port);
```

API:
```
short dg_GetCallerId(short port, char *szCallerID);
```

Descrição:

Para recuperar o número de quem chamou (CallerID) é necessário chamar este método. Os eventos pertinentes às threads de controle sinalizarão quando a informação estiver disponível.

Nas placas FXO/FXS/GSM, quando utilizado o método IdleStart, o GetCallerID pode ser chamado no evento OnCallerID para se obter o número identificado.

Parâmetros:

Port – Porta onde deve ser lida a sinalização
szCallerID (API)**–** null terminated string que receberá o número

Valores de Retorno:

ActiveX:

Retorna uma string com os dígitos ou nulo caso não haja número disponível

API:

Copyright © 2010 Digivoice Eletrônica

DG_EXIT_SUCCESS - Executado com sucesso

API / Active X:

DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1

Copyright © 2010 Digivoice Eletrônica

GetCardBus (dg_GetCardBus)

Recupera bus pci da placa

Declarações:

ActiveX:
```
LONG GetCardBus(SHORT Card);
```

API:
```
unsigned int dg_GetCardBus(short card);
```

Descrição:

Este método devolve o bus do barramento PCI da motherboard onde a placa está instalada.

Parâmetros:

Card – Valor da placa (1 a n)

Valores de Retorno:

Valor relativo ao bus do barramento PCI ou zero em caso de erro

Copyright © 2010 Digivoice Eletrônica

GetCardInfo (dg_GetCardInfo)

Informa dados sobre a placa.

Declarações:

ActiveX:
```
SHORT GetCardInfo(SHORT Card, LPCTSTR CardInfo);
```

API:
```
short dg_GetCardInfo(short card, char *szCardInfo);
```

Descrição:

Dados informados: Nome da placa, quantidade de canais, tipo de barramento (PCI, PCIe...), tipo de conector (BNC, RJ45...) e interface (E1, FXO...)

Parâmetros:

Card – Valor da placa (1 a n)
CardInfo - Armazena as informações sobre a placa

Valores de Retorno:

Se o retorno for sucesso retorna as informações sobre a placa no parâmetro CardInfo

DG_EXIT_SUCCESS - Comando executado com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GetCardInterface (dg_GetCardInterface)

Recupera tipo de placa.

Declarações:

ActiveX:
```
LONG GetCardInterface(SHORT Card);
```

API:
```
short dg_GetCardInterface(short card);
```

Descrição:

Este método devolve o tipo de placa, indicando se é placa FXO/FXS/GSM ou placa digital. O método GetCardType devolve o modelo da placa, mas como existe mais de um modelo de placa E1, a utilização do GetCardInterface facilita a tarefa de saber o tipo da placa, independente do código do modelo.

Parâmetros:

Card – Valor da placa (1 a n)

Valores de Retorno:

Se o retorno for succeso:
DG_DIGITAL_INTERFACE (1)
DG_FXO_INTERFACE (2)
DG_FX_INTERFACE(3)
DG_GSM_INTERFACE(4)
DG_UNKNOWN_INTERFACE (99)
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro card fora do intervalo de placas instaladas

Copyright © 2010 Digivoice Eletrônica

GetCardNumber (dg_GetCardNumber)

Retorna a placa de um determinada porta absoluta

Declarações:

ActiveX:
```
SHORT GetCardNumber(SHORT AbsolutePort);
```

API:
```
short dg_GetCardNumber(short absoluteport);
```

Descrição:

Retorna o número da placa de acordo com a porta absoluta passado no parâmetro. Por exemplo, a porta 61 pode ser a porta relativa 1 da segunda placa de 60 canais.

Parâmetros:

AbsolutePort – Indica a valor da porta

Valor de Retorno:

Número da placa maior ou igual a um em caso de sucesso
Zero em caso de erro

Copyright © 2010 Digivoice Eletrônica

GetCardPortsCount (dg_GetCardPortsCount)

Lê a quantidade de portas de uma determinada placa

Declarações:

ActiveX:
```
SHORT GetCardPortsCount(SHORT Card);
```

API:
```
short dg_GetCardPortsCount(short card);
```

Descrição:

Retorna o número de portas de uma determinada placa.

Parâmetros:

Card – Indica a placa que se deseja saber a quantidade de canais

Valor de Retorno:

Número de portas instaladas, ou zero para nenhuma.

Copyright © 2010 Digivoice Eletrônica

GetCardsCount (dg_GetCardsCount)

Lê a quantidade de placas disponíveis.

Declarações:

ActiveX:
```
SHORT GetCardsCount(void);
```

API:
```
short dg_GetCardsCount(void);
```

Descrição:

O método GetCardsCount retorna o número de placas instaladas no sistema.

Valor de Retorno:

Número de placas instaladas, ou zero para nenhuma.

Copyright © 2010 Digivoice Eletrônica

GetCardSlot (dg_GetCardSlot)

Recupera slot pci da placa

Declarações:

ActiveX:
```
LONG GetCardSlot(SHORT Card);
```

API:
```
unsigned int dg_GetCardSlot(short card);
```

Descrição:

Este método devolve o slot do barramento PCI da motherboard onde a placa está instalada.

Parâmetros:

Card – Valor da placa (1 a n)

Valores de Retorno:

Valor relativo ao slot do barramento PCI ou zero no caso de erro

Copyright © 2010 Digivoice Eletrônica

GetCardType (dg_GetCardType)

Recupera o tipo da placa

Declarações:

ActiveX:
```
SHORT GetCardType(SHORT Card);
```

API:
```
short dg_GetCardType(short card);
```

Descrição:

Este método devolve o código do modelo da placa (1 a n)

Parâmetros:

Card – Valor da placa (1 a n)

Valores de Retorno:
Caso seja retornado sucesso:

VBE13060PCI - Placa E1 3060 PCI
VBE16060PCI - Placa E1 60 canais PCI
VB6060PCIE - Placa E1 60 canais PCI Express
VBE13030PCI - Placa E1 30 canais PCI
VB3030PCIE - Placa E1 30 canais PCI Express
VB0408PCI - Placa FXO de 4/8 canais
VB0408PCIE - Placa FXO de 4/8 canais PCI Express
VB0404FX - Placa FXS de 4 canais
VB0404FX_R - Placa FXS de 4 canais com revisão
VB0404GSM - Placa GSM de 2/4 canais
VB1224PCIE - Placa para gravação
VB1200PCIe - Placa E1 120 canais PCI Express

Copyright © 2010 Digivoice Eletrônica

DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

GetDigits (dg_GetDigits)

Inicia a espera de uma sequência de dígitos.

Declarações:

ActiveX:

```
SHORT GetDigits(SHORT Port, SHORT MaxDigits, LPCTSTR
TermDigits, LONG DigitsTimeOut, LONG
interDigitsTimeOut);
```

API:

```
short dg_GetDigits(short port, short maxdigits, char
*termdigits, long digitstimeout, long
interdigitstimeout);
```

Descrição:

O método GetDigits permite iniciar a espera de um conjunto de dígitos, por um determinado tempo ou até receber um dígito finalizador.

Como a VoicerLib tem um processamento assíncrono, após a execução de GetDigits, é necessário tratar o resultado no evento OnDigitsReceived, o que pode acontecer segundos mais tarde. Para recuperar os dígitos detectados utilize o método ReadDigits.

Ao executar o GetDigits, o programa segue seu fluxo normal, isto é, não fica esperando a execução do GetDigits até o fim.

Para interromper a execução do GetDigits, deve ser chamado o CancelGetDigits. Esta é uma prática importante ao terminar uma ligação.

Copyright © 2010 Digivoice Eletrônica

Caso seja necessário, o método ClearDigits deve ser chamado para apagar o buffer de dígitos, antes da execução do GetDigits.

Parâmetros:

Port – Indica a porta da placa
MaxDigits – Número máximo de dígitos permitido. Utilize esta propriedade para limitar o número de dígitos que o usuário poderá teclar.
TermDigits – É uma string contendo um ou mais dígitos, que ao serem detectados finaliza a execução do GetDigits e gera o evento OnDigitsReceived. Se qualquer dígito puder interromper utilize o símbolo "@" ao invés de colocar "0123456789#*", apesar de a segunda forma também estar correta. Se não houver dígito finalizador, passar "" (vazio). É importante ressaltar que este parâmetro é uma string terminada em nulo.
DigitsTimeOut –Refere-se ao tempo máximo de espera pelo primeiro dígito programado no GetDigits. Caso seja detectado o primeiro dígito, este timeout não ocorrerá mais.
InterDigitsTimeOut - É o tempo máximo que o GetDigits esperará de intervalo entre cada dígito. Após este tempo, será gerado o evento OnDigitsReceived como código correspondete ao time-out interdígito. Em milissegundos.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GetDriverEnabled (dg_GetDriverEnabled)

Indica se a VoicerLib está inicializada.

Declarações:

ActiveX:
```
VARIANT_BOOL GetDriverEnabled(void);
```

API:
```
short dg_DriverEnabled(void);
```

Descrição:

Este método permite saber se a VoicerLib já foi inicializada através do método StartVoicerLib.

Valores de Retorno:

Na versão ActiveX o retorno é do tipo booleano (true ou false). Já a versão API tem como retorno zero (false) e um (true).

Copyright © 2010 Digivoice Eletrônica

GetE1Number (dg_GetE1Number)

Lê o número recebido durante a troca de sinalização R2D em uma chamada entrante

Declarações:

ActiveX:
```
BSTR GetE1Number(SHORT Port);
```

API:
```
short dg_GetE1Number(short port, char *szNumber);
```

Descrição:

Este método retorna uma string contendo o número recebido durante a troca de sinalização. Deve ser chamado quando chegar o evento OnE1StateChange com o status C_NUMBER_RECEIVED.

Este número pode chegar em partes, dependendo de como está configurado o recebimento da identificação de A através do método ConfigE1Thread. Se a thread for configurada para receber a identificação de A a partir do segundo dígito recebido, serão gerados dois eventos OnE1StateChange com o status C_NUMBER_RECEIVED: o primeiro receberá os dois primeiros dígitos e o segundo evento receberá o resto. A aplicação final deverá tratar isso.

Se a thread de logger estiver sendo utilizada ao invés da thread E1, este método poderá ser chamado quando o evento OnLoggerEvent com status LOGGER_LINEREADY (Início da ligação).

Parâmetros:

Port – Porta onde deve ser lida a sinalização.

Copyright © 2010 Digivoice Eletrônica

szNumber (API)**–** null terminated - string que receberá o número

Valores de Retorno:

ActiveX:

Retorna uma string com os dígitos ou nulo caso não haja número disponível

API:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_THREAD_NOT_RUNNING - Thread de Logger ou E1 não está em execução
DG_ERROR_THREAD_DISABLED - A thread ISDN está desabilitada

Copyright © 2010 Digivoice Eletrônica

GetE1ThreadStatus (dg_GetE1ThreadStatus)

Retorna o status da thread R2 da porta especificada.

Declarações:

ActiveX:
```
SHORT GetE1ThreadStatus(short port);
```

API:
```
short dg_GetE1ThreadStatus(short port);
```

Descrição:

Este método indica se a thread R2D MFC está habilitada ou desabilitada.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

Caso o retorno seja sucesso: DG_E1_THREAD_ENABLE = 1 ou DG_E1_THREAD_DISABLE = 0

DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GetISDNThreadStatus (dg_GetISDNThreadStatus)

Retorna o status da thread ISDN da placa e span especificados.

Declarações:

ActiveX:
```
SHORT GetISDNThreadStatus(SHORT Card, SHORT Framer);
```

API:
```
short dg_GetISDNThreadStatus(short card, short
framer);
```

Descrição:

Este método indica se a thread ISDN está habilitada ou desabilitada.

Parâmetros:

Card - Indica qual placa será solicitado o status
Framer - Indica o span da placa que será solicitado o status

Valores de Retorno:

DG_EXIT_SUCCESS : Retorna o status da thread ISDN
DG_ERROR_DRIVER_CLOSED: O driver não foi iniciado
DG_ERROR_CARD_OUT_OF_RANGE: Parâmetro placa foi passado com valor zero ou maior do que o número de placas instaladas
DG_ERROR_PARAM_OUTOFRANGE: Parâmetro Framer fora do valor especificado
DG_FEATURE_NOT_SUPPORTED: Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1.

Copyright © 2010 Digivoice Eletrônica

GetLibVersion (dg_GetLibVersion)

Informa o número de versão da OCX e/ou da DLL.

Declarações:

ActiveX:
```
SHORT GetVersion(LONG szVersion, SHORT nInfo);
```

API:
```
short dg_GetVersion(char* szVersion, short nInfo);
```

Descrição:

Este método permite saber qual a versão da OCX e da DLL da VoicerLib.

Parâmetros:

szVersion – Retorna a versão da VoicerLib atual da máquina
nInfo - 0 se quiser saber a versão da DLL e 1 para saber a versão da OCX da VoicerLib

Valores de Retorno:

Retorna o número da versão em caso de sucesso ou zero em caso de erro

Copyright © 2010 Digivoice Eletrônica

GetLoggerCallType (dg_GetLoggerCallType)

Lê o tipo da ligação em determinada porta de logger.

Declarações:

ActiveX:
```
SHORT GetLoggerCallType(SHORT Port);
```

API:
```
short dg_GetLoggerCallType(short port);
```

Descrição:

Este método indica se uma ligação monitorada pela thread de logger é entrante ou sainte. O sentido da ligação dependerá da conexão física dos cabos. Consulte o capítulo sobre gravação em paralelo para maiores detalhes.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

Em caso de sucesso retorna: INCOMINGCALL = 1 ou OUTGOINGCALL = 2
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_THREAD_NOT_RUNNING - A thread de logger não foi inicializada ainda

Copyright © 2010 Digivoice Eletrônica

GetNameID (dg_GetNameID)

Recupera o nome do assinante chamador quando diponível no sistema FSK e no protocolo ISDN PRI.

Declarações:

ActiveX:
```
BSTR GetNameID(SHORT Port);
```

API:
```
short dg_GetNameId(short port, char *szNameID);
```

Descrição:

Para recuperar o nome de quem chamou é necessário chamar este método após o evento OnCallerID.

Parâmetros:

ActiveX
Port – Porta onde deve ser lida a sinalização

API
szNameID (API)**–** null terminated string que receberá o nome

Valores de Retorno:

ActiveX:

Retorna no campo NameID uma string com o nome ou nulo caso não haja número disponível

API:

Copyright © 2010 Digivoice Eletrônica

Em caso de sucesso retorna o nome na variável szNameID
DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GetPlayFormat (dg_GetPlayFormat)

Recupera o formato de reprodução de uma porta.

Declarações:

ActiveX:
```
SHORT GetPlayFormat(SHORT Port);
```

API:
```
short dg_GetPlayFormat(short port);
```

Descrição:

O método GetPlayFormat retorna o formato de reprodução configurado para determinada porta.

Parâmetros:

Port – Indica a porta da placa.

Valores de Retorno:

0 - ffWaveULaw (Lei mi)
1 - ffSig (obsoleto)
2 - ffWavePCM
3 - ffGsm610
4 - ffWaveALaw (Lei A)
5 - ffWave49
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GetPortCardType (dg_GetPortCardType)

Recupera o tipo da placa através do número da porta

Declarações:

ActiveX:
```
SHORT GetPortCardType(SHORT Port);
```

API:
```
short dg_GetPortCardType(short port);
```

Descrição:

Este método devolve o código do modelo da placa de uma determinada porta. Com isso é possível saber que tipo de placa tem determinada porta.

Parâmetros:

Port – Indica a porta da placa

Valores de Retorno:

VBE13060PCI - Placa E1 3060 PCI
VBE16060PCI - Placa E1 60 canais PCI
VB6060PCIE - Placa E1 60 canais PCI Express
VBE13030PCI - Placa E1 30 canais PCI
VB3030PCIE - Placa E1 30 canais PCI Express
VB0408PCI - Placa FXO de 4/8 canais
VB0408PCIE - Placa FXO de 4/8 canais PCI Express
VB0404FX - Placa FXS de 4 canais
VB0404FX_R - Placa FXS de 4 canais com revisão
VB0404GSM - Placa GSM de 2/4 canais
VB1224PCIE - Placa para gravação
VB1200PCIE - Placa E1 120 canais PCI Express

Copyright © 2010 Digivoice Eletrônica

DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GetPortInterface (dg_GetPortInterface)

Recupera tipo de placa de uma determinada porta.

Declarações:

ActiveX:
```
SHORT GetPortInterface(SHORT Port);
```

API:
```
short dg_GetPortInterface(short port);
```

Descrição:

Este método devolve o tipo de placa de uma determinada porta, indicando se é placa FXO/FX/GSM ou digital. O método GetCardType devolve o modelo da placa, mas como existe mais de um modelo de placa E1, a utilização do GetCardInterface facilita a tarefa de saber se a placa é digital/FXO/FXS/GSM, independente do código do modelo.

Parâmetros:

Port – Indica a porta da placa

Valores de Retorno:

DG_DIGITAL_INTERFACE (1)
DG_FXO_INTERFACE (2)
DG_FX_INTERFACE (3)
DG_GSM_INTERFACE(4)
DG_FXS_INTERFACE(5)
DG_UNKNOWN_INTERFACE (99)

Copyright © 2010 Digivoice Eletrônica

GetPortsCount (dg_GetPortsCount)

Lê a quantidade de portas disponíveis.

Declarações:

ActiveX:
```
SHORT GetPortsCount(void);
```

API:
```
short dg_GetPortsCount(void);
```

Descrição:

O método GetPortsCount retorna o número de portas instaladas no sistema, somatória de todas as portas de todas as placas.

Valor de Retorno:

Número de portas instaladas, ou zero para nenhuma.

Copyright © 2010 Digivoice Eletrônica

GetPortStatus (dg_GetPortStatus)

Recupera o status da porta especificada.

Declarações:

ActiveX:
```
SHORT GetPortStatus(SHORT Port);
```

API:
```
short dg_GetPortStatus(short port);
```

Descrição:

Este método permite saber o que a porta especificada está executando em um determinado momento.

Parâmetros:

Port – Indica a porta da placa

Valores de Retorno:

spFlashing - Executando um Flash
spDialing - Discando
spNone - Ocioso
spWaitingDigits - Esperando Dígitos
spOffHook - Indicando "fora do gancho" quando utilizada a thread E1
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GetRecordFormat (dg_GetRecordFormat)

Recupera o formato de gravação de uma porta.

Declarações:

ActiveX:
```
SHORT GetRecordFormat(SHORT Port);
```

API:
```
short  dg_GetRecordFormat(short port);
```

Descrição:

O método GetRecordFormat retorna o formato de gravação configurado para determinada porta.

Parâmetros:

Port – Indica a porta da placa.

Valores de Retorno:

0 - ffWaveULaw (Lei mi)
1 - ffSig (obsoleto)
2 - ffWavePCM
3 - ffGsm610
4 - ffWaveALaw (Lei A)
5 - ffWave49
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GetRelativeChannelNumber (dg_getRelativeChannelNumber)

Converte o número de porta absoluto para o número relativo à placa

Declarações:

ActiveX:
```
SHORT GetRelativeChannelNumber(SHORT AbsolutePort);
```

API:
```
short dg_GetRelativeChannelNumber(short
absoluteport);
```

Descrição:

Converte o número de porta absoluto para o número relativo à placa. Por exemplo, a porta 61 pode ser a porta relativa 1 da segunda placa de 60 canais.

Parâmetros:

AbsolutePort – Indica a valor da porta

Valor de Retorno:

Número de porta relativa à placa, maior ou igual a um.
Caso de erro retorna zero

Copyright © 2010 Digivoice Eletrônica

GetText (dg_GetText)

Retorna uma mensagem de texto recebida pela VoicerLib (protocolo ISDN PRI).

Declarações:

ActiveX:
```
SHORT GetText(SHORT Port, LPCTSTR Text)
```

API:
```
short dg_GetText(short port, char *szText);
```

Descrição

Após receber o evento OnText é necessário chamar este método para obter o texto.
O texto não excederá 128 caracteres, não conterá acentuação nem caracteres especiais, serão compostas de caracteres ANSI (tabela ASCII Normal).

Parâmetros:

Port - Indica a porta que recebeu a mensagem
Text - Indica o texto recebido

Valores de Retorno:

DG_EXIT_SUCCESS: Executou com sucesso
DG_ERROR_DRIVER_CLOSED: O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE: Parâmetro porta foi passado com valor zero ou maior do que o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1

Copyright © 2010 Digivoice Eletrônica

GetVersion (dg_GetVersion)

Pede o número de versão do firmware

Declarações:

ActiveX:
```
SHORT GetVersion(SHORT Card)
```

API:
```
short dg_GetVersion(short card);
```

Descrição:

Este método permite saber qual a versão do firmware da placa. Pode ser útil para rastreamento de atualizações

Parâmetros:

Card – Indica a placa que se deseja saber a versão do firmware

Valores de Retorno:

Retorna o número da versão em caso de sucesso ou zero em caso de erro

Copyright © 2010 Digivoice Eletrônica

GSMCallControl (dg_GSMCallControl)

Este comando executa as funções de atendimento de segunda chamada e conferência nos módulos.
Essas funcionalidades dependem dos serviços operadora de telefônia móvel.

Declarações:

ActiveX:

```
SHORT GSMCallControl(SHORT Port,SHORT command, SHORT
call);
```

API:

```
SHORT dg_GSMCallControl(short port, short command,
short call);
```

Parâmetros:

Port - Indica a porta da placa
command - Qual função será executada
GSM_RELEASE_CALLS - Libera todas as chamadas em espera:
- Se uma chamada está chamando (waiting), libera a chamada. O chamador receberá indicação de Ocupado.
- Ou termina todas as chamadas em espera (hold).

GSM_TERMINATE_AND_ACCEPT - Termina todas as chamadas ativas (se houver) e aceita "a outra chamada" como chamada ativa:
- Se uma chamada está chamando(waiting), a chamada será aceita.
- Ou , se uma chamada está em espera (held) , esta se tornará a chamada ativa.

1+call- Termina uma chamada específica (call com call=1-7). A chamada pode estar ativa, em espera (held) ou chamando (waiting): O outro assinante receberá indicação de

Copyright © 2010 Digivoice Eletrônica

queda de ligação.

GSM_PUT_ON_HOLD - Coloca todas as chamadas ativas em espera(HOLD) e aceita " a outra chamada" como ativa:
- Se a chamada está chamando(waiting), esta será aceita.
- Senão, se uma chamada em espera(held) está presente, a mesma se tornará ativa.

2+call – Coloca todas as chamadas ativas, exceto a chamada call (call=1-7) em espera (Hold).

GSM_CONFERENCE -Adiciona uma chamada em espera(held) a uma chamada ativa para início de chamada em conferência.

Ex.
A - Assinante 1
B - Assinante 2
C - Assinante 3
D - Porta 1 VB0404GSM

A Liga para **D**
D recebe o evento OnCallerID. Lê o número de **A** com método GetCallerID.
D atende chamada (método PickUp)
B Liga para **D**
D recebe o evento OnGSMOtherCall. Lê o número de **B** com o método GetCallerID.
D coloca **A** em espera(Hold) e atende **B** com o método GSMCallControl - GSM_PUT_ON_HOLD.
A recebe tom de chamando.
D coloca **B** em espera(Hold) e fala com **A** com o método GSMCallControl - GSM_PUT_ON_HOLD.
B recebe tom de chamando.
D faz conferência com **A** e B com o método GSMCallControl - GSM_CONFERENCE.
C Liga para **D**
D recebe o evento OnGSMOtherCall. Lê o número de **C** com o método GetCallerID.
D atende **C** com o método GSMCallControl - GSM_PUT_ON_HOLD.

Copyright © 2010 Digivoice Eletrônica

D faz conferência com **A**,**B** e **C** com o método GSMCallControl - GSM_CONFERENCE.
D derruba todos com GSMCallControl - GSM_RELEASE_CALLS.

call - Qual chamada será afetada pelo parâmetro command

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GSMCheckSignalQuality (dg_GSMCheckSignalQuality)

Este comando força o módulo GSM correspondente a porta especificada, verificar o nível de sinal recebido na antena. Ao obter o nível de sinal será gerado um evento OnGSMSignalQuality após o qual deverá ser chamado o método GSMGetSignalQuality.

Este método não deve ser chamado durante uma discagem, somente antes de uma discagem e após o atendimento de uma chamada).

Declarações:

ActiveX:
```
SHORT GSMCheckSignalQuality(SHORT Port);
```

API:
```
SHORT dg_GSMCheckSignalQuality(short port);
```

Parâmetros:

Port - Indica a porta da placa.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Veja também: GSMGetSignalQuality.

Copyright © 2010 Digivoice Eletrônica

GSMClearAllSMS (dg_GSMClearAllSMS)

Este comando lê a quantidade de mensagens do módulo, e se o segundo parâmetro estiver 1, apaga todas as mensagens SMS do módulo GSM correspondente a porta especificada.
Após o envio do comando o evento OnGSMMemory será gerado.
Para obter a quantidade de mensagens no módulo GSM utilize o método GSMGetMemory após o evento OnGSMMemory.

Declarações:

ActiveX:
```
SHORT GSMClearAllSMS(SHORT Port, SHORT flagClear);
```

API:
```
SHORT dg_GSMClearAllSMS(short port, short
flagClear);
```

Parâmetros:

Port - Indica a porta da placa.
flagClear - Indica se apaga ou não todas as mensagens armazenadas.
0 - Somente mostra a quantidade de mensagens e não as apaga do módulo
1 - Lê a quantidade de mensagens do módulo e as apaga

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GSMDeleteSMS (dg_GSMDeleteSMS)

Apaga mensagem SMS do módulo GSM.

Este comando necessita que seja indicado o índice da mensagem SMS na porta especificada, para que possa apaga-lá do módulo GSM. Para saber quais os índices das mensagens, consultar o método GSMListSMS para obter a lista dos índices.

Declarações:

ActiveX:
```
SHORT GSMDeleteSMS(SHORT Port, SHORT Index);
```

API:
```
SHORT dg_GSMDeleteSMS(short port, short Index);
```

Parâmetros:

Port - Indica a porta da placa.
Índice - Indica o número correspondente a mensagem SMS.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GSMGetIndexList (dg_GSMGetIndexList)

Recupera a lista dos índices das mensagens SMS do módulo GSM. Deve ser chamado no evento OnGSMReturnOK, para o Status GSM_LIST.

Declarações:

ActiveX:

```
SHORT GSMGetIndexList(SHORT Port, LPCTSTR
szMessage);
```

API:

```
SHORT dg_GSMGetIndexList(short port, char
*szMessage);
```

Parâmetros:

Port - Indica a porta da placa.
szMessage - String com a lista dos índices das mensagens SMS armazenadas no módulo GSM.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GSMGetLastCommand (dg_GSMGetLastCommand)

Um evento OnGSMMessage é enviado sempre que uma mensagem válida, não necessariamente SMS, é recebida pelo módulo GSM. Com esse evento é possível saber também, qual foi o último comando enviado para a placa, com o método GSMGetLastCommand.

Declarações:

ActiveX:

```
SHORT GsmGetLastCommand(SHORT Port, LPCTSTR
szCommand);
```

API:

```
SHORT dg_GsmGetLastCommand(short port, char
*szCommand);
```

Parâmetros:

Port - Indica a porta da placa
szCommand - String onde o último comando enviado será copiado pelo método

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GSMGetMemory (dg_GSMGetMemory)

Esse método disponibiliza ao usuário a quantidade de mensagens SMS no módulo GSM após o recebimento do evento OnGSMMemory.

Esse método informa a quantidade atual de mensagens e a capacidade máxima que o módulo especificado pode receber.

Declarações:

ActiveX:
```
SHORT GsmGetMemory(SHORT Port, LPCTSTR szMessage);
```

API:
```
SHORT dg_GsmGetMemory(short port, char *szMessage);
```

Parâmetros:

Port - Indica a porta da placa
szMessage - String com a quantidade de mensagens e a capacidade máxima do chip (SIM Card).
Por exemplo: 2,40 (Possui 2 mensagens armazenadas, e a capacidade máxima é de 40)

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GSMGetMessage (dg_GSMGetMessage)

Um evento OnGSMMessage é enviado sempre que uma mensagem válida, não necessariamente SMS, é recebida pelo módulo GSM. As mensagens recebidas podem ser obtidas pelo método GSMGetMessage.Este método deve ser usado para fins de debug.

Declarações:

ActiveX:
```
SHORT GsmGetMessage(SHORT Port, LPCTSTR szMessage);
```

API:
```
SHORT dg_GsmGetMessage(short port, char *szMessage);
```

Parâmetros:

Port - Indica a porta da placa
szMessage - String onde a mensagem recebida será copiada pelo método

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GSMGetSignalQuality (dg_GSMGetSignalQuality)

Recupera a string com o valor da qualidade do sinal após o recebimento do evento OnGSMSignalQuality .

Declarações:

ActiveX:
```
SHORT GsmGetSignalQuality(SHORT Port, LPCTSTR
szMessage);
```

API:
```
SHORT dg_GsmGetSignalQuality(short port, char
*szMessage);
```

Parâmetros:

Port - Indica a porta da placa
szMessage - Os valores podem variar de 0 a 31, sendo 0 igual ou inferior a -113 dBm e 31 igual ou superior a -51 dBm. O valor 99 indica que o sinal não pode ser detectável.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Veja também: GSMCheckSignalQuality.

Copyright © 2010 Digivoice Eletrônica

GSMGetSMS (dg_GSMGetSMS)

Permite ler uma mensagem SMS recebida. Este este método deve ser chamado no recebimento do evento que avisa que uma mensagem SMS foi recebida (OnGSMSMSReceived)

Declarações:

ActiveX:
```
SHORT GsmGetSMS(SHORT Port, LPCTSTR szMessage);
```

API:
```
SHORT dg_GsmGetSMS(short port, char *szMessage);
```

Parâmetros:

Port - Indica a porta da placa
szMessage - Mensagem recebida.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Veja também: GSMSendSMS.

Copyright © 2010 Digivoice Eletrônica

GSMGetSMSConfirmation (dg_GSMGetSMSConfirmation)

Permite ler a string de confirmação do recebimento de uma mensagem SMS por parte do destinatário. Este este método deve ser chamado no evento que confirma o recebimento de uma mensagem SMS por parte do destinatário (OnGSMSMSConfirmation).

Declarações:

ActiveX:

```
SHORT GsmGetSMSConfirmation(SHORT Port, LPCTSTR
szMessage);
```

API:

```
 SHORT dg_GsmGetSMSConfirmation(short port, char
*szMessage);
```

Parâmetros:

Port - Indica a porta da placa
szMessage - Mensagem de confirmação do destinatário.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Veja também: GSMSendSMS.

Copyright © 2010 Digivoice Eletrônica

GSMListSMS (dg_GSMListSMS)

Solicita ao módulo GSM, a lista de mensagens GSM armazenadas em sua memória..
O parâmetro Status representa quais tipos de mensagens serão listadas. Recomenda-se o uso do status ALL.

Declarações:

ActiveX:
```
SHORT GSMListSMS(SHORT Port, SHORT Status);
```

API:
```
SHORT dg_GSMListSMS(short port, short Status);
```

Parâmetros:

Port - Indica a porta da placa.
Status - Tipos de mensagens.
- REC_UNREAD - Mensagens recebidas e não lidas
- REC_READ - Mensagens recebidas e lidas
- STO_UNSENT - Mensagens armanzenadas e não enviadas
- STO_SENT - Mensagens armanzenadas e enviadas
- ALL - Todas as mensagens

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GsmRawToWave (dg_GsmRawToWave)

Converte arquivo de áudio do formato GSMRaw (sem cabeçalho) para o formato Wave.

Declarações:

ActiveX:
```
SHORT GsmRawToWave(LPCTSTR Source, LPCTSTR Target);
```

API:
```
SHORT dg_GsmRawToWave(char *Source, char *Target);
```

Parâmetros:

Source - String contendo o nome e o caminho completo do arquivo GsmRaw que será convertido.
Target - String contendo o nome e o caminho completo do arquivo Wave que será gravado.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_CONV_SOURCE - Erro ao abrir o arquivo de origem
DG_ERROR_CONV_TARGET - Erro ao abrir o arquivo de destino
DG_ERROR_CONV - Erro indeterminado na conversão

Veja também: WaveToGsmRaw.

Copyright © 2010 Digivoice Eletrônica

GsmRawToWave49 (dg_GsmRawToWave49)

Converte arquivo de áudio do formato GSMRaw (sem cabeçalho) para o formato Wave49 (GSM 6.10 modificado).

Declarações:

ActiveX:
```
SHORT GsmRawToWave49(LPCTSTR Source, LPCTSTR
Target);
```

API:
```
SHORT dg_GsmRawToWave49(char *Source, char *Target)
```

Parâmetros:

Source - String contendo o nome e o caminho completo do arquivo GsmRaw que será convertido.
Target - String contendo o nome e o caminho completo do arquivo Wave49 que será gravado.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_CONV_SOURCE - Erro ao abrir o arquivo de origem
DG_ERROR_CONV_TARGET - Erro ao abrir o arquivo de destino
DG_ERROR_CONV - Erro indeterminado na conversão

Veja também: Wave49ToGsmRaw.

Copyright © 2010 Digivoice Eletrônica

GSMReadAndDeleteSMS (dg_GSMReadAndDeleteSMS)

Este comando lê a mensagem SMS especificada pelo índice e em seguida apaga a mensagem do módulo GSM.
Essa mensagem pode ser lida com o método GSMGetSMS após o recebimento do evento OnGSMSMSReceived .

Declarações:

ActiveX:
```
SHORT GSMReadAndDeleteSMS(SHORT Port, SHORT Index);
```

API:
```
SHORT dg_GSMReadAndDeleteSMS(short port, short
Index);
```

Parâmetros:

Port - Indica a porta da placa.
Índice - Indica o número correspondente a mensagem SMS.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GSMRestartPort (dg_GSMRestartPort)

Este comando reinicia o módulo especificado pela porta.

Declarações:

ActiveX:
```
SHORT GSMRestartPort(SHORT Port);
```

API:
```
SHORT dg_GSMRestartPort(short port);
```

Parâmetros:

Port - Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GSMSendCommand (dg_GSMSendCommand)

Envia um comando ao módulo GSM correspondente à porta especificada.

Declarações:

ActiveX:
```
SHORT GsmSendCommand(SHORT Port, LPCTSTR szCommand);
```

API:
```
SHORT dg_GsmSendCommand(short port, char
*szCommand);
```

Descrição:

Os módulos GSM se comunicam com a VoicerLib através de um set de comandos conhecidos como comandos AT que foram estendidos para aplicações GSM. Nem todos os comandos tem um método correspondente na VoicerLib. O método GsmSendCommand permite o envio de um string contendo qualquer comando AT aos módulos GSM. O retorno dos módulos ao comando enviado pode ser obtido com o método GSMGetMessage, após o recebimento do evento OnGSMMessage.

Cuidado: O envio incorreto de comandos aos módulos podem prejudicar seu funcionamento, portanto não use este comando se não tiver certeza do que estiver fazendo.

Parâmetros:

Port - Indica a porta da placa
szCommand - string contendo o comando AT a ser enviado ao módulo GSM.

Valor de Retorno:

Copyright © 2010 Digivoice Eletrônica

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GSMSendSMS (dg_GSMSendSMS)

Permite o envio de mensagem SMS nas portas correspondentes.

Declarações:

ActiveX:
```
SHORT GsmSendSMS(SHORT Port, LPCTSTR szNumber,
LPCTSTR szMessage);
```

API:
```
SHORT dg_GsmSendSMS(short port, char *szNumber, char
*szMessage);
```

Descrição:

A mensagem não pode exceder 160 caracteres e deve ser finalizada com '\0'. A mensagem poderá conter CR (0xd) ou LF (0xa).As mensagens SMS devem ser compostas de caracteres ANSI pois serão automaticamente convertidas para carateres GSM pela VoicerLib antes do envio. Esta conversão afeta caracteres especiais, principalmente acentuados.

Se a mensagem for enviada com sucesso, o evento OnGSMSMSSent será gerado indicando que a porta está pronta e disponível para uma nova operação.

Parâmetros:

Port - Indica a porta da placa
szNumber - Número que vai receber a mensagem (destinatário)
szMessage - Mensagem que deseja enviar.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado

Copyright © 2010 Digivoice Eletrônica

DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro szMessage fora do intervalo permitido - quantidade de caracteres excedeu 160.

Veja também: GSMGetSMS.

Copyright © 2010 Digivoice Eletrônica

GSMSetPinNumber (dg_GSMSetPinNumber)

Ao ser iniciada a thread GSM a VoicerLib consulta os módulos GSM sobre a necessidade ou não do envio de PIN number ou PUK. Atualmente a maioria das operadoras não solicitam o PIN number a cada inicialização dos módulos GSM.

Se a operadora de telefonia móvel solicitar a validação com PIN number, o seu string deverá estar disponível antes da inicialização da thread GSM através do método GsmSetPinNumber. Em alguns casos há a troca do PIN number e o novo valor deve ser passado, caso contrário um valor nulo (NULL) deve ser colocado na passagem do parâmetro.

Declarações:

ActiveX:

```
SHORT GsmSetPinNumber(SHORT Port, LPCTSTR szPIN,
LPCTSTR szNewPIN);
```

API:

```
SHORT dg_GsmSetPinNumber(short port, char *szPIN,
char *szNewPIN);
```

Parâmetros:

Port - Indica a porta da placa
szPIN - String contendo o PIN number atual.
szNewPIN - String contendo o novo PIN number em caso de troca.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

GsmToWave (dg_GsmToWave)

Converte arquivo de áudio do formato GSM para o formato Wave.

Declarações:

ActiveX:
```
SHORT GsmToWave(LPCTSTR Source, LPCTSTR Target);
```

API:
```
SHORT dg_GsmToWave(char *Source,char *Target);
```

Parâmetros:

Source - String contendo o nome e o caminho completo do arquivo gsm que será convertido.
Target - String contendo o nome e o caminho completo do arquivo wave que será gravado.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_CONV_SOURCE - Erro ao abrir o arquivo de origem
DG_ERROR_CONV_TARGET - Erro ao abrir o arquivo de destino
DG_ERROR_CONV - Erro indeterminado na conversão

Veja também: WaveToGsm.

Copyright © 2010 Digivoice Eletrônica

GsmToWave49 (dg_GsmToWave49)

Converte arquivo de áudio do formato GSM para o formato Wave49 (GSM 6.10 modificado).

Declarações:

ActiveX:
```
SHORT GsmToWave49(LPCTSTR Source, LPCTSTR Target);
```

API:
```
SHORT dg_GsmToWave49(char *Source,char *Target);
```

Parâmetros:

Source - String contendo o nome e o caminho completo do arquivo Gsm que será convertido.
Target - String contendo o nome e o caminho completo do arquivo Wave49 que será gravado.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_CONV_SOURCE - Erro ao abrir o arquivo de origem
DG_ERROR_CONV_TARGET - Erro ao abrir o arquivo de destino
DG_ERROR_CONV - Erro indeterminado na conversão

Veja também: Wave49ToGsm.

Copyright © 2010 Digivoice Eletrônica

HangUp (dg_HangUp)

Libera a linha conectada a placa (desliga).

Declarações:

ActiveX:
```
SHORT HangUp(SHORT Port);
```

API:
```
short dg_HangUp(short port);
```

Descrição:

Ao chamar este método, a linha é desconectada e a porta liberada, equivalendo ao desligar do telefone. Na placa E1 o hangup faz com que a linha seja desconectada e liberada.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Porta especificada fora do intervalo de portas configuradas
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1
DG_ERROR_FIFO_UNAVAILABLE - Fifo de comunicação entre threads não está disponível. Falta de memória pode ser a causa mais comum
DG_ERROR_THREAD_NOT_RUNNING - A thread não está rodando ou não está habilitada
DG_EXIT_FAILURE - Ocorre caso não seja possível inserir

Copyright © 2010 Digivoice Eletrônica

comando na fila das threads de controle
DG_ERROR_THREAD_DISABLED - A thread ISDN está desabilitada

Veja Também: Se o protocolo usado for ISDN PRI, ler o tópico SetHangUpCause.

IdleAbort (dg_IdleAbort)

Interrompe a execução de uma função de Idle.

Declarações:

ActiveX:
```
SHORT IdleAbort(SHORT Port);
```

API:
```
short dg_IdleAbort(short port);
```

Descrição:

Ao chamar este método a monitoração de linha para atendimento será cancelada.

Parâmetros:

Port – Porta onde será feita o cancelamento da monitoração

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

IdleSettings (dg_IdleSettings)

Configura a monitoração do estado de espera.

Declarações:

ActiveX:

```
SHORT IdleSettings(SHORT Port, VARIANT_BOOL
AutoPickUp, SHORT RingCount, SHORT PauseAfterPickUp,
VARIANT_BOOL WatchTrunkBefore, VARIANT_BOOL
WatchTrunkAfter, SHORT Format, SHORT TimeOut, SHORT
Max, LPCTSTR TermDigits);
```

API:

```
short WCDECL dg_IdleSettings(short port, u8
AutoPickUp, short RingCount, short PauseAfterPickUp,
u8 WatchTrunkBefore, u8 WatchTrunkAfter,short
Format, short TimeOut, short Max, char *TermDigits);
```

Descrição:

Esta função configura as opções a serem monitoradas durante o estado de espera. Permite configurar detecção automática de BINA (OnCallerID), atendimento automático e integração com o PABX

Parâmetros:

Port – Indica a porta da placa que será monitorado
AutoPickUp – Indica se atende automático ou não. (0/1 na API e TRUE/FALSE no ActiveX)
RingCount – Número de rings para o atendimento automático
PauseAfterPickUp – Pausa após o pickup
WatchTrunkBefore – Monitora a linha antes do atendimento. (0/1 na API e TRUE/FALSE no ActiveX)
WatchTrunkAfter - Monitora a linha depois do atendimento. (0/1 na API e TRUE/FALSE no ActiveX)
Format - wtDTMF, wtMFP, wtCustom – Ver detalhes no capítulo

Copyright © 2010 Digivoice Eletrônica

Funções Especiais.
Timeout – Timeout interdigito
Max – Número máximo de dígitos da sinalização
TermDigits – Indica um ou mais dígitos como finalizadores.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

IdleStart (dg_IdleStart)

Inicia a execução de uma função de Idle.

Declarações:

ActiveX:
```
SHORT IdleStart(SHORT Port);
```

API:
```
short dg_IdleStart(short port);
```

Descrição:

Ao chamar este método, a função de monitoração do estado de espera é iniciada. O método IdleSettings deve ser chamado com todas as configurações necessárias antes de iniciar a monitoração.

Parâmetros:

Port – Porta onde será feita a monitoração

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_THREAD_ALREADY_RUNNING - A thread já havia sido iniciada anteriormente
DG_ERROR_COULD_NOT_CREATE_THREAD - Falha na inicialização da thread
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

IsCallInProgress

Indica se existe uma discagem em curso

Declarações:

ActiveX:

```
VARIANT_BOOL IsCallInProgress(SHORT Port);
```

Descrição:

Este método permite saber se existe uma discagem em curso, iniciada pelo MakeCall.

Parâmetros:

Port – Indica a porta da placa que gerou o evento

Valor de Retorno:

True - Existe discagem em curso
False - Nenhuma discagem em curso

Copyright © 2010 Digivoice Eletrônica

IsPlaying (dg_IsPlaying)

Indica se a placa está reproduzindo alguma mensagem

Declarações:

ActiveX:
```
VARIANT_BOOL IsPlaying(SHORT Port);
```

API:
```
short IsPlaying(short port);
```

Descrição:

Este método permite ao desenvolvedor saber se a porta especificada está reproduzindo uma mensagem

Parâmetros:

Port – Indica a porta da placa

Valores de Retorno:

TRUE(1) - A placa está reproduzindo alguma mensagem
FALSE(0)- A placa não está reproduzindo nenhuma mensagem

ActiveX - TRUE ou FALSE
API - 0 ou 1

Copyright © 2010 Digivoice Eletrônica

IsRecording (dg_IsRecording)

Indica se a placa está gravando alguma mensagem

Declarações:

ActiveX:
```
VARIANT_BOOL IsRecording(SHORT Port);
```

API:
```
short IsRecording(short port);
```

Descrição:

Este método permite ao desenvolvedor saber se a porta especificada está gravando uma mensagem

Parâmetros:

Port – Indica a porta da placa

Valores de Retorno:

TRUE(1) - A placa está gravando alguma mensagem
FALSE(0)- A placa não está gravando nenhuma mensagem

ActiveX - TRUE ou FALSE
API - 0 ou 1

Copyright © 2010 Digivoice Eletrônica

LocalBridgeConnect (dg_LocalBridgeConnect)

Efetua uma conexão bi-direcional entre duas portas na placa E1.

Declarações:

ActiveX:
```
SHORT LocalBridgeConnect(SHORT Port1, SHORT Port2);
```

API:
```
short dg_LocalBridgeConnect(short port1, short
port2);
```

Descrição:

Ao chamar este método, a porta Port1 é conectada em Port2 e vice-versa, permitindo que o áudio seja enviado nos dois sentidos. Ambas as portas precisam pertencer à mesma placa, obrigatoriamente.

Parâmetros:

Port1 – Indica a primeira porta
Port2 – Indica a segunda porta

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_FEATURE_NOT_SUPPORTED - Comando não suportado por este tipo de placa
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

LocalBridgeDisconnect (dg_LocalBridgeDisconnect)

Efetua uma desconexão de duas portas conectadas, em placas E1.

Declarações:

ActiveX:
```
SHORT LocalBridgeDisconnect(SHORT Port1, SHORT
Port2);
```

API:
```
short  dg_LocalBridgeDisconnect(short port1, short
port2);
```

Descrição:

Ao chamar este método, as duas portas voltam a ter o comportamento inicial, de antes de serem conectadas pelo LocalBridgeConnect

Parâmetros:

Port1 – Indica a primeira porta
Port2 – Indica a segunda porta

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

MakeCall

Inicia a discagem com supervisão.

Declarações:

ActiveX:

```
SHORT MakeCall(SHORT Port, SHORT CallType, LPCTSTR
Number, LPCTSTR InitialPhrase, VARIANT_BOOL
WithAnalysis, SHORT DialType);
```

Descrição:

Este método inicia a discagem com ou sem supervisão conforme configurado pelos métodos SetCall*xxxxx*. O término será tratado no evento OnAfterMakeCall.

Parâmetros:

Port – Indica a porta da placa
CallType – ctExternal(0) ou ctWithFlash(1)
Number – String do número que será discado
InitialPhrase – String da frase a ser reproduzida no início do processo
WithAnalysis – True/False que indicará se a discagem será com supervisão (monitorar ocupado, etc...) ou sem (ligação entregue após a discagem).
DialType– Define o tipo de discagem:dtPulse (pulso -> 0) /dtTone (tom -> 1).

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_DRIVER_CLOSED - Driver desabilitado

Copyright © 2010 Digivoice Eletrônica

MenuAbort

Interrompe a execução de uma função de Menu.

Declarações:

ActiveX:

```
SHORT MenuAbort(SHORT Port);
```

Descrição:

Ao chamar este método a função iniciada pelo MenuStart, será interrompida para a porta específica.

Parâmetros:

Port – Indica a porta da placa que gerou o evento

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

MenuErrorSettings

Configura as opções de menu

Declarações:

ActiveX:

```
SHORT MenuErrorSettings(SHORT Port, LPCTSTR
InvalidDigitPhrase, SHORT InvalidDigitRetries,
LPCTSTR TimeOutPhrase);
```

Descrição:

Tem a finalidade de configurar as frases e situações de erro de entrada de dados das funções especiais de menu

Parâmetros:

Port – Indica a porta da placa
InvalidDigitPhrase – Frase utilizada em caso de opção inválida
InvalidDigitRetries – Número de tentativas
TimeOutPhrase – Frase a ser reproduzida caso o usuário não digite nada no tempo especificado no MenuStart.

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Um dos parâmetros fora do intervalo permitido

Copyright © 2010 Digivoice Eletrônica

MenuStart

Inicia a função de menu.

Declarações:

ActiveX:

```
SHORT MenuStart(SHORT Port, LPCTSTR PlayMessage,
LPCTSTR ValidDigits, SHORT TimeOut, VARIANT_BOOL
EnablePulseDetection, SHORT PulseSensibility);
```

Descrição:

Inicia a execução de um menu. Após sua execução o método OnMenu será chamado devolvendo o dígito escolhido e o status. O parâmetro PlayMessage pode receber um arquivo de áudio literal ou "@" para executar uma lista de mensagens configurada pelas funções PlayListXXX.

Parâmetros:

Port – Indica a porta da placa
Message – Frase do menu ou "@"
ValidDigits – Dígitos considerados válidos. Devem ser colocados todos os dígitos de opções válidas sem separadores. Ex.: "235"
TimeOut – Tempo máximo para digitação da opção após a frase definida em Message.
EnablePulseDetection - Habilita(true)/Desabilita(false) detecção de pulse durante o funcionamento do Menu
PulseSensibility - Esse parâmetro permite alterar a sensibilidade de detecção de pulso, variando de -42dB até +12dB mas aconselha-se sempre passar zero, e variar o valor apenas caso haja algum problema na detecção de pulso.

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso

Copyright © 2010 Digivoice Eletrônica

DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PLAY_INVALID_FILENAME - Nome de arquivo inválido no parâmetro Message
DG_ERROR_PARAM_OUTOFRANGE - Um dos parâmetros fora do intervalo permitido
DG_ERROR_PLAY_EMPTYLIST - Lista de mensagens vazia, no caso de passar "@" no parâmetro PlayMessage

Copyright © 2010 Digivoice Eletrônica

PauseInputBuffer (dg_PauseInputBuffer)

Interrompe temporariamente o envio de amostras entre a placa e a voicerlib

Declarações:

ActiveX:
```
SHORT PauseInputBuffer(SHORT Port, SHORT Paused);
```

API:
```
short dg_PauseInputBuffer(short port, short paused);
```

Descrição:

O método PauseInputBuffer permite que o envio de amostras da placa para a aplicação fique em pausa. É útil quando o EnableInputBuffer é chamado apenas no início da aplicação e não a cada gravação, para evitar utilização de processador desnecessária, principalmente quando se utiliza as callbacks para tratar as amostras diretamente para a aplicação.

Parâmetros:

Port – Indica a porta da placa
Paused – DG_PAUSE (1) coloca em pausa e DG_RELEASE (0) retira da pausa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

PickUp (dg_PickUp)

Atende a linha conectada a placa ou atende a porta para discar.

Declarações:

ActiveX:
```
SHORT PickUp(SHORT Port, LONG PauseAfterPickup);
```

API:
```
short dg_PickUp(short port, long
pause_after_pickup);
```

Descrição:

Sempre que desejar tomar a linha conectada à placa para originar uma ligação ou para atender uma ligação entrante, é necessário fazer uso do método PickUp.

Na placa E1 se houver uma ligação entrante o método PickUp é interpretado como atendimento. Caso contrário, a porta é ocupada para realizar uma ligação sainte.

No protocolo R2D MFC quando a thread de controle E1 está habilitada, o PickUp é interpretado com um pedido de ocupação da porta e se a thread de controle estiver habilitada o comportamento é idêntico ao da placa FXO.

No protocolo ISDN PRI, se a porta especificada já tiver sido ocupada para uma ligação sainte, este método retorna o valor de retorno DG_ERROR_PORT_ALREADY_IN_USE.

A pausa após o atendimento permite ao desenvolvedor continuar os procedimentos de atendimento após um tempo especificado por este parâmetro. Neste caso o evento OnAfterPickUp é gerado após decorrido este tempo.

Copyright © 2010 Digivoice Eletrônica

Parâmetros:

Port – Indica a porta da placa
PauseAfterPickup - Pausa em milisegundos para gerar o evento OnAfterPickUp

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_EXIT_FAILURE - Ocorre caso não seja possível inserir comando na fila das threads de controle.
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Porta especificada fora do intervalo de portas configuradas
DG_ERROR_PARAM_OUTOFRANGE - PauseAfterPickUp menor que zero ou maior que 60000 ms
DG_ERROR_PORT_ALREADY_IN_USE - A porta já esta ocupada (ISDN_SET_EXCLUSIVE_CHANNEL = DG_ENABLE) e/ou não há outras portas disponíveis (ISDN_SET_EXCLUSIVE_CHANNEL = DG_DISABLE)
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1
DG_ERROR_FIFO_UNAVAILABLE - Fifo de comunicação entre threads não está disponível. Falta de memória pode ser a causa mais comum
DG_ERROR_THREAD_NOT_RUNNING - A thread não está rodando ou não está habilitada
DG_ERROR_THREAD_DISABLED - A thread ISDN está desabilitada

Copyright © 2010 Digivoice Eletrônica

PlayBuffer (dg_PlayBuffer)

Insere um vetor de amostras diretamente na placa

Declarações:

ActiveX:
```
SHORT PlayBuffer(SHORT Port, LONG Samples, SHORT
SamplesSize, LONG remaining_size);
```

API:
```
short  dg_PlayBuffer(short port, void *Samples,
short samples_size, int *remaining_size);
```

Descrição:

O PlayBuffer permite inserir diretamente na porta indicada um array de amostras de áudio a ser reproduzida pela placa. Este array pode ter até 8192 bytes, portanto o valor de samples_size não poderá ultrapassar este limite.

Também é possível passar um ponteiro para um inteiro e se este ponteiro for fornecido, a quantidade de bytes livres será retornada na parâmetro remaining_size.

A forma de utilização e as técnicas de programação deste método são discutidas no Guia do Programador, no tópico Streaming de Áudio.

(*) Não é possível enviar amostras para a placa enquanto um arquivo estiver sendo reproduzido nesta mesma porta.

Parâmetros:

Port – Indica a porta para onde as amostras serão enviadas
Samples – Vetor contendo as amostras a serem enviadas
SamplesSize - Tamanho do vetor Samples

Copyright © 2010 Digivoice Eletrônica

remaining_size - Retorna a quantidade de bytes livres no buffer

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_ALREADY_PLAYING - Uma reprodução já está em andamento
DG_ERROR_PARAM_OUTOFRANGE - samples_size maior que o tamanho do buffer (8Kbytes)

Copyright © 2010 Digivoice Eletrônica

PlayCardinal

Permite reproduzir numerais cardinais inteiros ou fracionários por extenso.

Declarações:

ActiveX:

```
SHORT PlayCardinal(SHORT Port, LPCTSTR Value, LPCTSTR
TermDigits, LONG PauseBefore);
```

Parâmetros:

Port – Indica a porta da placa
Value – Uma string ou variável contendo o número a ser reproduzido. Não deve utilizar ponto como separador de milhares e é necessário utilizar a vírgula como separador decimal
TermDigits - É uma string contendo um ou mais dígitos, que ao serem detectados finaliza a reprodução das mensagens e gera os eventos OnPlayStop e OnDigitsReceived.
PauseBefore – Indica uma pausa de n milisegundos antes de iniciar a reprodução

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

PlayCurrency

Permite reproduzir valores monetários por extenso.

Declarações:

ActiveX:

```
SHORT PlayCurrency(SHORT Port, LPCTSTR Value, LPCTSTR
TermDigits, LONG PauseBefore);
```

Parâmetros:

Port – Indica a porta da placa
Value – Uma string ou variável contendo o valor a ser reproduzido. Não deve utilizar ponto como separador de milhares e é necessário utilizar a vírgula como separador decimal
TermDigits - É uma string contendo um ou mais dígitos, que ao serem detectados finaliza a reprodução das mensagens e gera os eventos OnPlayStop e OnDigitsReceived.
PauseBefore – Indica uma pausa de n milisegundos antes de iniciar a reprodução

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

PlayDate

Permite reproduzir data por extenso

Declarações:

ActiveX:

```
SHORT PlayDate(SHORT Port, LPCTSTR Value, LPCTSTR
Mask, LPCTSTR TermDigits, LONG PauseBefore);
```

Parâmetros:

Port – Indica a porta da placa.
Value – Uma string ou variável contendo a data a ser reproduzida. Obrigatoriamente deve ser utilizado a barra "/" como separador.
Mask – Máscara utilizada que indica o formato da data. Pode assumir os seguintes valores:
d/m/y – Ex.: "25 de setembro de 2009"
d/m – Ex.: "25 de setembro"
m/d – Ex.: "Setembro 25"
m/d/y – Ex.: "Setembro 25 2009"
TermDigits - É uma string contendo um ou mais dígitos, que ao serem detectados finaliza a reprodução das mensagens e gera os eventos OnPlayStop e OnDigitsReceived.
PauseBefore – Indica uma pausa de n milisegundos antes de iniciar a reprodução

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

PlayFile (dg_PlayFile)

Inicia a reprodução de um arquivo através de uma porta da placa.

Declarações:

ActiveX:
```
SHORT PlayFile(SHORT Port, LPCTSTR  FileName,
LPCTSTR  TermDigits, LONG Origin);
```

API:
```
short  dg_PlayFile(short port, char *FileName, char
*TermDigits, long Origin);
```

Descrição:

O método PlayFile inicia a reprodução de um arquivo através da placa. É possível programar a interrupção através de um ou mais dígitos recebidos através do parâmetro TermDigits. Ao iniciar a reprodução o evento OnPlayStart é gerado e o evento OnPlayStop ao término, indicando o motivo da interrupção. É possível interromper também manualmente através do método StopRecordFile.

O PlayFile detecta automaticamente o tipo de arquivo, baseado na extensão para gsm ou se for wave, detecta se é PCM ou LeiMi. Isso permite uso de diversos formatos para reprodução sem preocupação para o programador. Se for utilizado uma extensão desconhecida, fica valendo o que está especificado no SetPlayFormat.

O método GetPlayFormat permite saber qual o formato configurado.

Parâmetros:

Port – Indica a porta da placa.

Copyright © 2010 Digivoice Eletrônica

File – String contendo o nome e caminho completo do arquivo a ser reproduzido.
TermDigits – É uma string contendo um ou mais dígitos, que ao serem detectados finaliza a reprodução das mensagens e gera os eventos OnPlayStop e OnDigitsReceived. Se qualquer dígito puder interromper utilize o símbolo "@". Se não houver dígito finalizador, passar "" (vazio).
Origin - Este parâmetro permite reproduzir a mensagem a partir de um determinado ponto. Se for passado 0 (zero) a mensagem é reproduzida do início. Se for passado **-**1 a mensagem é reproduzida a partir do final menos 2 segundos (ideal para ouvir em real-time). Qualquer número diferente de 0 e -1 significa a partir de quantos segundos a mensagem será reproduzida.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PLAY_INVALID_FILENAME - Nome de arquivo inválido
DG_ERROR_PLAY_OPENFILE - Arquivo não encontrado ou erro ao tentar abri-lo
DG_ERROR_AUDIO_FORMAT_UNSUPPORTED - Formato de áudio não suportado pela VoicerLib
DG_ERROR_COULD_NOT_CREATE_THREAD - Não foi possível inicializar thread de reprodução

Copyright © 2010 Digivoice Eletrônica

PlayList

Inicia a reprodução de uma lista de mensagens de um determinada porta

Declarações:

ActiveX:

```
SHORT PlayList(SHORT Port, LPCTSTR TermDigits);
```

Descrição:

Inicia a reprodução da lista de mensagens associada a porta indicada por Port que foi criada a partir do método PlayListAdd. O funcionamento é idêntico ao do PlayFile, sendo gerado apenas um evento OnPlayStart no início e um OnPlayStop no final da última mensagem da lista.

Parâmetros:

Port – Indica a porta da placa.
TermDigits – É uma string contendo um ou mais dígitos, que ao serem detectados finaliza a reprodução das mensagens e gera os eventos OnPlayStop e OnDigitsReceived.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_ALREADY_PLAYING - Tentativa de iniciar uma reprodução quando já existe uma em curso
DG_ERROR_PLAY_EMPTYLIST - Lista de mensagens vazia

Copyright © 2010 Digivoice Eletrônica

PlayListAdd

Permite adicionar itens a serem reproduzidos na lista da porta

Declarações:

ActiveX:

```
VARIANT_BOOL PlayListAdd(SHORT Port, SHORT ItemType,
LPCTSTR StringValue, LPCTSTR Mask, LONG
PauseBefore);
```

Parâmetros:

Port – Indica a porta da placa
ItemType – Configura o tipo de mensagem a ser reproduzida (ptFile, ptCurrency, ptCardinal, ptDate e ptTime)
Value – É a string que contém o valor a ser reproduzido, respeitando a sintaxe determinada por ItemType
Mask – Máscara utilizada somente para o tipo ptDate
PauseBefore – Indica uma pausa de n milisegundos antes de iniciar a reprodução

Valor de Retorno:

True - Adicionou mensagem com sucesso
False - Erro ao adicionar mensagem

Copyright © 2010 Digivoice Eletrônica

PlayListClear

Active X

Elimina todos os itens na lista da porta

Declarações:

ActiveX:

```
void PlayListClear(SHORT Port);
```

Descrição:

Este método deve ser chamado antes do método PlayListAdd para eliminar todos os elementos que possam estar na lista.

Parâmetros:

Port – Indica a porta da placa

Copyright © 2010 Digivoice Eletrônica

PlayListGetCount

Active X

Retorna a quantidade de elementos na lista daquele canal

Declarações:

ActiveX:

```
SHORT PlayListGetCount(SHORT Port);
```

Descrição:

Este método devolve a quantidade de elementos da lista de mensagens

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

Se sucesso, retorna um inteiro com a quantidade de elementos
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

PlayListRemoveItem

Active X

Remove o item especificado da lista de mensagens da porta

Declarações:

ActiveX:

```
VARIANT_BOOL PlayListRemoveItem(SHORT Port, SHORT
Index);
```

Descrição:

Remove o item especificado pelo parâmetro Index. Deve ser utilizado um valor entre 0 e n-1.

Parâmetros:

Port – Indica a porta da placa.
Index – Índice do elemento a ser removido

Valor de Retorno:

True se conseguiu remover e False no caso de erro.

Copyright © 2010 Digivoice Eletrônica

PlayNumber

Active X

Permite reproduzir números dígito a dígito

Declarações:

ActiveX:

```
SHORT PlayNumber(SHORT Port, LPCTSTR Value, LPCTSTR
TermDigits, LONG PauseBefore);
```

Parâmetros:

Port – Indica a porta da placa
Value – Uma string ou variável contendo o número a ser reproduzido. Não se deve utilizar ponto como separador de milhares e é necessário utilizar a vírgula como separador decimal
TermDigits - É uma string contendo um ou mais dígitos, que ao serem detectados finaliza a reprodução das mensagens e gera os eventos OnPlayStop e OnDigitsReceived. Pode falar também: barra, traço, vírgula e ponto
PauseBefore – Indica uma pausa de n milisegundos antes de iniciar a reprodução

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

PlayTime

Permite reproduzir hora por extenso

Declarações:

ActiveX:

```
SHORT PlayTime(SHORT Port, LPCTSTR Value, LPCTSTR
TermDigits, LONG PauseBefore);
```

Parâmetros:

Port – Indica a porta da placa
Value – Uma string ou variável contendo a hora a ser reproduzida. Obrigatoriamente a hora deve estar representada no formato hh:mm:ss ou hh:mm
TermDigits - É uma string contendo um ou mais dígitos, que ao serem detectados finaliza a reprodução das mensagens e gera os eventos OnPlayStop e OnDigitsReceived.
PauseBefore – Indica uma pausa de n milisegundos antes de iniciar a reprodução

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PLAY_EMPTYLIST - Lista de mensagens vazia

Copyright © 2010 Digivoice Eletrônica

PromptAbort

Interrompe a execução de uma função de PromptStart.

Declarações:

ActiveX:

```
SHORT PromptAbort(SHORT Port);
```

Descrição:

Ao chamar este método, a função iniciada pelo PromptStart será interrompida para a porta específica.

Parâmetros:

Port – Indica a porta da placa que gerou o evento

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

PromptSettings

Configura as opções de prompt

Declarações:

ActiveX:

```
SHORT PromptSettings(SHORT Port, LPCTSTR
PlaybackPhrase, LPCTSTR ConfirmationPhrase, LPCTSTR
RetryDigit, LPCTSTR ConfirmationDigit, LPCTSTR
CancelDigit);
```

Descrição:

Tem a finalidade de configurar as condições de confirmação e conferência dos dados digitados.

Parâmetros:

Port – Indica a porta da placa
PlaybackPhrase – Frase de boas vindas – ex.: "Você digitou"
ConfirmationPhrase – Frase para confirmação – ex.: "Tecle * para confirmar, # para cancelar ou 9 para digitar de novo"
RetryDigit – Dígito que indicará redigitação dos dados
ConfirmationDigit - Dígito que indicará aceitação dos dados
CancelDigit - Dígito que indicará cancelamento da entrada de dados

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

PromptStart

Inicia a função especial de entrada de dados

Declarações:

ActiveX:

```
SHORT PromptStart(SHORT Port, LPCTSTR PromptMessage,
SHORT MinDigit, SHORT MaxDigit, SHORT TimeOut, SHORT
InterdigitTimeOut, LPCTSTR TermDigits, VARIANT_BOOL
WithConfirmation, VARIANT_BOOL WithPlayback, SHORT
Retries, VARIANT_BOOL EnablePulseDetection, SHORT
PulseSensibility);
```

Descrição:

Ao chamar este método, a função de prompt é iniciada. O processo terminará gerando o evento OnPrompt que indicará o dado digitado e o Status

Parâmetros:

Port – Indica a porta da placa
Message – Mensagem ou lista de mensagens ("@") a serem reproduzidas para indicar a entrada de dados. Ex.: *"Digite sua senha..."*
MinDigit – Número mínimo de dígitos a serem esperados pelo método. Após o timeout ou dígito terminador, se o número mínimo não for alcançado, o método avisará no evento OnPrompt
MaxDigit – Número máximo de dígitos a serem esperados pelo método.
TimeOut – Timout global de espera de dígitos a ser contado após o término da mensagem.
InterdigitTimeOut – Timeout interdígito a ser considerado após a digitação do primeiro dígito.
TermDigits – Dígitos terminadores de entrada de dados. Ex.: *"Disque sua senha e # para terminar"* neste caso a # deve ser

Copyright © 2010 Digivoice Eletrônica

passada como parâmetro aqui.
WithConfirmation – Se true, Executa as funções de confirmação
WithPlayback – Se true, executa as funções de conferência, reproduzindo o que foi digitado.
Retries – Número de repetições do prompt em caso do usuário não digitar nada.
EnablePulseDetection - Habilita/Desabilita detecção de pulse durante o funcionamento do Menu
PulseSensibility - Esse parâmetro permite alterar a sensibilidade de detecção de pulso, variando de -42dB até +12dB mas aconselha-se sempre passar zero, e variar o valor apenas caso haja algum problema na detecção de pulso.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Um dos parâmetros fora do intervalo permitido
DG_ERROR_PLAY_EMPTYLIST - Lista de mensagens vazia, no caso de passar "@" no parâmetro PlayMessage.

Copyright © 2010 Digivoice Eletrônica

R2AskForGroupII (dg_R2AskForGroupII)

Envia um pedido do grupo II para a thread E1

Declarações:

ActiveX:
```
SHORT R2AskForGroupII(SHORT Port, SHORT
SignallingType);
```

API:
```
short dg_R2AskForGroupII(short port, short type);
```

Descrição:

Caso seja necessário enviar um comando de pedido de grupo II (C_PREP_RX_GRUPO_B) manualmente, diretamente para a thread de controle E1, deve ser utilizado este método. A thread de controle já deverá ter sido inicializada através do método CreateE1Thread.

Normalmente não será necessário utilizar este método, pois o método ConfigE1Thread permite que este e outros parâmetros sejam configurados previamente.

Parâmetros:

Port – Porta para onde será enviado o comando

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver não inicializado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro passado está fora do intervalo permitido
DG_ERROR_THREAD_NOT_RUNNING - Thread E1 não foi inicializada

Copyright © 2010 Digivoice Eletrônica

R2AskForID (dg_R2AskForId)

Envia um pedido da identificação do assinante para a thread E1

Declarações:

ActiveX:
```
SHORT R2AskForID(SHORT Port);
```

API:
```
short dg_R2AskForId(short port);
```

Descrição:

Caso seja necessário enviar um comando de pedido de identificação (C_ASK_FOR_ID) manualmente, diretamente para a thread de controle E1, deve ser utilizado este método. A thread de controle já deverá ter sido inicializada através do método CreateE1Thread.

Normalmente não será necessário utilizar este método, pois o método ConfigE1Thread permite que este e outros parâmetros sejam configurados previamente.

Parâmetros:

Port – Porta para onde será enviado o comando

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver não inicializado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro passado está fora do intervalo permitido
DG_ERROR_THREAD_NOT_RUNNING - Thread E1 não foi inicializada

Copyright © 2010 Digivoice Eletrônica

R2SendGroupB (dg_SendGroupB)

Envia o grupo B para a thread E1

Declarações:

ActiveX:
```
SHORT R2SendGroupB(SHORT Port, SHORT GroupBType);
```

API:
```
short dg_R2SendGroupB(short port, short
groupb_type);
```

Descrição:

Este método envia para a thread E1, o grupo B da sinalização E1, que pode assumir os valores abaixo.

Normalmente não será necessário utilizar este método, pois o método ConfigE1Thread permite que este e outros parâmetros sejam configurados previamente.

Parâmetros:

Port – Porta para onde será enviado o comando
GroupB_Type - Tipo de grupo B a ser enviado

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver não inicializado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro passado está fora do intervalo permitido
DG_ERROR_PARAM_OUTOFRANGE - Um dos parâmetros fora do intervalo permitido
DG_ERROR_THREAD_NOT_RUNNING - Thread E1 não foi inicializada

Copyright © 2010 Digivoice Eletrônica

ReadDigits (dg_ReadDigits)

Lê o conteúdo do buffer de dígitos da porta especificada

Declarações:

ActiveX:
```
BSTR ReadDigits(SHORT Port);
```

API:
```
short dg_ReadDigits(short port, char *szDigits);
```

Descrição:

O método ReadDigits permite ler o conteúdo do buffer de dígitos da porta informada. Este buffer é preenchido pelos métodos PlayFile, RecordFile ou GetDigits.

Parâmetros:

Port – Indica a porta da placa
szDigits - null-terminated - string que receberá os dígitos (somente API)

Valor de Retorno:

ActiveX:

Retorna uma string com os dígitos ou nulo caso não haja dígitos no buffer interno da biblioteca

API:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

RecordFile (dg_RecordFile)

Inicia a gravação de um arquivo de áudio através de uma porta da placa.

Declarações:

ActiveX:
```
SHORT RecordFile(SHORT Port, LPCTSTR  FileName,
LPCTSTR  TermDigits);
```

API:
```
short dg_RecordFile(short port, char *FileName, char
*TermDigits);
```

Descrição:

O método RecordFile inicia a gravação de um arquivo através da placa. É possível programar a interrupção da gravação através de um ou mais dígitos recebidos através do parâmetro TermDigits. Ao iniciar a gravação o evento OnRecordStart é gerado e o evento OnRecordStop ao término, indicando o motivo da interrupção. É possível interromper também manualmente através do método StopRecordFile. Também é possível colocar a gravação em pause, sem fechar o arquivo através do método RecordPause.

O RecordFile só poderá ser chamado após a chamada do método EnableInputBuffer, que inicia o envio de amostras de áudio da placa para a VoicerLib.

O método SetRecordFormat deve ser chamado para configurar o formato de gravação a ser utilizado. A VoicerLib não assume nenhum formato baseado apenas na extensão do arquivo. O método GetRecordFormat permite saber qual o formato configurado.

Parâmetros:

Copyright © 2010 Digivoice Eletrônica

Port – Indica a porta da placa
File – String contendo o nome e caminho completo do arquivo.
TermDigits – É uma string contendo um ou mais dígitos, que ao serem detectados finaliza a execução do GetDigits e gera o evento OnDigitsReceived. Se qualquer dígito puder interromper utilize o símbolo "@" ao invés de colocar "0123456789#*", apesar de a segunda forma também estar correta. Se não houver dígito finalizador, passar "" (vazio).

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_PLAY_OPENFILE - Não foi possível localizar ou abrir um arquivo para reprodução (ActiveX)
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_REC_OPENFILE - Erro na criação do arquivo de gravação
DG_ERROR_REC_STOPPING - Existe uma gravação sendo finalizada
DG_ERROR_REC_ALREADY_RECORDING - Já existe uma gravação em curso

Copyright © 2010 Digivoice Eletrônica

RecordPause (dg_RecordPause)

Interrompe temporariamente uma gravação, permitindo sua continuação no mesmo arquivo

Declarações:

ActiveX:
```
SHORT RecordPause(SHORT Port, VARIANT_BOOL Paused);
```

API:
```
short dg_RecordPause(short port, short paused);
```

Descrição:

O método RecordPause permite que a gravação fique em pausa. Isto facilita a programação principalmente quando o usuário quiser colocar o cliente em espera e não desejar que este período seja gravado.

Parâmetros:

Port – Indica a porta da placa
Paused – DG_PAUSE (1) coloca em pausa e DG_RELEASE (0) retira.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_REC_NOT_RECORDING - Não está gravando

Copyright © 2010 Digivoice Eletrônica

ResetError (dg_ResetError)

Envia um pedido para verificação de código de erro e zera código na placa

Declarações:

ActiveX:
```
SHORT ResetError(SHORT Card, SHORT Flag);
```

API:
```
short dg_ResetError(short card, short flag);
```

Descrição:

O firmware do hardware mantém um endereçamento de memória para arquivar os códigos de erro de hardware. Esses erros não deverão ocorrer nunca, porém a monitoração dos mesmos permitirá diagnosticar com mais facilidade problemas de hardware ou mesmo conflitos com o sistema operacional.

Parâmetros:

Card – Indica a placa que será enviado o comando
Flag – Este parâmetro pode assumir dois valores
- ONTSEND_ERROR (0): O flag com este valor simplesmente resetará a informação de erro da placa
- SEND_ERRORCODE (1): Passando-se este valor, será gerado um evento OnErrorDetected com o código de erro existente, além de resetar a informação de erro da placa.

IMPORTANTE: Nunca use este parâmetro dentro do próprio tratamento do evento para evitar reentrância infinita.

Valores de Retorno:

Copyright © 2010 Digivoice Eletrônica

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver não inicializado
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Flag não é DONTSEND_ERROR ou SEND_ERRORCODE
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro Card fora do número de placas instalado

Copyright © 2010 Digivoice Eletrônica

ResetPortResource (dg_ResetPortResource)

Recupera a configuração original de portas virtuais

Declarações:

ActiveX:
```
SHORT ResetPortResource(void);
```

API:
```
short dg_ResetPortResource(void);
```

Descrição:

A qualquer momento é possível voltar a numeração das portas para o reconhecimento automático, bastando para isso chamar o método ResetPortResource. Este método não exige nenhum parâmetro e fará com que a VoicerLib efetue o reconhecimento de portas feito na inicialização.

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado

Copyright © 2010 Digivoice Eletrônica

ReturnCodeGSMToString (dg_ReturnCodeGSMToString)

Retorna qual a mensagem de erro ocorrido na operação da thread GSM, a partir do código de retorno.

Declarações:

ActiveX:
```
void ReturnCodeGSMToString(SHORT wReturn, LPCTSTR
s);
```

API:
```
void dg_ReturnCodeGSMToString(short wReturn, char
*s);
```

Parâmetros:

wReturn - Código do erro.
s - Retorna a string referente ao codigo de erro

Valor de Retorno:

Retorna a mensagem de erro

Copyright © 2010 Digivoice Eletrônica

ReturnCodeToString (dg_ReturnCodeToString)

Retorna qual a mensagem de erro, a partir do código de retorno.

Declarações:

ActiveX:
```
void ReturnCodeToString(SHORT wReturn, LPCTSTR
s);
```

API:
```
void dg_ReturnCodeToString(short wReturn, char *s);
```

Parâmetros:

wReturn - Código do erro.
s - Retorna a string referente ao cÓdigo de erro

Valor de Retorno:

Retorna a mensagem de erro

Copyright © 2010 Digivoice Eletrônica

ReturnCodeToStringCauseBusy (dg_ReturnCodeToStringCauseBusy)

Retorna qual a descrição da causa de ocupado, a partir do código de retorno.

Declarações:

ActiveX:
```
void ReturnCodeToString(SHORT wReturn, LPCTSTR s);
```

API:
```
void dg_ReturnCodeToString(short wReturn, char *s);
```

Parâmetros:

wReturn - Código do retorno
s - Retorna a string referente ao código de retorno

Valor de Retorno:

Retorna a mensagem de descrição da causa

Veja Também: GetBusyCause.

Copyright © 2010 Digivoice Eletrônica

SendISDNCommand (dg_SendISDNCommand)

Envia um comando ISDN referente a estabelecimento de chamadas entrantes.
Só é possível enviar o comando se a configuração ISDN_AUTO_SIGNALING (ConfigISDNThread) estiver desabilitada (DG_DISABLE).

Declarações:

ActiveX:
```
SHORT SendISDNCommand(SHORT Port, SHORT Command);
```

API:
```
short dg_SendISDNCommand(short port, short command);
```

Parâmetros:

Port – Indica a porta do tronco ISDN
Command – Indica o comando ISDN a ser passado, sendo:
C_WAITFORDIGITS - Indica ao PABX ou à Central Pública que a aplicação (VoicerLib) precisa de mais dígitos para completar a chamada.
C_CALLPROCEEDING - Indica ao PABX ou à Central Pública que a aplicação (VoicerLib) está processando a chamada entrante.
C_RING - Indica ao PABX ou à Central Pública que o número discado está chamando.

Valores de Retorno:

DG_EXIT_SUCCESS - Enviou comando ISDN com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta foi passado

Copyright © 2010 Digivoice Eletrônica

com valor zero ou maior do que o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1
DG_ERROR_FIFO_UNAVAILABLE - Fifo de comunicação entre threads não está disponível. Falta de memória pode ser a causa mais comum
DG_ERROR_PORT_IS_NOT_IN_USE - A porta não está atendida
DG_ERROR_AUTO_SIGNALING_ON - É necessário que a configuração ISDN_AUTO_SIGNALING (ConfigISDNThread) esteja desabilitada (DG_DISABLE).
DG_ERROR_INVALIDPARAM - Parâmetro Command fora do intervalo permitido
DG_ERROR_THREAD_DISABLED - A thread ISDN está desabilitada

Copyright © 2010 Digivoice Eletrônica

SendR2Command (dg_SendR2Command)

Envia um comando R2 diretamente para a porta de um tronco E1

Declarações:

ActiveX:
```
SHORT SendR2Command(SHORT Port, LONG Command);
```

API:
```
short dg_SendR2Command(short port, int r2);
```

Descrição:

Este método permite enviar comandos R2 diretamente para a porta da placa. Além disso, serve para habilitar ou desabilitar a detecção dos comandos R2 de um determinada porta. Caso se esteja utilizando a thread de controle E1, não é necessário chamar este método diretamente, pois isto é feito pela própria thread.

Ao habilitar a detecção, o evento OnR2Received será gerado toda vez que for detectado um sinal R2.

Parâmetros:

Port – Indica a porta do tronco E1
Command – Indica o comando R2 a ser passado, sendo:
- R2_IDLE (0x9)
- R2_CLEAR_FOWARD (0x9)
- R2_SEIZURE (0x1)
- R2_BACKWARD_DISCONNECTION (0x1)
- R2_SEIZURE_ACK (0xd)
- R2_BILLING (0xd)
- R2_CLEAR_BACK (0xd)
- R2_ANSWERED (0x5)
- R2_BLOCKED (0xd)

Copyright © 2010 Digivoice Eletrônica

- R2_FAILURE (0xd)
- R2_ENABLEDETECTION (0x10) - Habilita a detecção
- R2_DISABLEDETECTION (0x20) - Desabilita a detecção

Valores de Retorno:

DG_EXIT_SUCCESS - Thread iniciada com sucesso
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro Port está fora do intervalo permitido
DG_ERROR_DRIVER_CLOSED - Driver não inicializado

Copyright © 2010 Digivoice Eletrônica

SetAlarmMode (dg_SetAlarmMode)

Configura o modo de notificação dos alarmes dos framers E1

Declarações:

ActiveX:
```
SHORT SetAlarmMode(SHORT Card, SHORT Mode);
```

API:
```
short dg_SetAlarmMode(short card, short mode);
```

Descrição:

A placa E1 gera eventos de alarmes indicando qualquer tipo de problema nos troncos E1. Por padrão inicial a notificação dos alarmes é manual, ou seja, a placa não envia estes alarmes para a VoicerLib e esta consequentemente não gera o evento OnE1Alarm. Se for necessário saber o status do alarme, é necessário chamar o método GetAlarmStatus.

Chamando este método com o parâmetro Mode com valor ALARM_AUTOMATIC_NOTIFY, os eventos de alarme ocorrerão automaticamente logo que ocorrerem.

Recomenda-se utilizar o modo automático mas somente depois de se configurar o sincronismo das placas.

(Os tipos de alarmes são explicados no evento OnE1Alarm).

Parâmetros:

Card – Indica a placa que será enviado o comando
Mode – Modo de notificação
- ALARM_MANUAL_NOTIFY - O evento OnE1Alarm só é gerado

Copyright © 2010 Digivoice Eletrônica

após a chamada do método GetAlarmStatus
- ALARM_AUTOMATIC_NOTIFY - O evento OnE1Alarm ocorre sempre quando um alarme for detectado

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro Card fora do intervalo permitido, excedendo o número de placas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Mode fora do intervalo permitido

Copyright © 2010 Digivoice Eletrônica

SetAudioInputCallback (dg_SetAudioInputCallback)

Configura a função CallBack que poderá receber as amostras de áudio da placa

Declarações:

ActiveX:
```
SHORT SetAudioInputCallback(LONG FunctionPointer);
```

API:
```
void dg_SetAudioInputCallback(void *ptrFunc);
```

Descrição:

Este método passa para a VoicerLib o ponteiro da função Callback que tratará as amostras de áudio recebidas da placa. Para receber estas amostras é necessário habilitar o seu envio através do método EnableInputBuffer

Parâmetros:

Ponteiro da função CallBack

Copyright © 2010 Digivoice Eletrônica

SetBearerCapability (dg_SetBearerCapability)

Configura o tipo da chamada do span.

Declarações:

ActiveX:

```
SHORT SetBearerCapability(SHORT Port, SHORT
Coding_Standard, SHORT Transfer_Capability, SHORT
Layer1_Protocol);
```

API:

```
 short dg_SetBearerCapability(short port, char
coding_standard, char transfer_capability,
                              char transfer_mode, char
transfer_rate, char layer1_protocol);
```

Descrição:

Indica particularidades da chamada, que devem ser semelhantes aquelas utilizadas pelo PABX ou pela Central Pública.
Se este método não for configurado a VoicerLib assumirá o padrão.

Parâmetros:

Port – Indica a porta da placa
Coding_Standard - Codificação padrão do protocolo ISDN PRI
Transfer_Capability - Tipo de transferência da chamada
Layer1_Protocol - Codec utiizado na chamada

API
Transfer_Mode - Modo de transmissão da chamada
Transfer_Rate - Taxa de transmissão da chamada

Copyright © 2010 Digivoice Eletrônica

Valores:

Coding_Standard - Padrão: ISDN_CODING_STANDARD

ISDN_CODING_STANDARD (0) - EURO/ITU
ISDN_CODING_ISO (1)
ISDN_CODING_NATIONAL (2)
ISDN_CODING_NETWORK (3)

Transfer_Capability - Padrão:
ISDN_BEARER_CAPABILITY_SPEECH

ISDN_BEARER_CAPABILITY_SPEECH (0)
ISDN_BEARER_CAPABILITY_UNRESTRICTED (8)
ISDN_BEARER_CAPABILITY_RESTRICTED (9)
ISDN_BEARER_CAPABILITY_AUDIO (16)
ISDN_BEARER_CAPABILITY_UNRESTRICTED_TONES (17)
ISDN_BEARER_CAPABILITY_VIDEO (24)

Layer1_Protocol - Padrão
ISDN_BEARER_USER_LAYER_1_G711_ALAW (aLaw)

ISDN_BEARER_USER_LAYER_1_G711_ULAW (2)
ISDN_BEARER_USER_LAYER_1_G711_ALAW (3)

API
Transfer_Mode - ISDN_BEARER_TRANSFER_CIRCUIT (0)
Transfer_Rate - ISDN_BEARER_RATE_64K (16)

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro fora do valor especificado
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1

Copyright © 2010 Digivoice Eletrônica

SetCallAfterAnswer

Configura as opções após a detecção de atendimento do método MakeCall

Declarações:

ActiveX:

```
SHORT SetCallAfterAnswer(SHORT Port, LPCTSTR
FileName, SHORT Pause, VARIANT_BOOL AutoHangUp);
```

Descrição:

Este método configura uma frase para ser reproduzida após a detecção de atendimento

Parâmetros:

Port – Indica a porta da placa
FileName – Indica o nome do arquivo de voz a ser reproduzido
Pause – Em milisegundos, indica a pausa após o atendimento antes de reproduzir a mensagem
AutoHangUp – Indica se, após uma discagem do tipo Flash, desliga automaticamente quando detectado o atendimento

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

SetCallAfterPickup

Active X

Configura as opções após o pickup do método MakeCall

Declarações:

ActiveX:

```
    SHORT SetCallAfterPickup(SHORT Port, LPCTSTR Digit,
SHORT PauseAfter);
```

Descrição:

Este método configura os dígitos a serem discados após o atendimento (ex. Para pegar linha externa) especificado no método MakeCall.

Parâmetros:

Port – Indica a porta da placa
Digit – Indica o(s) dígito(s) a serem discados
PauseAfter – Em milisegundos, indica a pausa após pegar linha(atendimento) dado pelo método MakeCall

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

SetCallBusyPhrase

Configura a frase a ser reproduzida em caso de ocupado

Declarações:

ActiveX:

```
SHORT SetCallBusyPhrase(SHORT Port, LPCTSTR
FileName);
```

Descrição:

Este método configura uma frase para ser reproduzida em caso de ocupado

Parâmetros:

Port – Indica a porta da placa
FileName – Indica o nome e caminho do arquivo de voz a ser reproduzido

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

SetCallBusyReturnFlash

Active X

Configura o flash em caso de ocupado

Declarações:

Delphi:
```
   SHORT SetCallBusyReturnFlash(SHORT Port, SHORT Count,
LPCTSTR Digit, SHORT PauseAfter);
```

Descrição:

Este método configura as opções de flash de retomada em caso de ocupado.

Parâmetros:

Port – Indica a porta da placa
Count – Número de flashes
Digit – Dígito após o flash (se existir)
PauseAfter – Pausa após o flash

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

SetCalledNumber (dg_SetCalledNumber)

Configura as propriedades do número a ser discado.

Declarações:

ActiveX:
```
SHORT SetCalledNumber(SHORT Port, SHORT Number_Type,
SHORT Numbering_Plan);
```

API:
```
short dg_SetCalledNumber(short port, char
number_type, char numbering_plan);
```

Parâmetros:

Port – Indica a porta da placa
Number_Type - Tipo do número
Numbering_Plan - Plano de discagem

Valores:

Number_Type - Padrão:
ISDN_PARTY_NUMBER_TYPE_SUBSCRIBER

ISDN_PARTY_NUMBER_TYPE_UNKNOWN (0)
ISDN_PARTY_NUMBER_TYPE_INTERNATIONAL (1)
ISDN_PARTY_NUMBER_TYPE_NATIONAL (2)
ISDN_PARTY_NUMBER_TYPE_NETWORK (3)
ISDN_PARTY_NUMBER_TYPE_SUBSCRIBER (4)
ISDN_PARTY_NUMBER_TYPE_ABBREVIATED (6)
ISDN_PARTY_NUMBER_TYPE_RESERVED (7)

Numbering_Plan - Padrão: ISDN_PARTY_NUMBER_PLAN_ISDN

ISDN_PARTY_NUMBER_PLAN_UNKNOWN (0)

Copyright © 2010 Digivoice Eletrônica

ISDN_PARTY_NUMBER_PLAN_ISDN (1)
ISDN_PARTY_NUMBER_PLAN_DATA (3)
ISDN_PARTY_NUMBER_PLAN_TELEX (4)
ISDN_PARTY_NUMBER_PLAN_NATIONAL (8)
ISDN_PARTY_NUMBER_PLAN_PRIVATE (9)
ISDN_PARTY_NUMBER_PLAN_RESERVED (15)

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro fora do valor especificado

Copyright © 2010 Digivoice Eletrônica

SetCallingNumber (dg_SetCallingNumber)

Configura as propriedades do número do identificador de A (BINA).

Declarações:

ActiveX:

```
SHORT SetCallingNumber(SHORT Port, SHORT
Number_Type, SHORT Numbering_Plan, SHORT
Presentation_Indicator,
                        SHORT Screening_Indicator);
```

API:

```
 short dg_SetCallingNumber(short port, char
number_type, char numbering_plan, char
presentation_indicator,
                           char Screening_indicator);
```

Parâmetros:

Port – Indica a porta da placa
Number_Type - Tipo do número
Numbering_Plan - Plano de discagem
Presentation_Indicator - Propagação da identificação (opção se vai ou não repassar o BINA)
Screening_Indicator - Apresentação da identificação (opção se vai ou não mostrar o BINA)

Valores:

Number_Type - Padrão:
ISDN_PARTY_NUMBER_TYPE_SUBSCRIBER

ISDN_PARTY_NUMBER_TYPE_UNKNOWN (0)
ISDN_PARTY_NUMBER_TYPE_INTERNATIONAL (1)

Copyright © 2010 Digivoice Eletrônica

ISDN_PARTY_NUMBER_TYPE_NATIONAL (2)
ISDN_PARTY_NUMBER_TYPE_NETWORK (3)
ISDN_PARTY_NUMBER_TYPE_SUBSCRIBER (4)
ISDN_PARTY_NUMBER_TYPE_ABBREVIATED (6)
ISDN_PARTY_NUMBER_TYPE_RESERVED (7)

Numbering_Plan - Padrão: ISDN_PARTY_NUMBER_PLAN_ISDN

ISDN_PARTY_NUMBER_PLAN_UNKNOWN (0)
ISDN_PARTY_NUMBER_PLAN_ISDN (1)
ISDN_PARTY_NUMBER_PLAN_DATA (3)
ISDN_PARTY_NUMBER_PLAN_TELEX (4)
ISDN_PARTY_NUMBER_PLAN_NATIONAL (8)
ISDN_PARTY_NUMBER_PLAN_PRIVATE (9)
ISDN_PARTY_NUMBER_PLAN_RESERVED (15)

Presentation_Indicator - Padrão:
ISDN_PARTY_NUMBER_PRESENTATION_NOT_USED

ISDN_PARTY_NUMBER_PRESENTATION_ALLOWED (0)
ISDN_PARTY_NUMBER_PRESENTATION_RESTRICTED (1)
ISDN_PARTY_NUMBER_PRESENTATION_UNAVAILABLE (2)
ISDN_PARTY_NUMBER_PRESENTATION_RESERVED (3)
ISDN_PARTY_NUMBER_PRESENTATION_NOT_USED (4)

Screening_indicator - Padrão:
ISDN_PARTY_NUMBER_SCREENING_NOT_USED

ISDN_PARTY_NUMBER_SCREENING_NOT_SCREENED (0)
ISDN_PARTY_NUMBER_SCREENING_VER_PASS(1)
ISDN_PARTY_NUMBER_SCREENING_VER_FAIL (2)
ISDN_PARTY_NUMBER_SCREENING_NETWORK(3)

ISDN_PARTY_NUMBER_SCREENING_NOT_USED (4)

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso

Copyright © 2010 Digivoice Eletrônica

DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro fora do valor especificado

SetCallFlashTime

Configura o flash geral do método MakeCall

Declarações:

ActiveX:

```
SHORT SetCallFlashTime(SHORT Port, LONG FlashTime);
```

Descrição:

Este método configura o tempo de flash a ser utilizado em todas as situações executadas pelo método MakeCall.

Parâmetros:

Port – Indica a porta da placa
FlashTime – Tempo de flash em milisegundos

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

SetCallNoAnswerPhrase

Active X

Configura a frase a ser reproduzida em caso de não atendimento

Declarações:

ActiveX:

```
SHORT SetCallNoAnswerPhrase(SHORT Port, LPCTSTR
FileName);
```

Descrição:

Este método configura uma frase para ser reproduzida em caso de não atendimento

Parâmetros:

Port – Indica a porta da placa.
FileName – Indica o nome e caminho do arquivo de voz a ser reproduzido

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

SetCallNoAnswerReturnFlash

Configura o flash em caso de não atendimento

Declarações:

ActiveX:

```
SHORT SetCallNoAnswerReturnFlash(SHORT Port, SHORT
Count, LPCTSTR Digit, SHORT PauseAfter);
```

Descrição:

Este método configura as opções de flash de retomada em caso de não atendimento

Parâmetros:

Port – Indica a porta da placa
Count – Número de flashes
Digit – Dígito após o flash (se existir)
PauseAfter – Pausa após o flash

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

SetCallNoAnswerRingCount

Configura a quantidade de rings a serem considerados como não atendimento

Declarações:

ActiveX:

```
SHORT SetCallNoAnswerRingCount(SHORT Port, SHORT
RingCount);
```

Descrição:

Este método configura o tempo de flash a ser utilizado em todas as situações executadas pelo método MakeCall

Parâmetros:

Port – Indica a porta da placa
RingCount – Número de rings

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

SetCallPauseBeforeAnalysis

Active X

Configura o tempo de pausa antes de iniciar a supervisão do método MakeCall

Declarações:

ActiveX:

```
SHORT SetCallPauseBeforeAnalysis(SHORT Port, LONG
PauseBefore);
```

Descrição:

Este método configura o tempo de pausa antes de iniciar a supervisão. Pode ser utilizado para evitar falso atendimento em algumas situações

Parâmetros:

Port – Indica a porta da placa
PauseBefore – Tempo de flash em milisegundos

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

SetCallStartFlash

Active X

Configura o flash inicial em discagens com flash

Declarações:

ActiveX:

```
SHORT SetCallStartFlash(SHORT Port, SHORT Count,
LPCTSTR Digit, SHORT PauseAfterFlash, SHORT
PauseAfterDigit);
```

Descrição:

Este método configura as opções de flash inicial. É utilizado quando o método MakeCall é chamado com o formato ctWithFlash(1).

Parâmetros:

Port – Indica a porta da placa
Count – Número de flashes
Digit – Dígito após o flash (se existir)
PauseAfterFlash – Pausa após o flash
PauseAfterDigit - Pausa após o dígito

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

SetCallWaitForDialTone

Active X

Indica se o tom de discagem deverá ser aguardado.

Declarações:

ActiveX:

```
SHORT SetCallWaitForDialTone(SHORT Port, VARIANT_BOOL
WaitDialTone);
```

Descrição:

Este método indica se o método MakeCall deverá esperar pelo tom de discagem antes de discar. Se não receber este tom, gera o evento OnAfterMakeCall indicando a situação.

Parâmetros:

Port – Indica a porta da placa
WaitForDialTone – True/False

Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

SetCardDetections (dg_SetCardDetections)

Configura o valor a ser utilizado nas detecções de tons.

Declarações:

ActiveX:
```
SHORT SetCardDetections(SHORT Card, SHORT
DetectionType, FLOAT Value1, FLOAT Value2);
```

API:
```
short dg_SetCardDetections(short card, short type,
float value1, float value2);
```

Descrição:

Este método permite configurar os valores das frequências que serão monitoradas pelo sistema. A VoicerLib já vem com valores pré-configurados que servirão para a maioria dos casos. A utilização deste método permite alterar estes valores, caso seja necessário.

É importante ressaltar que este método altera as configurações de frequências de tons para determinada placa, e não por porta.

Parâmetros:

Card – Indica a placa que será enviado o comando.
Type – Este parâmetro indica qual dos presets será alterado
- CFG_DETECT_FREQTONE1 - Pré-configurado em 425Hz
- CFG_DETECT_FREQTONE2 - Pré-configurado em 1100Hz (Fax)
- CFG_DETECT_FREQTONE3 - Pré-configurado em 2100Hz (Fax)
- CFG_DETECT_FREQTONE4 - Livre
- CFG_DETECT_FREQTONE5 - Livre
- CFG_DETECT_FREQTONE6 - Livre

Copyright © 2010 Digivoice Eletrônica

- CFG_DETECT_FREQTONE7 - Livre
- CFG_DETECT_FREQTONE8 - Livre

Value1 – Indica o valor da primeira frequência a ser configurada (em Hz)
Value2 – Indica o valor da segunda frequência a ser configurada (em Hz). Se não existir segunda frequência, utilize o valor zero.

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver não inicializado
DG_ERROR_CARD_OUT_OF_RANGE - Número da placa fora do intervalo permitido
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Type diferente dos valores mostrados anteriormente

Copyright © 2010 Digivoice Eletrônica

SetCardSyncMode (dg_SetCardSyncMode)

Configura o tipo de sincronismo que a placa funcionará

Declarações:

ActiveX:
```
SHORT SetCardSyncMode(SHORT Card, SHORT SyncMode);
```

API:
```
short  dg_SetCardSyncMode(unsigned short card,
unsigned short SyncMode);
```

Descrição:

Devido à características existentes somente nas linhas digitais (E1), o desenvolvedor deve prestar atenção às configurações de sincronismo após a inicialização. Sempre após inicializar o driver (StartVoicerLib) é necessário configurar o modo de operação e o sincronismo das placas. Caso esta configuração seja omitida, a aplicação poderá apresentar erros de sinalização, etc...

Para maiores informações, leia o tópico Configurações de Sincronismo referentes à placa E1 no Guia de Programação deste manual.

Parâmetros:

Card – Indica a placa que será enviado o comando.
SyncMode – Este parâmetro indica o tipo de sincronismo que a placa funcionará
- SYNC_INTERNAL - Placa MASTER com sincronismo interno
- SYNC_LINE_A - Placa MASTER com sincronismo externo no E1 A (Primeiro E1)
- SYNC_LINE_B - Placa MASTER com sincronismo externo no E1 B (Segundo E1, em placas de 60 canais)

Copyright © 2010 Digivoice Eletrônica

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_CARD_OUT_OF_RANGE - Card informado fora do intervalo de placas instaladas.
DG_FEATURE_NOT_SUPPORTED - A placa não precisa de configuração de sincronismo
DG_ERROR_PARAM_OUTOFRANGE - SyncMode inválido

Copyright © 2010 Digivoice Eletrônica

SetDetectionType (dg_SetDetectionType)

Habilita ou desabilita diversos tipos de detecções

Declarações:

ActiveX:
```
SHORT SetDetectionType(SHORT Port, SHORT Command,
SHORT Enable);
```

API:
```
short dg_SetDetectionType(short port, short command,
short enable);
```

Descrição:

Este é o método que permite habilitar (Enable=DG_ENABLE) ou desabilitar (Enable=DG_DISABLE) as seguintes detecções, definidas pelo parâmetro Command:

- DETECT_OFF - Desabilita todas as detecções independente do parâmetro Enable
- DETECT_DTMF - Habilita/Desabilita detecção de DTMF
- DETECT_MFT - Habilita/Desabilita detecção de MFT (MF para "trás")
- DETECT_MFF - Habilita/Desabilita detecção de MFF (MF para "frente")
- DETECT_MF - Habilita/Desabilita detecção de MF definido pelo usuário
- DETECT_TONE1 - Habilita/Desabilita detecção de tom 425Hz puro
- DETECT_TONE2 - Habilita/Desabilita detecção de tom 1100Hz puro
- DETECT_TONE3 - Habilita/Desabilita detecção de tom 2100Hz puro
- DETECT_TONE4 - Habilita/Desabilita detecção de tom puro definido pelo usuário

Copyright © 2010 Digivoice Eletrônica

- DETECT_AUDIO - Habilita/Desabilita detecção de áudio (qualquer coisa diferente de silêncio)
- DETECT_TONE5 - Habilita/Desabilita detecção de tom definido pelo usuário
- DETECT_TONE6 - Habilita/Desabilita detecção de tom definido pelo usuário
- DETECT_TONE7 - Habilita/Desabilita detecção de tom definido pelo usuário
- DETECT_TONE8 - Habilita/Desabilita detecção de tom definido pelo usuário
- DETECT_ALL_MF - Habilita/Desabilita todos os tipos de MF
- DETECT_ALL_TONE - Habilita/Desabilita todos os tipos de tons

Se for passado o comando DETECT_OFF todas as detecções desta porta são desabilitadas. Chamadas consecutivas deste método acumula as detecções, ou seja, é possível habilitar a detecção do TONE1 e TONE3 simultaneamente, por exemplo.

Para fins de supervisão de linha, recomenda-se o uso das funções de CallProgress (consulte o tópico Supervisão de Linha) por ser de implementação mais simples.

Os tons definidos pelo usuário devem ser configurados pelo método SetCardDetections**.**

Parâmetros:

Port – Indica a porta da placa
Command - Valores que indicam qual detecção habilitar, conforme lista citada anteriormente.
Enable - DG_ENABLE(1) ou DG_DISABLE(0)

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Command fora do especificado

Copyright © 2010 Digivoice Eletrônica

SetDialDelays (dg_SetDialDelays)

Configura os tempos de pausa dos símbolos de discagem

Declarações:

ActiveX:
```
SHORT SetDialDelays(USHORT CommaDelay, USHORT
DotDelay, USHORT SemicolonDelay);
```

API:
```
short dg_SetDialDelays(unsigned short CommaDelay,
unsigned short DotDelay, unsigned short
SemicolonDelay);
```

Descrição:

Quando o método Dial ou MakeCall é chamado, um dos parâmetros passados é uma string contendo uma sequência de dígitos a serem discados pela placa. Além dos dígitos, é possível passar os caracteres vírgula, ponto e ponto-e-vírgula, cada um representando um determinado tempo de pausa em milissegundos a ser respeitada durante a discagem.

Ex.: Se for discado "123,4", o sistema discará "123", dará uma pausa de x milissegundos e depois discará o "4".

Esse método unifica para a API e o ActiveX o acesso às antigas propriedades do ActiveX (DelayComma, DelayDot e DelaySemicolon) que foram retiradas desta versão.

Parâmetros:

CommaDelay – Determina o tempo em milissegundos para a pausa relativa ao caracter "," (vírgula) .
DotDelay – Determina o tempo em milissegundos para a pausa relativa ao caracter "." (ponto) .

Copyright © 2010 Digivoice Eletrônica

SemiColonDelay – Determina o tempo em milissegundos para a pausa relativa ao caracter ";" (ponto-e-vírgula) .

Valores de Retorno:

DG_EXIT_SUCCESS- Executado com sucesso

Copyright © 2010 Digivoice Eletrônica

SetDigitFrequency(dg_SetDigitFrequency)

Configura as frequências que compõem os sinais MF

Declarações:

ActiveX:

```
SHORT SetDigitFrequency(SHORT Card, SHORT Cmd, SHORT
Frequency, CHAR Digit, LONG FrequencyValue);
```

API:

```
short  dg_SetDigitFrequency(short card, short
command, short freqindex, char cDigit, int
frequency_value);
```

Descrição:

Os sinais multifrequenciais (MF) têm este nome por serem compostos de mais de uma frequência. Este método permite configurar as frequências de cada dígito separadamente, dando grande versatilidade. A mudança dessa configuração afeta todas as portas de determinada placa.

Parâmetros:

Card – Placa a ser configurada
Cmd - Indica qual o tipo de MF a ser configurado e pode ser:
- DIAL_CFG_FREQDTMF: frequencia do sinal de DTMF
- DIAL_CFG_FREQMFT: frequencia do sinal de MFT
- DIAL_CFG_FREQMFF: frequencia do sinal de MFF
- DIAL_CFG_FREQMF: frequencia do sinal de MF (= DTMF) ou configurado pelo usuário
- DIAL_CFG_FREQTOM1: frequencia do sinal do TOM1 (425 Hz)
- DIAL_CFG_FREQTOM2: frequencia do sinal do TOM2 (425 Hz)

Frequency - Indica qual das frequências será configurada (FREQ1, FREQ2 ou NONE). No caso dos xxxFREQTOM1 e

Copyright © 2010 Digivoice Eletrônica

xxxFREQTOM2 este parâmetro é ignorado
Digit - Dígito a ser configurado (0-9, #, *, A B C D)
FrequencyValue - de 200Hz a 3500Hz

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro fora do intervalo permitido
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro card fora do intervalo de placas instaladas

Copyright © 2010 Digivoice Eletrônica

SetDigitGain(dg_SetDigitGain)

Configura ganho/atenuação dos dígitos ou tons gerados

Declarações:

ActiveX:

```
SHORT SetDigitGain(SHORT Card, SHORT Cmd, FLOAT
GainValue);
```

API:

```
short dg_SetDigitGain(short card, short command,
float gain_value);
```

Descrição:

É possível dar ganho ou atenuar o sinal dos dígitos ou tons gerados pela placa. Esse tipo de configuração afeta todos as portas da placa informada em card.

O parâmetro Cmd/command determina qual tipo de MF ou tom terá seu ganho alterado. Já o parâmetro GainValue fornece o valor em dBm do ganho a ser aplicado.

Parâmetros:

Card – Indica a placa a altera
Cmd/GainValue - O comando e os valores permitidos de ganho são associados:

Copyright © 2010 Digivoice Eletrônica

Cmd/Command	Descrição	GainValue
DIAL_CFG_GAINDTMF	Ganho dos dígitos DTMFs	-3dBm ate -42 dBm
DIAL_CFG_GAINMFT	Ganho dos MF "pra trás"	-3dBm ate -42 dBm
DIAL_CFG_GAINMFF	Ganho dos MF "pra frente"	-3dBm ate -42 dBm
DIAL_CFG_GAINMF	Ganho dos MF definidos pelo usuário	-3dBm ate -42 dBm
DIAL_CFG_GAINTONE1	Ganho dos tons definidos pelo usuário	+3.17dBm até -42dBm
DIAL_CFG_GAINTONE2	Ganho dos tons definidos pelo usuário	+3.17dBm até -42dBm

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro card fora do intervalo de placas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro fora do intervalo permitido (Cmd ou GainValue)

Copyright © 2010 Digivoice Eletrônica

SetDTMFConfig (dg_SetDTMFConfig)

Determina a duração do tom DTMF gerado e da pausa entre dígitos

Declarações:

ActiveX:
```
SHORT SetDTMFConfig(LONG Duration, LONG Pause);
```

API:
```
short dg_SetDtmfConfig(long duration, long pause);
```

Descrição:

O tom tem duração de 90ms como padrão, mas poderá ser alterado conforme a necessidade. A pausa também tem valor padrão em 90ms.

Quando um novo valor é atribuído, todas as portas de todas as placas são afetadas.

Parâmetros:

Duration – Duração do dtmf em milissegundos
Pause - Duração da pausa entre dígitos, em milissegundos

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PARAM_OUTOFRANGE - Duration ou Pause maior que 60000ms

Copyright © 2010 Digivoice Eletrônica

SetE1CRC4Option (dg_SetE1CRC4Option)

Habilita ou desabilita opção de CRC4 no framer E1

Declarações:

ActiveX:

```
SHORT SetE1CRC4Option(SHORT Card, SHORT E1Device,
SHORT Enable);
```

API:

```
short dg_SetE1CRC4Option(short card, short e1, short
enable);
```

Descrição:

Esta opção permite habilitar ou desabilitar a opção de CRC4 do framer E1, para eventualmente compatibilizar com a central pública. Não é comum ter que utilizar esta opção.

Parâmetros:

Card - Indica qual a placa na qual será configurado este parâmetro
E1Device - Pode assumir os valores CFG_FRAMER_E1_B(2) ou CFG_FRAMER_E1_A(1)
Enable – DG_ENABLE (1) ou DG_DISABLE(0)

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro fora do intervalo permitido
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro card fora do intervalo de placas instaladas

Copyright © 2010 Digivoice Eletrônica

SetFastDetection (dg_SetFastDetection)

Habilita ou desabilita o algoritmo de detecção rápida de MF

Declarações:

ActiveX:
```
SHORT SetFastDetection(SHORT Port, SHORT Enable);
```

API:
```
short WCDECL dg_SetFastDetection(short port, short
enable);
```

Descrição:

Este método habilita ou desabilita o algoritmo de detecção rápida de MF, permitindo sua identificação em 20ms. Normalmente, o modo de detecção rápida não deve ser utilizado pois em aplicações normais a duração dos MFs é bem maior que 20ms e este algoritmo consome alto processamento do DSP da placa.

Devido a este alto consumo, não é possível utilizar o modo rápido em conjunto com gravação gsm, por exemplo.

A detecção propriamente dita continua a ser habilitada pelo método SetDetectionType. O método SetFastDetection, caso seja necessário, deverá ser chamado antes de se habilitar as detecções.

Parâmetros:

Port - Porta da placa que será aplicada a configuração
Enable - DG_ENABLE(1) ou DG_DISABLE(0)

Valor de Retorno:

Copyright © 2010 Digivoice Eletrônica

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Enable diferente dos valores mostrados anteriormente
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

SetFaxFrequencies (dg_SetFaxFrequencies)

Modificar a frequência padrão para detecção de fax

Declarações:

ActiveX:
```
SHORT SetFaxFrequencies(SHORT First, SHORT Second);
```

API:
```
short dg_SetFaxFrequencies(short first, short
second);
```

Descrição:

Esta frequência deve ser alterada caso seja necessário detectar alguma frequência fora de padrão. Neste caso o fax não será mais detectado. As frequências padrão de fax são 1100Hz e 2100hz.

Este método chama, na verdade, os seguintes comandos:

```
dg_SetCardDetections(card, CFG_DETECT_FREQTONE2,
first,0);
dg_SetCardDetections(card, CFG_DETECT_FREQTONE3,
second,0);
```

Para todas as placas instaladas.

Caso seja necessário tratar essas frequências de maneira diferente por placa, utilizar o dg_SetCardDetections diretamente.

Parâmetros:

Card - Indica qual a placa na qual será configurado este parâmetro
Frequency – Valor da frequência em Hz

Copyright © 2010 Digivoice Eletrônica

Valor de Retorno:

DG_EXIT_SUCCESS - Thread iniciada com sucesso
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro diferente dos valores mostrados anteriormente

Copyright © 2010 Digivoice Eletrônica

SetFramerLoop (dg_SetFramerLoop)

Coloca os framers em loop para fins de testes ou monitoração

Declarações:

ActiveX:
```
SHORT SetFramerLoop(SHORT Card, SHORT Device, SHORT
LoopType);
```

API:
```
short dg_SetFramerLoop(short card, short device,
short loop_type);
```

Descrição:

Os framers são os troncos E1s das placas digitais. Este método permite fechar um looping local ou remoto entre o TX e o RX de cada E1 para fins de testes ou mesmo de monitoração.

Em operação normal, os framers estão configurados como FRAMER_LOOP_OFF(0). Caso seja necessário efetuar um teste , por exemplo, é possível modificá-lo para FRAMER_LOOP_REMOTE(1).

OBS: A opção LOOP_REMOTE(1) também é útil na operação de gravação em paralelo quando não se quiser utilizar o conector "T".

Parâmetros:

Card - Indica qual a placa na qual será configurado este parâmetro
Device - Indica qual dos troncos E1 será configurado. Pode assumir E1_A(1) ou E1_B(2)
Loop_Type – Tipo de looping:
- FRAMER_LOOP_OFF (0) - Sem looping - Opção padrão e

Copyright © 2010 Digivoice Eletrônica

utilizada em operações normais
- FRAMER_LOOP_REMOTE (1) - Looping remoto - Testes de linha ou gravação em paralelo
- FRAMER_LOOP_LOCAL (2) - Looping local - Utilizado somente em testes de assistência técnica

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro card fora do intervalo de placas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro fora do intervalo permitido (device ou loop_type)

Copyright © 2010 Digivoice Eletrônica

SetFrequency (dg_SetFrequency)

Modificar a frequência padrão de supervisão do tom CFG_DETECT_FREQTONE1

Declarações:

ActiveX:
```
SHORT SetFrequency(SHORT Card, SHORT Frequency);
```

API:
```
short dg_SetFrequency(short card, short Frequency);
```

Descrição:

A frequência base para detecção dos tons de chamada, ocupado, etc... é 425hz. Algumas centrais trabalham com valores diferentes do padrão. Alterando esta frequência as placas Digivoice passarão a monitorar outra faixa de tons. Ao atribuir este valor, todas as portas de todas as placas são afetados.

Este método é mantido para fins de compatibilidade e para facilitar a mudança dessa frequência, que é a mais utilizada. Internamente ele faz chamada ao método SetCardDetections(card, CFG_DETECT_FREQTONE1, Frequency, 0)

Parâmetros:

Card - Indica qual a placa na qual será configurado este parâmetro
Frequency – Valor da frequência. O padrão é 425hz

Valor de Retorno:

Copyright © 2010 Digivoice Eletrônica

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro card fora do intervalo permitido, excedendo o número de placas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Valor da frequência fora do intervalo entre 0 e 3000hz

Copyright © 2010 Digivoice Eletrônica

SetFXCardType (dg_SetFXCardType)

API | Active X

Configura o tipo da porta para a placa VB0404FX

Declarações:

ActiveX:
```
SHORT SetFXCardType(SHORT Port, SHORT Type);
```

API:
```
short dg_SetFXCardType(short Port, short Type);
```

Parâmetros:

Port - Indica a porta da placa
Type - O tipo que a porta será configurada: 0 (DG_FX_TYPE_NONE)
1 (DG_FX_TYPE_FXS)
2 (DG_FX_TYPE_FXO)

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro fora do intervalo permitido

Copyright © 2010 Digivoice Eletrônica

SetGSMMode (dg_SetGSMMode)

Modificar o formato padrão dos arquivos GSM

Declarações:

ActiveX:
```
SHORT SetGSMMode(SHORT Mode);
```

API:
```
short dg_SetFrequency(short mode);
```

Descrição:

Este método configura a VoicerLib se deverá ser utilizado o GSM compatível com o Asterisk(c) (GSM_RAW) ou o GSM padrão da Digivoice (GSM_DIGIVOICE). A diferença entre os dois formatos consiste apenas na existência de um cabeçalho no padrão Digivoice. A codificação em si é a mesma e por isso não foi criado um novo formato de gravação. Este método afeta todas as portas de todas as placas e o padrão assumido atualmente é o GSM_RAW que não contém cabeçalho no arquivo.

Este método, se for necessário, pode ser chamado logo após iniciar a VoicerLib(StartVoicerLib)

Parâmetros:

Mode - GSM_RAW(1) ou GSM_DIGIVOICE(0)

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

SetHangUpCause (dg_SetHangUpCause)

API | Active X

Configura a causa do desligamento da porta no protocolo ISDN PRI.

Declarações:

ActiveX:
```
SHORT SetHangUpCause(SHORT Port, SHORT Cause);
```

API:
```
short dg_SetHangUpCause(short port, short cause);
```

Descrição:

Antes de excutar o comando HangUp é possível informar a causa do desligamento.

Este método não deve ser executado quando receber o evento OnBusyDetected.

Parâmetros:

Port – Indica a porta da placa
Cause - Causa do desligamento

Causas Normais

- ISDN_CAUSE_UNALLOCATED_NUMBER (1)
- ISDN_CAUSE_NO_ROUTE_SPECIFIC_TRANSIT_NETWORK (2)
- ISDN_CAUSE_NO_ROUTE_DESTINATION (3)
- ISDN_CAUSE_SEND_SPECIAL_INFORMATION_TONE (4)
- ISDN_CAUSE_MISDIALLED_TRUNK_PREFIX (5)

Copyright © 2010 Digivoice Eletrônica

- ISDN_CAUSE_CHANNEL_UNACCEPTABLE (6)
- ISDN_CAUSE_CALL_AWARDED (7)
- ISDN_CAUSE_PREEMPTION (8)
- ISDN_CAUSE_PREEMPTION_CIRCUIT_RESERVED (9)
- ISDN_CAUSE_NORMAL_CALL_CLEARING (16)
- ISDN_CAUSE_USER_BUSY (17)
- ISDN_CAUSE_NO_USER_RESPONDING (18)
- ISDN_CAUSE_NO_ANSWER_FROM_USER (19)
- ISDN_CAUSE_SUBSCRIBER_ABSENT (20)
- ISDN_CAUSE_CALL_REJECTED (21)
- ISDN_CAUSE_NUMBER_CHANGED (22)
- ISDN_CAUSE_REDIRECTION_TO_NEW_DESTINATION (23)
- ISDN_CAUSE_EXCHANGE_ROUTING_ERROR (25)
- ISDN_CAUSE_NON_SELECTED_USER_CLEARING (26)
- ISDN_CAUSE_DESTINATION_OUT_OF_ORDER (27)
- ISDN_CAUSE_INVALID_NUMBER_FORMAT (28)
- ISDN_CAUSE_FACILITY_REJECTED (29)
- ISDN_CAUSE_STATUS_ENQUIRY (30)
- ISDN_CAUSE_NORMAL_UNSPECIFIED (31)

Valor de Retorno:

DG_EXIT_SUCCESS - Setou a causa do desligamento com sucesso
DG_ERROR_DRIVER_CLOSED - O driver não foi iniciado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta foi passado com valor zero ou maior do que o número de portas instaladas
DG_ERROR_PORT_IS_NOT_IN_USE - A porta não está atendida
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1

Veja Também: HangUp.

Copyright © 2010 Digivoice Eletrônica

SetLoggerSilenceThreshold (dg_SetLoggerSilenceThreshold)

Permite setar o limiar de silêncio para as portas da placa VB6060 no caso de aplicações utilizando o thread de Logger

Declarações:

ActiveX:
```
SHORT SetLoggerSilenceThreshold(SHORT Port, SHORT
silence);
```

API:
```
short dg_SetLoggerSilenceThreshold(short port, short
silence);
```

Descrição:

O limiar de silêncio define, de modo simplificado, como a VoicerLib tratará os dígitos a serem detectados (MFF e MFT). O parâmetro value pode variar de +12 até -42 dBm. O valor padrão é -30 dBm mas este valor pode ser alterado. Quanto mais próximo de zero, menos sensível será a detecção.

Parâmetros:

Port – Indica a porta da placa
silence - Valor em dBm, variando de -42 até +12

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

DG_ERROR_THREAD_NOT_RUNNING - A thread não foi inicializada ainda
DG_ERROR_INVALIDPARAM - Parâmetro fora do intervalo permitido (-42 < Value < +12)

Copyright © 2010 Digivoice Eletrônica

SetNameID (dg_SetNameID)

Especifica o nome da identificação de uma porta no tronco digital quando se utiliza protocolo ISDN PRI.

Declarações:

ActiveX:
```
SHORT SetNameID(SHORT Port, LPCTSTR Name);
```

API:
```
short dg_SetNameID(short port, char *szName);
```

Descrição:

Ao iniciar uma chamada, o último nome de identificação configurado será passado adiante e será mantido para todas as chamadas.

Parâmetros:

Port – Indica a porta da placa
Name - Nome identificador que será passado na chamada

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1

Copyright © 2010 Digivoice Eletrônica

SetNextE1RxCount (dg_SetNextE1RxCount)

Configura os próximos dígitos a serem recebidos na thread E1

Declarações:

ActiveX:
```
SHORT SetNextE1RxCount(SHORT Port, SHORT Count);
```

API:
```
short  dg_SetNextE1RxCount(short port, short count);
```

Descrição:

Complementar à SetStartE1RxCount, este método é destinado a permitir o controle "manual" da troca de sinalização R2D acontecendo em uma determinada porta. Este método configura os próximos dígitos que a thread E1 receberá antes de gerar o evento OnE1StateChange com status C_NUMBER_RECEIVED.

Utilize apenas este método caso seja necessário o tipo controle explicado. Caso seja utilizada uma sinalização tradicional, dê preferências às configurações do método ConfigE1Thread.

Parâmetros:

Port – Indica a porta da placa
Count - Número de dígitos a serem esperados

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro count fora do

Copyright © 2010 Digivoice Eletrônica

intervalo permitido (0 < count < 30)
DG_ERROR_THREAD_NOT_RUNNING - Thread E1 não foi iniciada pelo CreateE1Thread

Copyright © 2010 Digivoice Eletrônica

SetPlayFormat (dg_SetPlayFormat)

Especifica o formato de reprodução de uma porta.

Declarações:

ActiveX:
```
SHORT SetPlayFormat(SHORT Port, SHORT FileFormat);
```

API:
```
short dg_SetPlayFormat(short port, enum
EnumFileFormat file_format);
```

Descrição:

O método SetPlayFormat permite indicar formatos de reprodução diferentes por canal, os outros canais assumirão o formato especificado na propriedade FileFormat, que determina o formato para todos os canais. Como o novo formato GSM oferece uma qualidade de áudio inferior ao Wave pode ser necessário gravar em GSM e reproduzir mensagens no formato Wave. Com este método é possível manter formatos de reprodução diferentes no mesma porta ou em canais distintos.

O PlayFile detecta automaticamente o tipo de arquivo, baseado na extensão wave, detecta se é PCM ou LeiMi. Isso permite uso de diversos formatos para reprodução sem preocupação para o programador. Se for utilizado uma extensão desconhecida, fica valendo o que está especificado no SetPlayFormat.

Parâmetros:

Port – Indica a porta da placa.
FileFormat – Formato para reprodução.
0 - ffWaveULaw (Lei mi)
1 - ffSig - obsoleto - é forçado formato ffWaveULaw
2 - ffWavePCM

Copyright © 2010 Digivoice Eletrônica

3 - ffGsm610
4 - ffWaveALaw (Lei A)
5 - ffWave49

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

SetPortChatLog(dg_SetPortChatLog)

Conecta uma conferência a uma porta específica para fins de gravação

Declarações:

ActiveX:
```
SHORT SetPortChatLog(SHORT Port, SHORT ChatRoom,
SHORT Enable);
```

API:
```
short dg_SetPortChatLog(short port , long ChatRoom,
short enable);
```

Descrição:

Este método deverá ser utilizado quando deseja gravar uma sala de conferência (chat room) que pode ser uma conferência de apenas 2 canais (utilizada na gravação em paralelo) ou uma com várias portas conectadas.

Quando uma conferência (chat room) é criada através do método CreateChatRoom ele recebe um handle que é um identificador único para esta sala. Este handle é o que deverá passado no parâmetro ChatRoom. A eventual gravação da conferência indicada por ChatRoom deverá ser iniciada na porta port que é a porta que ouve todas as outras da sala de conferência. Port não pode fazer parte da conferência.

O parâmetro Enable pode receber DG_ENABLE(1), para ligar a porta à sala ou DG_DISABLE(0) para desligá-la.

Parâmetros:

Port – Indica a porta da placa que gravará a conferência.
ChatRoom – Indicador da sala de conferência criada por

Copyright © 2010 Digivoice Eletrônica

CreateChatRoom
Enable – DG_ENABLE ou DG_DISABLE

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_EXIT_FAILURE - Erro ao tentar executar o comando

Copyright © 2010 Digivoice Eletrônica

SetPortGain (dg_SetPortGain)

Especifica o ganho para a uma determinada porta

Declarações:

ActiveX:

```
SHORT SetPortGain(SHORT Port, SHORT txrx, SHORT
Value);
```

API:

```
short dg_SetPortGain(short port, short rx_tx, short
value);
```

Descrição:

Este método permite dar ganho/atenuação em uma determinada porta, no RX e TX separadamente. Com isso, é possível controlar a qualidade do áudio que se "escuta" ou o que se "fala". O padrão é ganho zero. Cada 6dB a menos, equivale a metade de ganho e a cada 6dB a mais equivale ao dobro.

Parâmetros:

Port – Indica a porta da placa
rx_tx – Indica se o valor será alterado no RX ou no TX. Para isso pode-se utilizar as constantes TX_GAIN (0) ou RX_GAIN (1)
Value – Valor de ganho a ser passado no método, podendo variar de -42 dBm até +12 dBm

Valores de Retorno:

EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetros port fora do intervalo especificado
DG_ERROR_PARAM_OUTOFRANGE - Este erro pode ocorrer caso

Copyright © 2010 Digivoice Eletrônica

se especifique Value maior que 12 ou menor que -42, ou parâmetro txrx diferente do previsto

Copyright © 2010 Digivoice Eletrônica

SetPortID (dg_SetPortID)

Especifica a identificação de uma porta no tronco digital quando se utiliza uma das threads de controle.

Declarações:

ActiveX:
```
SHORT SetPortID(SHORT Port, LPCTSTR ID);
```

API:
```
short dg_SetPortId(short port, char *szID);
```

Descrição:

Este método permite identificar cada porta um número de identificação diferente a ser fornecido ao destino como BINA.

Ao iniciar uma chamada, a última identificação de porta configurada será passada adiante e será mantida para todas as chamadas.

Parâmetros:

Port – Indica a porta da placa
ID – String contendo o número de identificação da porta

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1

Copyright © 2010 Digivoice Eletrônica

SetRecordFormat (dg_SetRecordFormat)

Especifica o formato de gravação de uma porta

Declarações:

ActiveX:
```
SHORT SetRecordFormat(SHORT Port, SHORT FileFormat);
```

API:
```
short  dg_SetRecordFormat(short port, enum
EnumFileFormat file_format);
```

Descrição:

O método SetRecordFormat permite indicar formatos de gravação diferentes por canal, os outros canais assumirão o formato especificado na propriedade FileFormat, que determina o formato para todos os canais. Como o novo formato GSM oferece uma qualidade de áudio inferior ao Wave pode ser necessário gravar em GSM e reproduzir mensagens no formato Wave. Com este método é possível manter formatos de gravação diferentes no mesma porta ou em portas distintos.

É possível alterar o formato de gravação sem a necessidade de chamar o método DisableInputBuffer porém não é possível fazê-lo com uma gravação em curso

Parâmetros:

Port – Indica a porta da placa
FileFormat – Formato para gravação
0 - ffWave
1 - ffSig obsoleto - é forçado formato ffWaveULaw
2 - ffWavePCM
3 - ffGsm610

Copyright © 2010 Digivoice Eletrônica

4 - ffWaveALaw (Lei A)
5 - ffWave49

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_REC_ALREADY_RECORDING - Operação não permitida pois a porta está gravando

Copyright © 2010 Digivoice Eletrônica

SetReverseCharging (dg_SetReverseCharging)

Solicita a inversão de cobrança para chamada sainte.

Declarações:

ActiveX:
```
SHORT SetReverseCharging(SHORT Port, SHORT
Reverse_Charging);
```

API:
```
short dg_SetReverseCharging(short port, short
reverse_charging);
```

Descrição:

Ao iniciar uma chamada, a última configuração será passada adiante e será mantida para todas as chamadas.

Parâmetros:

Port – Indica a porta da placa
Reverse_Charging - Desabilita (DG_DISABLE (0)) ou habilita a solicitação (DG_ENABLE (1))

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1

Copyright © 2010 Digivoice Eletrônica

SetRecordGain (dg_SetRecordGain)

Especifica o ganho para a gravação

Declarações:

ActiveX:
```
SHORT SetRecordGain(SHORT Port, SHORT Gain);
```

API:
```
short dg_SetRecordGain(short port, short gain);
```

Descrição:

Especifica o ganho para a gravação de uma porta independente. O ganho de gravação aqui pode variar de -42dB até +12dB. Nesta opção, o método é apenas uma interface para o método SetPortGain, aplicando ganho apenas no RX. O padrão do ganho é zero.

Como o ganho interferirá na qualidade do áudio, a checagem de intervalo faz com que um Gain menor que -40dB seja utilizado -40dB. E Gain maior que +12dB é fixado em +12dB.

Parâmetros:

Port – Indica a porta da placa
Ganho – Valores conforme o tipo da placa (-40 até +12)

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetros fora do intervalo especificado

Copyright © 2010 Digivoice Eletrônica

SetSilenceThreshold (dg_SetSilenceThreshold)

Permite definir o limiar de silêncio relacionado às detecções de dígitos

Declarações:

ActiveX:

```
SHORT SetSilenceThreshold(SHORT Port, SHORT Value);
```

API:

```
short  dg_SetSilenceThreshold(short port, short
value);
```

Descrição:

O limiar de silêncio define, de modo simplificado, como a VoicerLib tratará os dígitos a serem detectados (DTMF, MFF, MFT, etc...). O parâmetro value pode variar de 0 até -42 dBm. O valor padrão é -30 dBm mas este valor pode ser alterado. Quanto mais próximo de zero, menos sensível será a detecção.

Este limiar de silêncio tem resolução de 3dB, portanto, -42dB e -39dB tem o mesmo efeito na placa.

Parâmetros:

Port – Indica a porta da placa
Value - Valor em dBm, variando de 0 até -42

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado

Copyright © 2010 Digivoice Eletrônica

DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas

Copyright © 2010 Digivoice Eletrônica

SetStartE1RxCount (dg_SetStartE1RxCount)

Configura o número inicial de dígitos a serem recebidos na thread E1

Declarações:

ActiveX:
```
SHORT SetStartE1RxCount(SHORT Port, SHORT Count);
```

API:
```
short dg_SetStartE1RxCount(short port, short count);
```

Descrição:

Este método é destinado a permitir o controle "manual" da troca de sinalização R2D acontecendo em uma determinada porta. Muitas vezes uma aplicação precisa ir analisando os dígitos recebidos um a um, permitindo algum tipo de encaminhamento específico, analise de rotas, etc... Ela configura o número inicial de dígitos que a thread E1 receberá antes de gerar o evento OnE1StateChange com status C_NUMBER_RECEIVED.

Utilize apenas este método caso seja necessário o tipo controle explicado. Caso seja utilizada uma sinalização tradicional, dê preferências às configurações do método ConfigE1Thread.

Parâmetros:

Port – Indica a porta da placa
Count - Número de dígitos a serem esperados

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado

Copyright © 2010 Digivoice Eletrônica

DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro count fora do intervalo permitido (0 < count < 30)

Copyright © 2010 Digivoice Eletrônica

SetText (dg_SetText)

Especifica uma mensagem de texto a ser enviada pela VoicerLib por uma porta no tronco digital no protocolo ISDN PRI.

Declarações:

ActiveX:
```
SHORT SetText(SHORT Port, LPCTSTR Text);
```

API:
```
short dg_SetText(short port, char * szText);
```

Descrição:

A mensagem não pode exceder 128 caracteres e deve ser finalizada com '\0'. A mensagem poderá conter CR (0xd) ou LF (0xa).
As mensagens devem ser compostas de caracteres ANSI (tabela ASCII Normal), **não** utilizar caracteres especiais nem acentuados.
Utilize este método somente antes da execução do método Dial.

Ao iniciar uma chamada, a última mensagem de texto configurada será passada adiante e será mantida para todas as chamadas.

Parâmetros:

Port – Indica a porta da placa
Text - Mensagem de texto a ser enviada

Valor de Retorno:

DG_EXIT_SUCCESS: Executado com sucesso
DG_ERROR_DRIVER_CLOSED: O driver não foi iniciado

Copyright © 2010 Digivoice Eletrônica

DG_ERROR_PORT_OUT_OF_RANGE: Parâmetro porta foi passado com valor zero ou maior do que o número de portas instaladas
DG_FEATURE_NOT_SUPPORTED - Comando inválido para o protocolo ISDN PRI ou a placa instalada não é de interface E1

SetTwist (dg_SetTwist)

Configura os níveis de rigor para a detecção de dígitos

Declarações:

ActiveX:

```
SHORT SetTwist(SHORT Port, SHORT Twist1, SHORT
Twist2);
```

API:

```
short dg_SetTwist(short port, short twist1, short
twist2);
```

Descrição:

A VoicerLib procura, como padrão, manter uma boa relação entre uma boa detecção de dígitos sem, no entanto, pegar talkoffs (falsos dígitos). Isso é feito através de cálculos entre as frequências encontradas nos DTMFs.

O Twist1 é a diferença máxima aceita entre a primeira e segunda frequências tem valor padrão de 9dB. Já o Twist2 é a diferença mínima das duas primeiras frequências com uma eventual terceira frequência e tem valor padrão de 15dB.

No Twist1, quanto maior o valor, menor é o rigor na detecção. No Twist2, quanto maior o valor, maior é o rigor.

Parâmetros:

Copyright © 2010 Digivoice Eletrônica

Port – Indica a porta da placa
Twist1 - Diferença em dB entre a primeira e segunda frequência de DTMF -> maior valor, *menor* rigor
Twist2 - Diferença entre as primeiras duas frequências e uma terceira -> maior valor, *mais* rigor

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro porta fora do intervalo permitido, excedendo o número de portas instaladas
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro fora do intervalo permitido

Copyright © 2010 Digivoice Eletrônica

ShutdownVoicerLib (dg_ShutdownVoicerLib)

Finaliza a comunicação da placa com a aplicação

Declarações:

ActiveX:
```
SHORT ShutdownVoicerLib(void);
```

API:
```
short dg_ShutdownVoicerLib(void);
```

Descrição:

Este método faz com que se finalize a comunicação da placa com a aplicação. Logo, a sua chamada deverá ser a última coisa que o programador deverá fazer antes de sair da aplicação. Caso a aplicação seja encerrada sem que o este método tenha sido chamado, o micro corre o risco de "travar" a qualquer momento pois recursos de memória e interrupção ficam ativos. Então, NUNCA FECHE A APLICAÇÃO ANTES DE CHAMAR ESTE MÉTODO.

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - A VoicerLib já está finalizada (não é necessariamente um erro, pode ser simplesmente ignorado)

Copyright © 2010 Digivoice Eletrônica

StartVoicerLib (dg_StartVoicerLib)

Inicializa a comunicação da placa com a aplicação.

Declarações:

ActiveX:
```
SHORT StartVoicerLib(void);
```

API:
```
short dg_StartVoicerlib(char *szConfigPath);
```

Descrição:

Este método faz com que se inicie a comunicação da placa com a aplicação. A sua chamada deverá ser a primeira coisa que o programador deverá fazer na aplicação.

No ActiveX, pode-se preencher a propriedade ConfigPath com o caminho dos arquivos de configuração com extensão .i00 . Caso esta propriedade esteja vazia a VoicerLib assumirá como padrão o diretório \%PROGRAM FILES%\VoicerLib4. A mesma regra se aplica ao parâmentro szConfigPath na API.

Sempre deverá ser testado os valores de retorno para poder diagnosticar algum problema no caso do valor ser diferente de DG_EXIT_SUCCESS.

Valores de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_MEMORY_ALLOCATION - Memória insuficiente para iniciar os serviços
DG_ERROR_DRIVER_ALREADY_OPEN - Driver já está inicializado
DG_ERROR_READING_PORTCOUNT - O número de portas lido é

Copyright © 2010 Digivoice Eletrônica

inconsistente, muito provavelmente causado por problemas de I/O no barramento PCI
DG_ERROR_LOADING_DEVICEDRIVER - Erro ao iniciar o device driver. Só acontece no caso de uma instalação incompleta ou corrompida.
DG_ERROR_CREATING_EVENT - Erro ao criar eventos de controle da VoicerLib
DG_ERROR_COULD_NOT_CREATE_THREAD - Erro ao criar threads de controle.
DG_ERROR_MAXCARDS - Falha na recuperação do número de placas instaladas

Veja Também: ShutdownVoicerLib

Copyright © 2010 Digivoice Eletrônica

StopPlayBuffer (dg_StopPlayBuffer)

Interrompe a reprodução iniciada pelo PlayBuffer

Declarações:

ActiveX:
```
SHORT StopPlayBuffer(SHORT Port);
```

API:
```
short dg_StopPlayBuffer(short port);
```

Descrição:

Este método envia para a placa um comando dizendo para que a reprodução dos seus buffers internos seja interrompida. Deve ser chamado sempre quando não há mais amostras a serem enviadas pelo PlayBuffer.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_ALREADY_PLAYING - A porta está ocupada por uma reprodução de arquivo
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro fora do intervalo permitido

Copyright © 2010 Digivoice Eletrônica

StopPlayFile (dg_StopPlayFile)

Interrompe a reprodução de um arquivo de áudio.

Declarações:

ActiveX:
```
SHORT StopPlayFile(SHORT Port);
```

API:
```
short dg_StopPlayFile(short port);
```

Descrição:

Este método interrompe a reprodução de um arquivo de áudio. Neste caso, o evento OnPlayStop é gerado e o parâmetro Status receberá o valor ssStopped.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_NOT_PLAYING - A porta não está reproduzindo nada
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro fora do intervalo permitido

Copyright © 2010 Digivoice Eletrônica

StopRecordFile (dg_StopRecordFile)

Interrompe a gravação de um arquivo.

Declarações:

ActiveX:
```
SHORT StopRecordFile(SHORT Port);
```

API:
```
short dg_StopRecordFile(short port);
```

Descrição:

Este método interrompe a gravação de um arquivo sem cancelar o envio de amostras para a VoicerLib.

Chamando este método, o evento OnRecordStop é gerado e o parâmetro Status receberá o valor ssStopped.

Para cancelar o envio de amostras, o método DisableInputBuffer deve ser chamado.

Parâmetros:

Port – Indica a porta da placa

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro fora do intervalo permitido
DG_ERROR_THREAD_NOT_RUNNING - O envio de amostras da placa não foi habilitado e, portanto não há gravação para finalizar

Copyright © 2010 Digivoice Eletrônica

TxRxMixEnable (dg_TxRxMixEnable)

Possibilita gravação do TX e RX. Apesar de funcionar em qualquer placa com interfaces analógicas ou digitais, foi desenvolvida para uso em URAs desenvolvidas com placas de interfaces digitais.

Declarações:

ActiveX:
```
SHORT TxRxMixEnable(SHORT Port, SHORT Enable);
```

API:
```
void TxRxMixEnable(short port, short enable);
```

Parâmetros:

Port – Indica a porta da placa
Enable - DG_ENABLE ou DG_DISABLE

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_DRIVER_CLOSED - Driver desabilitado
DG_ERROR_PORT_OUT_OF_RANGE - Parâmetro fora do intervalo permitido
DG_ERROR_PARAM_OUTOFRANGE - Parâmetro Enable fora do intervalo permitido (DG_ENABLE ou DG_DISABLE)

Copyright © 2010 Digivoice Eletrônica

Wave49ToGsm (dg_Wave49ToGsm)

Converte arquivo de áudio do formato Wave49 para o formato GSM

Declarações:

ActiveX:
```
SHORT Wave49ToGsm(LPCTSTR Source, LPCTSTR Target);
```

API:
```
SHORT dg_Wave49ToGsm(char *Source, char *Target);
```

Parâmetros:

Source - String contendo o nome e o caminho completo do arquivo Wave49 que será convertido
Target - String contendo o nome e o caminho completo do arquivo Gsm que será gravad

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_CONV_SOURCE - Erro ao abrir o arquivo de origem
DG_ERROR_CONV_TARGET - Erro ao abrir o arquivo de destino
DG_ERROR_CONV - Erro indeterminado na conversão

Veja também: GsmToWave49.

OBS.: O arquivo wave tem ser do formato Wave 49 (GSM 6.10 modificado) para que a conversão seja possível

Copyright © 2010 Digivoice Eletrônica

Wave49ToGsmRaw (dg_Wave49ToGsmRaw)

Converte arquivo de áudio do formato Wave49 para o formato GSMRaw (sem cabeçalho)

Declarações:

ActiveX:
```
SHORT Wave49ToGsmRaw(LPCTSTR Source, LPCTSTR
Target);
```

API:
```
SHORT dg_Wave49ToGsmRaw(char *Source, char *Target);
```

Parâmetros:

Source - String contendo o nome e o caminho completo do arquivo Wave49 que será convertido
Target - String contendo o nome e o caminho completo do arquivo GsmRaw que será gravado

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_CONV_SOURCE - Erro ao abrir o arquivo de origem
DG_ERROR_CONV_TARGET - Erro ao abrir o arquivo de destino
DG_ERROR_CONV - Erro indeterminado na conversão

Veja também: GsmRawToWave49.

OBS.: O arquivo wave tem ser do formato Wave 49 (GSM 6.10 modificado) para que a conversão seja possível

Copyright © 2010 Digivoice Eletrônica

WaveToGsm (dg_WaveToGsm)

Converte arquivo de áudio do formato Wave para o formato GSM

Declarações:

ActiveX:
```
SHORT WaveToGsm(LPCTSTR Source, LPCTSTR Target);
```

API:
```
SHORT dg_WaveToGsm(char *Source, char *Target);
```

Parâmetros:

Source - String contendo o nome e o caminho completo do arquivo Wave que será convertido
Target - String contendo o nome e o caminho completo do arquivo Gsm que será gravado

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_CONV_SOURCE - Erro ao abrir o arquivo de origem
DG_ERROR_CONV_TARGET - Erro ao abrir o arquivo de destino
DG_ERROR_CONV - Erro indeterminado na conversão

Veja também: GsmToWave.

OBS.: O arquivo wave tem que ser do formato: 8kHz, 16 bits, Mono para que a conversão seja possível.

Copyright © 2010 Digivoice Eletrônica

WaveToGsmRaw (dg_WaveToGsmRaw)

Converte arquivo de áudio do formato Wave para o formato GsmRaw (sem cabeçalho)

Declarações:

ActiveX:
```
SHORT WaveToGsmRaw(LPCTSTR Source, LPCTSTR Target);
```

API:
```
SHORT dg_WaveToGsmRaw(char *Source, char *Target);
```

Parâmetros:

Source - String contendo o nome e o caminho completo do arquivo Wave que será convertido
Target - String contendo o nome e o caminho completo do arquivo GsmRaw que será gravado

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_CONV_SOURCE - Erro ao abrir o arquivo de origem
DG_ERROR_CONV_TARGET - Erro ao abrir o arquivo de destino
DG_ERROR_CONV - Erro indeterminado na conversão

Veja também: GsmRawToWave.

OBS.: O arquivo wave tem que ser do formato: 8kHz, 16 bits, Mono para que a conversão seja possível

Copyright © 2010 Digivoice Eletrônica

WriteCode (dg_WriteCode)

Permite gravar string de segurança na memória da placa

Declarações:

ActiveX:
```
SHORT WriteCode(SHORT wCard, SHORT *wNewData);
```

API:
```
SHORT dg_WriteCode(short wCard, short *wNewData);
```

Parâmetros:

wCard - A placa que deverá ser gravada
wNewData - Dado a ser gravado

Valor de Retorno:

DG_EXIT_SUCCESS - Executado com sucesso
DG_ERROR_CARD_OUT_OF_RANGE - Parâmetro placa foi passado com valor maior do que o número de placas instaladas

Veja também: CheckCode.

Copyright © 2010 Digivoice Eletrônica

Eventos

Conceitualmente, eventos e mensagens se referem à mesma coisa, ou seja, são avisos que a VoicerLib repassa para a aplicação, indicando o acontecimento de algo ou resposta de alguma ação iniciada através de comando do programa.

No ActiveX, estes avisos são chamados de eventos, termo usual aos programadores que utilizam ferramentas visuais, como Delphi ou Visual Basic. Já o termo mensagem é mais genérico e conhecido dos programadores habituados a utilizar C/C++.

Normalmente, quando um evento é tratado no programa, a ferramenta visual gera uma rotina que será responsável por tratar este evento. Todas estas rotinas assumirão o nome padrão Onxxxxx onde o xxxxx indicará o tipo do evento. Estas rotinas receberão parâmetros padrões, de acordo com a funcionalidade de cada evento. Alguns recebem apenas a porta, outros além da porta, recebem um estado ou qualquer outro tipo de identificação. Neste guia de referência, estes parâmetros serão relacionados no item Parâmetros Recebidos de cada evento.

Nas funções de API, a VoicerLib repassa os eventos que ocorrem na placa através das mensagens que devem ser tratadas em uma rotina específica, que foi chamada ReceiveEvents (leia o capítulo "Conceitos Basicos - API"). Neste Guia de Referência os dados referentes ao context_data são relacionados no item Context_Data (API).

Copyright © 2010 Digivoice Eletrônica

OnAfterDial (EV_AFTERDIAL)

Ocorre ao término de uma discagem feita por um único comando Dial.

Descrição:

Para efetuar uma discagem, é necessário utilizar o método Dial, que pode discar de 1 a 28 números. Visando facilitar o controle de fluxo do programa, o evento OnAfterDial ocorrerá tão logo todos os dígitos passados como parâmetro no método Dial sejam discados pela placa. Se for utilizado o parâmetro PauseAfterDial do método Dial, o evento OnAfterDial só será gerado após a discagem mais a pausa.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_AFTERDIAL (0x2b)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnAfterFlash (EV_AFTERFLASH)

Ocorre ao término de um comando Flash

Descrição:

De maneira semelhante ao evento OnAfterDial, o evento OnAfterFlash ocorrerá ao término do comando executado pelo método Flash

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_AFTERFLASH (0x2c)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnAfterMakeCall

Ocorre ao término do método de discagem MakeCall

Descrição:

Ocorre ao término da discagem com supervisão quando utilizado o MakeCall, disponível somente no ActiveX.

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento

Status – Pode assumir:
- mkOK (0) - Discou com sucesso
- mkNoDialTone (1) – Sem tom de discagem
- mkDelivered (2) – Ligação entregue sem supervisão
- mkNoAnswer (3) – Ligação não foi atendida
- mkBusy (4) – Ligação ocupada
- mkAnswered (5) - Ligação atendida
- mkAborted (6) – Ligação cancelada pelo AbortCall
- mkDialToneAfterDial (7) - Indica recebimento de tom de linha depois da discagem. Normalmente esta situação indica algum problema com o ambiente (PABX, linha, etc...)
- mkFaxDetected (8) - Indica detecção de fax

Copyright © 2010 Digivoice Eletrônica

OnAfterPickUp (EV_AFTERPICKUP)

Ocorre ao término de um comando PickUp.

Descrição:

De maneira semelhante ao evento OnAfterDial, o evento OnAfterPickUP ocorrerá após a pausa determinada na chamada do método PickUp.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_AFTERPICKUP (0x2d)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnAnswerDetected (EV_ANSWERED)

Ocorre quando a placa detecta atendimento da ligação originada.

Descrição:

A placa possui recursos de monitoração do status da linha à ela conectada, e de repassá-los para a aplicação. Para tanto, ela deve ser habilitada a utilizar estes recursos.

Nas placas FXO, o programador tem a flexibilidade de a qualquer ponto da aplicação habilitar e desabilitar tais recursos, fazendo uso dos métodos EnableAnswerDetection para habilitar e DisableAnswerDetection para desabilitar.

Quando uma ligação é originada pela placa, e os recursos citados estiverem habilitados, o evento OnAnswerDetected ocorrerá no momento em que essa ligação for atendida. Isto é válido também para a placa digital, pois ao receber o atendimento, a VoicerLib também gera o evento.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento
AnswerType - Indica se o atendimento foi por áudio (AUDIO_DETECTED - 0) ou por timeout (TIMEOUT_DETECTED - 1)

Context_Data (API):

command - EV_ANSWERED (0x1f)
port: Indica a porta da placa que gerou o evento
data: Mesmos valores do parâmetro AnswerType
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnAudioSignalDetected (EV_AUDIO_SIGNAL)

Ocorre quando a porta recebe um sinal de áudio, MF, tom, etc...

Descrição:

Este evento tem por finalidade facilitar a análise dos sinais recebidos e suas respectivas cadências pois repassa para a aplicação todos os sinais recebidos, sem tratamento nenhum.

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento
SignalCode: Indica o sinal de áudio recebido:
- CP_SILENCE=20H
- CP_AUDIO=21H
- CP_TONE1=22H
- CP_TONE2=23H
- CP_TONE3=24H
- CP_TONE4=25H
- CP_TONE5=26H
- CP_TONE6=27H
- CP_TONE7=28H
- CP_TONE8=29H
- CP_UNDEFINED = 0x40
- CP_INVALID = 0x41

Context_Data (API):

command - EV_AUDIO_SIGNAL (0x3e)
port/card: Indica a porta que gerou o evento
data: Mesmos valores do parâmetro SignalCode
data_aux: Não utilizado
card: Indica a placa que gerou o evento

Copyright © 2010 Digivoice Eletrônica

OnBusyDetected (EV_BUSY)

Ocorre quando a placa detecta tom de ocupado na linha.

Descrição:

Nas placas analógicas, o programador tem a flexibilidade de a qualquer ponto da aplicação habilitar e desabilitar tais recursos, fazendo uso dos métodos EnableCallProgress para habilitar e DisableCallProgress para desabilitar.

Sempre quando a placa tomar a linha conectada à ela, para originar uma ligação, e receber tom de ocupado, o evento OnBusyDetected ocorrerá, desde que os recursos de monitoração estejam habilitados. Isto é válido também para a placa digital ao receber bloqueio, não disponibilidade da porta, etc... a VoicerLib também gera o evento.

No protocolo ISDN PRI, é possível executar o método GetBusyCause para obter a causa do desligamento.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_BUSY (0x21)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnCallerID (EV_CALLERID)

Ocorre quando a placa detecta o identificador de A.

Descrição:

Este evento é gerado quando o identificador de A é recebido pelo tronco.

É possível ler o número através do método GetCallerID.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento
Número– Número identificado exatamente como enviado pela companhia telefônica

Context_Data (API):

command - EV_CALLERID (0x32)
port: Indica a porta da placa que gerou o evento
data: Quantidade de dígitos disponíveis para leitura
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnCalling (EV_CALLING)

Ocorre quando a placa detecta tom chamada na linha

Descrição:

Nas placas analógicas, o programador tem a flexibilidade de a qualquer ponto da aplicação habilitar e desabilitar tais recursos, fazendo uso dos métodos EnableCallProgress para habilitar e DisableCallProgress para desabilitar.

Sempre quando a placa tomar a linha conectada à ela, originar uma ligação e receber tom de chamada, o evento OnCalling ocorrerá, desde que os recursos de monitoração estejam habilitados. Isto é válido também para a placa digital, pois ao finalizar a troca de sinalização R2D MFC, a VoicerLib também gera o evento na cadência 1x4 (1 segundo de tom para 4 de intervalo).

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_CALLING (0x1d)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnCallStateChange

Ocorre em várias situações durante o MakeCall, disponível somente no ActiveX

Descrição:

O método MakeCall inicia uma discagem completa, com supervisão, frase inicial, discagem, etc... O evento OnCallStateChange indica, por canal, em qual etapa o MakeCall está no momento. Este evento deve ser utilizado para fins de monitoramento e depuração de programas.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento
Status - Indica o estado atual do MakeCall, podendo assumir os seguintes valores:
- csPlayingStartPhrase (0) - Falando frase inicial
- csPickingUp (1) - Iniciando discagem sem transferência
- csInitialFlash (2) - Dando flash inicial
- csDialingPrefix (3) - Prefixo apos o flash
- csStartAnalysis (4) - Iniciando supervisão
- csNoAnswerReturn (5) - Retomada em caso de não atendimento
- csPlayingBusyPhrase (6) - Falando frase de retorno no caso de ocupado
- csPlayingNoAnswerPhrase (7) - Falando frase de retorno no caso de ocupado
- csWaitingDialTone (8) - Esperando tom de discagem
- csDialing (9) - Discando
- csCalling (10) - Chamando

Copyright © 2010 Digivoice Eletrônica

OnChannel (EV_CHANNEL)

Indica a mudança de porta de uma chamada.

Descrição:

Este evento ocorrerá sempre que a porta escolhida estiver ocupada e outra porta for alocada para a chamada. Lembrando que este evento não ocorrerá quando o parâmetro ISDN_SET_EXCLUSIVE_CHANNEL (ConfigISDNThread) estiver desabilitado (DG_DISABLE).

Caso a configuração ISDN_SET_EXCLUSIVE_CHANNEL (ConfigISDNThread) estiver habilitada (DG_ENABLE) e a porta escolhida estiver ocupada, o evento de ocupado será gerado (OnBusyDetected).

Parâmetros Recebidos (ActiveX):

OldPort – Indica a porta antiga
NewPort - Indica a porta nova

Context_Data (API):

command - EV_CHANNEL (0x70)
port: porta antiga
data: porta nova
data_aux: Não utilizado
card: Não utilizado

Veja Também: ConfigISDNThread.

Copyright © 2010 Digivoice Eletrônica

OnDialToneDetected (EV_DIALTONE)

Ocorre quando a placa detecta tom de discagem na linha.

Descrição:

Nas placas analógicas, o programador tem a flexibilidade de a qualquer ponto da aplicação habilitar e desabilitar tais recursos, fazendo uso dos métodos EnableCallProgress para habilitar e DisableCallProgress para desabilitar. Também é necessário configurar a frequência de tons de acordo com a Central PABX utilizada (SetFrequency).

Sempre quando a placa tomar a linha conectada à ela para originar uma ligação e receber um tom de discagem, o evento OnDialToneDetected ocorrerá.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_DIALTONE (0x1e)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnDigitDetected (EV_DTMF)

Ocorre sempre que a placa detecta um dígito

Descrição:

Toda vez que a placa detectar um dígito (DTMF, MFF, MFT, etc), o evento OnDigitDetected ocorrerá, e o código ASCII dígito será passado através do parâmetro Digit.

Um dígito DTMF é um par de frequências pré-definidas e na faixa da voz. O programador deve estar ciente que a placa sempre está habilitada a detectar os dígitos, logo, durante uma conversação é comum a placa detectar alguns dígitos indevidamente, pois ela analisa todo o conteúdo de frequência da voz dos locutores e toda vez em que ela encontrar na voz um par de frequências que coincidam com um dígito DTMF, o evento OnDigitDetected ocorrerá. Quando o programador quiser validar os dígitos apenas em certos momentos, é necessário fazer uso do método GetDigits e do evento OnDigitsReceived ao invés de tratar dígito a dígito.

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento
Digit – Código ASCII do dígito recebido

Context_Data (API):

command - EV_DTMF (0x1c)
port: Indica a porta da placa que gerou o evento
data: Código ASCII do dígito recebido
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnDigitsReceived (EV_DIGITSRECEIVED)

Ocorre sempre em resposta ao método GetDigits é chamado.

Descrição:

Este evento ocorre em resposta ao método GetDigits.
Basicamente este método espera um determinado número de dígitos e/ou finalizador por um intervalo de tempo. Os dígitos detectados devem ser recuperados através do método ReadDigits.

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento
Status – Indica qual das situações gerou o evento:
edOff (0) - Desligado
edWaiting (1) - Em espera
edMaxDigits (2) – Recebeu o número máximo de dígitos estipulado
edDigitTimeout (3) – Estourou o tempo total para entrada de todos os dígitos
edInterDigitTimeOut (4) – Estourou o tempo interdígito
edTermDigit (5) – Recebeu o dígito finalizador
edDigitOverMessage (6) – Detectou dígito durante a reprodução

Context_Data (API):

command - EV_DIGITSRECEIVED (0x31)
port: Indica a porta da placa que gerou o evento
data: Código ASCII do dígito recebido
- EdMaxDigits(2) – Recebeu o número máximo de dígitos estipulado.
- EdTermDigit(5) – Recebeu o digito finalizador
- EdDigitTimeout(3) – Estourou o tempo total para entrada de todos os dígitos.
- EdInterDigitTimeOut(4) – Estourou o tempo inter-dígito.
- EdDigitOverMessage – Detectou dígito durante a reprodução
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnE1Alarm (EV_E1_ALARM)

Indica a ocorrência de um alarme nos framers E1

Descrição:

A monitoração dos alarmes é essêncial para o funcionamento das aplicações que utilizam a placa E1.

Parâmetros Recebidos:

Card – Indica a placa da placa que gerou o evento
E1ID - Indica qual dos 2 framers E1 gerou o alarme

AlarmCode - Mostra o alarme gerado:
- ALARM_RSLIP (0x01) - Escorregamento (problema de sincronismo)
- ALARM_RAIS (0x02) - Alarme remoto
- ALARM_AISS (0x04) - Indicação de alarme
- ALARM_AIS16S (0x08) - Indicação de alarme canal 16
- ALARM_LOSS (0x10) - Perda de sinal
- ALARM_CRC4SYNC (0x20) - Alarme de CRC4
- ALARM_MFSYNC (0x40) - Sincronismo de multiquadro
- ALARM_SYNC (0x80) - Sincronismo de quadro
- ALARM_LINK (0x100) - Sincronismo do protocolo ISDN

AlarmData - Se for ALARM_RSLIP, indica o contador de vezes que detectou o escorregamento.
Nos outros alarmes, pode receber ON (1) ou OFF (0).

Context_Data (API):

command - EV_E1_ALARM (0x38)
port: Indica a placa que gerou o evento
data: Código do alarme (opções do AlarmCode)
data_aux: Estado do alarme (ON/OFF), contador no caso do SLIP
card: Indica a placa que gerou o evento

Copyright © 2010 Digivoice Eletrônica

OnE1FramerResponse (EV_FRAMER)

Ocorre sempre que for solicitado um comando para o Framer E1

Descrição:

Este evento permite analisar a situação dos framers E1 e deverá somente ser utilizado quando houver algum problema de comunicação com uma outra central por exemplo.

Parâmetros Recebidos:

Card – Indica a placa que gerou o evento
Data1 - Dado indicativo do primeiro framer
Data2 - Dado indicativo do segundo framer

Context_Data (API):

command - EV_FRAMER (0x3b)
port: Indica a placa que gerou o evento
data: Dado indicativo do primeiro framer
data_aux: Dado indicativo do segundo framer
card: Indica a placa que gerou o evento

Copyright © 2010 Digivoice Eletrônica

OnE1GroupB (EV_GROUP_B)

Ocorre quando a thread E1 recebe um grupo B

Descrição:

Este evento ocorrerá sempre quando a thread E1 estiver em uso.

Parâmetros Recebidos:

Card – Indica a porta da placa que gerou o evento
Value: Indica o tipo de dado

Context_Data (API):

command - EV_GROUP_B (0x34)
port: Indica a porta que gerou o evento
data: Mesmos valores do parâmetro Value
data_aux: Não utilizado
card: Indica a placa que gerou o evento

Copyright © 2010 Digivoice Eletrônica

OnE1StateChange (EV_E1CHANGESTATUS)

Ocorre sempre quando algum estado da sinalização R2D é alterado ou quando a porta especificada foi liberada no protocolo ISDN PRI (C_E1_IDLE).

Descrição:

Durante a troca de sinalização R2D, ocorrem diversos eventos indicando o ponto onde o protocolo se situa. Basicamente este evento deve ser utilizado para monitoração, não sendo necessário nenhum tratamento específico já que a thread E1 cuida de todos os detalhes para o desenvolvedor.

Caso o programador esteja tratando toda a sinalização sem utilizar a thread R2 (não recomendado), este evento deverá ser utilizado para dar as informações necessárias para a troca de sinalização.

No protocolo ISDN PRI, este evento indica que a porta especificada foi liberada (C_E1_IDLE). Ele normalmente ocorre após a execução do método HangUp.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento
State - Indica o estado atual e pode receber os seguintes valores:
- C_GRUPO_B (0x1005) - Solicita grupo B para aplicação
- C_NOTCOMPLETED (0x1015) - Ligação não completada
- C_B_ENDCALL (0x1014) - Assinante B desligou
- C_ANSWERED (0x1012) - Assinante B atendeu ou retornou depois de C_B_ENDCALL
- C_CONGESTION (0x1013) - Congestionamento
- C_SEIZURE (0x100b) - Ocupação
- C_GROUP_II (0x1017) - Grupo II recebido
- C_NUMBER_RECEIVED (0x100d) - Número recebido durante ligação entrante
- C_UNAVAILABLE (0x1011) - Canal não disponível
- C_GROUP_I (0x101c) - Recebeu grupo I

Copyright © 2010 Digivoice Eletrônica

- C_ID_RECEIVED (0x100a) - Recebeu identificação de assinante A
- C_E1_IDLE (0x102a) - Canal R2 foi para o estado LIVRE
- C_E1_SEIZURE_ACK (0x102d) - Canal R2 recebeu confirmação de ocupação
- C_SEND_GROUP_B (0x102e) - Recebeu Grupo B da aplicação
- C_SEND_BACKWARD_SIGNAL (0x102f) - Recebeu comando de envio de sinal "para trás" da aplicação

Context_Data (API):

command - EV_E1CHANGESTATUS (0x33)
port: Indica a porta da placa que gerou o evento
data: Mesmos valores do State
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnErrorDetected (EV_ERRORDETECTED)

Ocorre quando acontece algum erro de hardware

Descrição:

Este erro só deverá acontecer se houver alguma falha de hardware, que pode ser causada por falhas de instalação ou conflitos com outras placas instaladas.

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento
ErrorType: Indica o tipo de erro, podendo assumir
- ERROR_FIFO_FULL (0x79) - Está cheia a fifo de comandos para a placa

Context_Data (API):

command - EV_ERRORDETECTED (0x39)
port/card: Indica a placa que gerou o evento
data: Mesmos valores do parâmetro ErrorType
data_aux: Não utilizado
card: Indica a placa que gerou o evento

Copyright © 2010 Digivoice Eletrônica

OnErrorNodeConfig (EV_ERROR_NODE_CONFIG)

Ocorre quando há erro de configuração no ISDN PRI.

Descrição:

O tipo de ambiente local e remoto são iguais. Ambos configurados como CPE ou como NET.

Parâmetros Recebidos (ActiveX):

AlarmData - Indica a placa que gerou o evento
Span - Indica o span que gerou o evento
AlarmCode - Mostra o alarme gerado:
- ISDN_NET (0x00) - Ambos configurados como NET
- ISDN_CPE (0x01) - Ambos configurados como CPE

Context_Data (API):

command - EV_ERROR_NODE_CONFIG (0x74)
port: Indica o span da placa que gerou o evento
data: ISDN_NET ou ISDN_CPE
data_aux: Indica a placa que gerou o evento
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnFaxDetected (EV_FAX)

Ocorre quando um sinal de fax é detectado pela placa

Descrição:

O programador tem a flexibilidade de a qualquer ponto da aplicação habilitar e desabilitar tais recursos, fazendo uso dos métodos EnableCallProgress para habilitar e DisableCallProgress para desabilitar.

Sempre quando for detectado um sinal de fax na linha, o evento OnFaxDetected será chamado, permitindo tratamentos especiais, como por exemplo, desviar para o ramal do aparelho de fax.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_FAX (0x22)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnGSMSMSConfirmation (EV_GSMSMSCONFIRMATION)

Permite receber a confirmação do recebimento de uma mensagem SMS por parte do destinatário (este recurso depende da disponibilidade da operadora).

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_GSMCONFIRMATION (0x63)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnGSMError (EV_GSMERROR)

Ocorre quando um erro é detectado na porta ou na troca de sinalização nos módulos GSM.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento
State - Constante indicando o tipo de erro. Para obter a string correspondente ao tipo de erro, utilize o método ReturnCodeGSMToString.

Context_Data (API):

command - EV_GSMERROR (0x53)
port: Indica a porta da placa que gerou o evento
data: Mesmos valores do State
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnGSMMemory (EV_GSMMEMORY)

Ocorre em resposta ao método GSMClearAllSMS, informa que a quantidade de mensagens armazenadas no módulo GSM está disponível para ser lida com o método GSMGetMemory.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_GSMMEMORY (0x59)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnGSMMemoryFull (EV_GSMMEMORYFULL)

É gerado quando o módulo GSM atinge sua capacidade máxima de mensagem. Utilize o método GSMClearAllSMS para limpar as mensagens do módulo.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_GSMMEMORYFULL (0x60)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnGSMMessage (EV_GSMMESSAGE)

Ocorre quando uma mensagem é detectada pela placa.

Descrição:

Quando o módulo GSM recebe uma mensagem válida o evento é gerado, a mensagem não precisa ser necessariamente um SMS.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento
State - Constante indicando o tipo de erro. Para obter a string correspondente ao tipo de erro, utilize o método ReturnCodeGSMToString.

Context_Data (API):

command - EV_GSMMESSAGE (0x51)
port: Indica a porta da placa que gerou o evento
data: Mesmos valores do State
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnGSMOtherCall (EV_GSMOTHERCALL)

Quando o módulo GSM já atendeu uma chamada e recebe uma "outra chamada" este evento é gerado.
Para se obter o número do assinante da "outra chamada" utilize o método GetCallerID.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_GSMOTHERCALL (0x58)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnGSMReady (EV_GSMREADY)

Ocorre quando a thread GSM foi criada e inicializada com sucesso, esse evento deve ser aguardado antes do envio de qualquer outro comando a porta inicializado.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento
State - Constante indicando o tipo de erro. Para obter a string correspondente ao tipo de erro, utilize o método ReturnCodeGSMToString.

Context_Data (API):

command - EV_GSMREADY (0x55)
port: Indica a porta da placa que gerou o evento
data: Mesmos valores do State
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnGSMReturnOK (EV_GSMRETURNOK)

Evento gerado após a execução dos métodos GSMListSMS (status GSM_LIST) e GSMClearAllSMS (status GSM_CLEAR).
Se o status retornado for GSM_LIST, executar o método GSMGetIndexList para recuperar os indices das mensagens SMS do modulo GSM na porta correspondente.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento
State - Constante indicando o tipo de retorno. Para obter a string correspondente ao tipo de erro, utilize o método ReturnCodeGSMToString.

Context_Data (API):

command - EV_GSMRETURNOK (0x57)
port: Indica a porta da placa que gerou o evento
data: Mesmos valores do State
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnGSMSIM (EV_GSMSIM)

Ocorre na inserção ou retirada do SIM Card (chip) nos módulos GSM, permitindo por exemplo que o programador reinicie a thread GSM após a inserção de um chip na placa.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento
State - 0: Retirou SIM Card
1: Inseriu SIM Card

Context_Data (API):

command - EV_GSMSIM(0x62)
port: Indica a porta da placa que gerou o evento
data: Mesmos valores do State
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnGSMSignalQuality (EV_GSMSIGNALQUALITY)

Ocorre em resposta ao método GSMCheckSignalQuality, informa que o valor respectivo a qualidade do sinal já está disponível.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_GSMSIGNALQUALITY (0x56)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnGSMSMSReceived (EV_GSMSMSRECEIVED)

Ocorre quando uma mensagem SMS foi recebida e está disponível para leitura através do método GSMGetSMS.

Descrição:

Ao receber uma mensagem, o evento é gerado indicando a porta que recebeu a mensagem.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_GSMSMSRECEIVED (0x54)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnGSMSMSSent (EV_GSMSMSSENT)

Ocorre ao término do envio de uma mensagem SMS na porta especificada. Novas mensagens só devem ser enviadas após o recebimento desse evento.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_GSMSMSSENT (0x52)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnGSMTimeout (EV_GSMTIMEOUT)

Ocorre quando um comando enviado pela Voicerlib para o módulo GSM correspondente a porta da placa VB0404GSM não obtem uma resposta no tempo especificado no método ConfigGSMThread.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento
State - Constante indicando o tipo de erro. Para obter a string correspondente ao tipo de erro, utilize o método ReturnCodeGSMToString.

Context_Data (API):

command - EV_GSMTIMEOUT (0x50)
port: Indica a porta da placa que gerou o evento
data: Mesmos valores do State
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnLineOff (EV_LINEOFF)

Ocorre quando o usuário desliga o aparelho telefônico conectado à placa em paralelo.

Descrição:

O Evento OnLineOff ocorre sempre que o aparelho telefônico for desligado.

Nas placas digitais, este evento também é gerado de acordo com a sinalização R2D indicando fim de ligação.

Exclusivamente para a placa FXO, nas versões anteriores da VoicerLib este evento era gerado apenas quando havia uma conexão em paralelo com a placa. Agora este evento também ocorre quando a placa executa um comando HangUp indicando que está "no gancho".

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_LINEOFF (0x27)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnLineReady (EV_LINEREADY)

Ocorre quando um aparelho telefônico conectado à placa em paralelo toma a linha para originar uma ligação ou após executar o comando PickUp.

Descrição:

Nas placas FXO/GSM/FXS, quando a conexão é feita de acordo com o explicado no tópico Gravação em Paralelo deste manual, o evento OnLineReady também é gerado quando se detecta loop na linha.

Nas placas digitais, este evento também é gerado de acordo com a sinalização R2D indicando início de ligação.

Nas versões anteriores da VoicerLib este evento era gerado apenas quando havia uma conexão em paralelo com a placa. Agora este evento também ocorre quando a placa executa um comando PickUp indicando que está "fora do gancho".

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_LINEREADY (0x25)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnLoggerEvent (EV_LOGGEREVENT)

Ocorre em diversas situações quando se utiliza a thread de Logger

Descrição:

Este evento só ocorrerá quando estiver utilizando a thread de controle de logger, criada através do método CreateLoggerControl. O parâmetro LoggerStatus indicará diversas situações para que o aplicativo possa controlar as várias etapas da sinalização R2, o início e o fim das ligações, disparando ou finalizando as gravações. Para maiores informações, leia o capítulo Gravação em Paralelo

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento
LoggerStatus - Pode assumir os seguintes valores:
- LOGGER_FREE_WITH_BILLING (1) - Assinante livre com tarifação
- LOGGER_BUSY (2) - Ocupado
- LOGGER_NUMBER_CHANGED (3) - Número de destino mudou
- LOGGER_CONGESTION (4) - Congestionamento
- LOGGER_FREE_WITHOUT_BILLING (5) - Livre sem tarifação
- LOGGER_FREE_RETENTION (6) - Livre com retenção
- LOGGER_LEVEL_NUMBER_AVAILABLE (7) - Número disponível
- LOGGER_B_ENDCALL (20) - Assinante B desligou durante conversação
- LOGGER_B_RETURN (21) - Assinante B retornou à ligação
- LOGGER_LINEREADY (22) - Início da conversação
- LOGGER_LINEOFF (23) - Fim da conversação

Context_Data (API):

command - EV_LOGGEREVENT (0x37)
port: Indica a porta da placa que gerou o evento
data: Mesmos valores do parâmetro LoggerStatus
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnMailBoxDetected(EV_MBDETECTED)

O evento OnMailBoxDetected é gerado ao detectar atendimento por caixa postal ou por atendimento humano, em seguida a VoicerLib automaticamente desabilita deteccção de caixa postal (DisableMailBoxDetected).

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento
State – Indica o que aconteceu:
- 0 : Áudio, ou seja, é uma mensagem automática de caixa postal (secretária eletrônica)
- 1 : Silêncio, ou seja, ocorreu um atendimento humano

Context_Data (API):

command - EV_MBDETECTED (0x45)
port: Indica a porta da placa que gerou o evento
data: Mesmos valores do parâmetro State
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnMenu

Ocorre ao término de uma função de menu iniciada pelo MenuStart

Descrição:

O evento OnMenu é gerado após o término da função especial de menu

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento
OptionSelected – Opção digitada pelo usuário
Status – Indica o que aconteceu:
- msAborted (0) – Cancelado pelo método MenuAbort
- msTimeOut (1) – Usuário não digitou nada
- msRetriesExceeded (2) – Número de tentativas excedido
- msValidOptionDetected (3) – Foi digitada uma opção válida

Copyright © 2010 Digivoice Eletrônica

OnNameID (EV_NAMEID)

Ocorre quando a placa detecta a descrição do identificador da chamada (nome).

Descrição:

Após este evento, caso utilize a DLL, execute o método GetNameID.
Na OCX o nome é recebido no parâmetro Name do evento.

Parâmetros Recebidos (ActiveX):

Port – Indica a porta da placa que gerou o evento
Name– Nome identificado

Context_Data (API):

command - EV_NAMEID (0x71)
port: Indica a porta que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Veja Também: GetNameID.

Copyright © 2010 Digivoice Eletrônica

OnPlayStart (EV_PLAYSTART)

Ocorre sempre quando for iniciado o playback de qualquer mensagem.

Descrição:

Sempre quando desejar que a placa fale uma mensagem, é necessário utilizar o método PlayFile, e toda vez que este método é chamado, o evento OnPlayStart ocorrerá.

Este evento é indicado para atualizações na interface, como por exemplo, desabilitar botões, mostrar mensagens, etc...

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_PLAYSTART (0x2a)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnPlayStop (EV_PLAYSTOP)

Ocorre sempre quando for finalizado o playback de qualquer mensagem.

Descrição:

O evento OnPlayStop ocorrerá quando o playback for interrompido. Este evento retorna na variável StopStatus o motivo da interrupção.

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento
StopStatus – Indica o motivo da interrupção. Os códigos são:
- ssNormal (0) – Significa a mensagem foi falada por completo (terminou normalmente)
- ssDigitReceived (1) – Significa que a mensagem foi interrompida pelo recebimento de um dos dígitos passados como parâmetro para interrupção da mensagem. O dígito deve ser recuperado na propriedade Digits
- ssAbnormal (2) – Significa que a mensagem foi interrompida por causa de algum erro indeterminado
- ssStopped (3) – Significa que a mensagem foi interrompida pela chamada do método StopPlayFile

Context_Data (API):

command - EV_PLAYSTOP (0x29)
port: Indica a porta da placa que gerou o evento
data: Mesmos valores do parâmetro StopStatus
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnPrompt

Ocorre quando a função de prompt chega ao fim.

Descrição:

O Evento ocorre ao final do prompt, que foi iniciado pelo método PromptStart.

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento
Value – Valor digitado pelo usuário
Status – Pode assumir os valores:
- mpAborted (0) – Cancelado pelo método PromptAbort
- mpRetriesExceeded (1) – Número de tentativa excedido
- mpOk (2) – Usuário confirmou entrada de dados
- mpCanceled (3) – Usuário cancelou

Copyright © 2010 Digivoice Eletrônica

OnR2Received (EV_R2)

Ocorre sempre quando a sinalização R2 é recebida pela placa.

Descrição:

Este evento ocorrerá sempre quando alguma sinalização R2 é recebida pela placa. Antes de receber qualquer sinalização é necessário habilitar a recepção de R2 através do método SendR2Command passando como parâmetro R2_ENABLEDETECTION.

Quando a thread R2 estiver sendo utilizada, o controle de receber ou não estas informações fica sob responsabilidade da biblioteca.

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento
Signal - Indica o sinal R2 recebido:
- R2_IDLE (0x9)
- R2_CLEAR_FOWARD (0x9)
- R2_SEIZURE (0x1)
- R2_BACKWARD_DISCONNECTION (0x1)
- R2_SEIZURE_ACK (0xd)
- R2_BILLING (0xd)
- R2_CLEAR_BACK (0xd)
- R2_ANSWERED (0x5)
- R2_BLOCKED (0xd)
- R2_FAILURE (0xd)
- R2_ENABLEDETECTION (0x10)
- R2_DISABLEDETECTION (0x20)

Context_Data (API):

command - EV_R2 (0x35)
port: Indica a porta da placa que gerou o evento
data: Mesmos valores do parâmetro Signal
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnRecording (EV_RECORDING)

Ocorre durante a gravação de uma mensagem.

Descrição:

O evento OnRecording é ideal para monitorar o andamento de uma gravação. A variável ElapsedTime contém a duração em segundos da gravação até aquele momento.

Através deste evento é possível limitar o tamanho das mensagens ou mesmo exibir informações sobre o andamento da gravação.

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento
ElapsedTime – Indica a duração da gravação em segundos naquele instante

Context_Data (API):

command - EV_RECORDING (0x30)
port: Indica a porta da placa que gerou o evento
data: Indica a duração da gravação em segundos naquele instante
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnRecordStart (EV_RECORDSTART)

Ocorre sempre quando for iniciado a gravação de qualquer mensagem.

Descrição:

Sempre quando se desejar gravar uma conversa, faça uso do método RecordFile, e toda vez que este método é chamado, o evento OnRecordStart ocorrerá.

Este evento é indicado para atualizações na interface, como por exemplo, desabilitar botões, mostrar mensagens, etc...

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_RECORDSTART (0x2e)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnRecordStop (EV_RECORDSTOP)

Ocorre sempre quando for finalizado a gravação de qualquer mensagem.

Descrição:

O evento OnRecordStop ocorrerá quando a gravação for interrompida. Este evento retorna na variável Status o motivo da interrupção.

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento
StopStatus – Indica o motivo da interrupção. Os códigos são:
- ssNormal (0) - Significa que a mensagem foi interrompida com sucesso
- ssDigitReceived (1) – Significa que a mensagem foi interrompida pelo recebimento de um dos dígitos passados como parâmetro para interrupção da mensagem. O dígito deve ser recuperado na propriedade Digits
- ssAbnormal (2) – Significa que a mensagem foi interrompida por causa de algum erro indeterminado
- ssStopped (3) – Significa que a mensagem foi interrompida pela chamada do método StopRecordFile

Context_Data (API):

command - EV_RECORDSTOP (0x2f)
port: Indica a porta da placa que gerou o evento
data: Mesmos valores do parâmetro StopStatus
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnReverseCharging (EV_REVERSECHARGING)

Ocorre sempre quando a chamada entrante requisitar a inversão de cobrança da chamada no protocolo ISDN PRI(Ligação a cobrar).

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_REVERSECHARGING (0x73)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnRingDetected (EV_RINGS)

Ocorre sempre quando for detectado o sinal de ring na linha.

Descrição:

O evento ring ocorre sempre quando o sinal de ring for detectado pela placa, portanto, se o usuário demorar 3 toques para atender o telefone, o evento OnRingDetected ocorrerá 3 vezes.

Isto é válido também para a placa digital, a VoicerLib gera o evento na cadência 1x4 (1s de ring para 4s de intervalo).

No protocolo ISDN PRI, este evento ocorrerá somente se o campo ISDN_AUTO_SIGNALING (ConfigISDNThread) estiver habilitado (DG_ENABLE).

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command - EV_RINGS (0x1b)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnSilenceDetected (EV_SILENCE)

Ocorre sempre quando for detectado o silêncio ou áudio em determinada porta

Descrição:

Para habilitar a detecção de silêncio utiliza-se o EnableSilenceDetection passando os valores mínimos de silêncio ou áudio. Cada vez que o estado se altera entre silêncio e áudio, o evento OnSilenceDetected é gerado, passando no parâmetro State o que foi detectado.

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento
State - Indica se foi detectado um silêncio (DG_SILENCE_DETECTED - 1) ou áudio (DG_AUDIO_DETECTED - 0)

Context_Data (API):

command - EV_SILENCE (0x3f)
port: Indica a porta da placa que gerou o evento
data: Indica se foi detectado um silêncio (DG_SILENCE_DETECTED) ou áudio (DG_AUDIO_DETECTED)
data_aux: Não utilizado
card: Não utilizado

Copyright © 2010 Digivoice Eletrônica

OnText (EV_TEXT)

Ocorre sempre que um texto for recebido pela aplicação (VoicerLib) no protocolo ISDN PRI.

Descrição:

Após receber o evento, executar o método GetText para obter o texto recebido.

Parâmetros Recebidos:

Port – Indica a porta da placa que gerou o evento

Context_Data (API):

command: EV_TEXT (0x72)
port: Indica a porta da placa que gerou o evento
data: Não utilizado
data_aux: Não utilizado
card: Não utilizado

Veja Também: GetText.

Copyright © 2010 Digivoice Eletrônica

Propriedades exclusivas do ActiveX

Neste capítulo, são mostradas as propriedades que são de uso exclusivo do ActiveX. A versão de API tem funções que desempenham as mesmas funções.

ConfigPath

Tipo: String

Função: Determina o arquivo e o caminho de configuração para todas placas.
Descrição: Esta propriedade permite ao programador customizar o nome e o caminho do arquivo de configuração para todas placas. Se nenhum valor for informado será utizado **\%PROGRAM FILES%\Voicerlib4**.

NA API o caminho de configuração é passado diretamente pelo método StartVoicerLib**.**

Copyright © 2010 Digivoice Eletrônica

StockSigPath

Tipo: String

Função: Determina o caminho onde serão encontradas os arquivos de áudio (wave) utilizados para reproduzir valores por extenso, data, etc...

Descrição: Ao distribuir uma aplicação onde são utilizadas os métodos PlayCurrency, PlayDate, PlayTime, PlayNumber ou PlayList, o desenvolvedor necessita também enviar as mensagens a serem utilizadas. Esta propriedade permite configurar o caminho para estas mensagens.

A seguir apresentamos os nomes dos arquivos fornecidos com a biblioteca e a respectiva frase que reproduz.

Copyright © 2010 Digivoice Eletrônica

Tabela de frases do stocksigs

0.wav	zero
1.wav	um
1f.wav	uma
2.wav	dois
2f.wav	duas
3.wav	três
4.wav	quatro
5.wav	cinco
6.wav	seis
7.wav	sete
8.wav	oito
9.wav	nove
10.wav	dez
11.wav	onze
12.wav	doze
13.wav	treze
14.wav	quartorze
15.wav	quinze
16.wav	dezesseis
17.wav	dezessete
18.wav	dezoito
19.wav	dezenove
20.wav	vinte
30.wav	trinta
40.wav	quarenta
50.wav	cinquenta
60.wav	sessenta
70.wav	setenta
80.wav	oitenta

Copyright © 2010 Digivoice Eletrônica

90.wav	noventa
100.wav	cem
200.wav	duzentos
200f.wav	duzentas
300.wav	trezentos
300f.wav	trezentas
400.wav	quatrocentos
400f.wav	quatrocentas
500.wav	quinhentos
500f.wav	quinhentas
600.wav	seiscentos
600f.wav	seiscentas
700.wav	setecentos
700f.wav	setecentas
800.wav	oitocentos
800f.wav	oitocentas
900.wav	novecentos
900f.wav	novecentas
abril.wav	abril
agosto.wav	agosto
barra.wav	barra
BEEP.wav	beep
bilhao.wav	bilhão
bilhoes.wav	bilhões
branco.wav	branco
centavo.wav	centavo
centavos.wav	centavos
cento.wav	cento
de.wav	de

Copyright © 2010 Digivoice Eletrônica

dereais.wav	de reais
dezembro.wav	dezembro
e 1.wav	e
e.wav	e
fevereiro.wav	fevereiro
hora.wav	hora
horas.wav	horas
janeiro.wav	janeiro
julho.wav	julho
junho.wav	junho
maio.wav	maio
marco.wav	março
mil.wav	mil
milhao.wav	milhão
milhoes.wav	milhões
minuto.wav	minuto
novembro.wav	novembro
outubro.wav	outubro
ponto.wav	ponto
reais.wav	reais
real.wav	real
segundo.wav	segundo
segundos.wav	segundos
setembro.wav	setembro
traco.wav	traço
trilhao.wav	trilhão
trilhoes.wav	trilhões
virgula.wav	vírgula

Caso o desenvolvedor queira fazer sua própria locução ou mesmo em outra língua basta criar as frases mostradas anteriormente, respeitando-se sempre o nome do arquivo e o formato utilizado **(wave lei mi, 8Khz, 8 bits, mono).**

Copyright © 2010 Digivoice Eletrônica

Índice Remissivo

- 3 -

32/64 bits 14

- A -

AbortCall 142
Alarmes 99
Atendendo e Desligando 47
AudioMonitor 143

- B -

BINA 67

- C -

CancelGetDigits 144
ChatAddPort 145
ChatDisablePort 146
ChatEnablePort 147
ChatRemovePort 149
CheckCode 150
ClearDigits 151
Compilando a Voicerlib Linux 25
Conceitos Básicos ActiveX 32
Conceitos Básicos API 28
Conferência entre portas 79
ConfigCallProgress 153
ConfigCustomCAS 158
ConfigE1Thread 161
ConfigGSMThread 164
ConfigISDNThread 166
Configurações de Sincronismo 98
ConnectAudioChannels 152
CreateCallProgress 168
CreateChatRoom 170
CreateCustomCAS 171
CreateE1Thread 173
CreateGSMThread 175
CreateISDNThread 177
CreateLoggerCCS 180
CreateLoggerControl 178

- D -

Debug E1 117
DefinePortResource 182
Depurando aplicativos 117
DestroyCallProgress 184
DestroyChatRoom 185
DestroyCustomCAS 186
DestroyE1Thread 187
DestroyGSMThread 188
DestroyISDNThread 189
DestroyLoggerCCS 191
DestroyLoggerControl 190
Detecção de Dígitos 63
Detecção de Ring 46
Detecção de Silêncio 60
Detecção de Tons 61
Dg_SetEventCallback 192

Copyright © 2010 Digivoice Eletrônica

Índice Remissivo

Dial 193
DisableAGC 195
DisableAnswerDetection 196
DisableAutoFramers 198
DisableCallProgress 199
DisableDebug 200
DisableDTMFFilter 201
DisableE1Thread 202
DisableEchoCancelation 203
DisableGSMThread 204
DisableInputBuffer 205
DisableISDNThread 206
DisableMailBoxDetection 207
DisablePulseDetection 208
DisableSilenceDetection 209
DisconnectAudioChannels 210

- E -

Efetuando Chamadas E1 112
Efetuando Chamadas ISDN 106
EnableAGC 211
EnableAnswerDetection 212
EnableCallProgress 214
EnableDebug 216
EnableDTMFFilter 217
EnableE1Thread 219
EnableEchoCancelation 220
EnableFSKDetection 222
EnableGSMThread 223
EnableInputBuffer 224
EnableISDNThread 226
EnableMailBoxDetection 227
EnablePulseDetection 229
EnableSilenceDetection 231
Exemplo em linguagem C 26

- F -

Finalizando os Serviços 45
FinalizandoChamadas 110
Flash 233
ForceSingleSpan 235
Funções de Controle da Thread ISDN 105
Funções de Controle da Thread R2 116

- G -

GenerateMF 236
GetAbsolutePortNumber 238
GetAlarmStatus 239
GetBusyCause 240
GetCallerID 244, 263
GetCardBus 246
GetCardInfo 247
GetCardInterface 248
GetCardNumber 249
GetCardPortsCount 250
GetCardsCount 251
GetCardSlot 252
GetCardType 253
GetDigits 254
GetDriverEnabled 256
GetE1Number 257

Copyright © 2010 Digivoice Eletrônica

Índice Remissivo

GetE1ThreadStatus 259
GetISDNThreadStatus 260
GetLibVersion 261
GetLoggerCallType 262
GetPlayFormat 265
GetPortCardType 266
GetPortInterface 268
GetPortsCount 269
GetPortStatus 270
GetRecordFormat 271
GetRelativeChannelNumber 272
GetText 273
GetVersion 274
Gravação em Paralelo 91
Gravando uma Conversa 71
GSMCallControl 275
GSMCheckSignalQuality 278
GSMClearAllSMS 279
GSMDeleteSMS 280
GSMGetIndexList 281
GSMGetLastCommand 282
GSMGetMemory 283
GSMGetMessage 284
GSMGetSignalQuality 285
GSMGetSMS 286
GSMGetSMSConfirmation 287
GSMListSMS 288
GsmRawToWave 289
GsmRawToWave49 290
GSMReadAndDeleteSMS 291
GSMRestartPort 292
GSMSendCommand 293
GSMSendSMS 295
GSMSetPinNumber 297
GsmToWave 298
GsmToWave49 299
Guia De Migração De Versões Anteriores 33
Guia de programação 27

- H -

HangUp 300

- I -

Identificação de Chamadas 67
IdleAbort 301
IdleSettings 302
IdleStart 304
Inicialização E1 111
Inicializacao Protocolo ISDN 102
Inicializando os Serviços 44
Instalação no Linux 20
Instalação Windows 16
Instruções de instalação de placas 41
Introdução 12
Introdução CAS Customizado 101
IsCallInProgress 305
IsPlaying 306
IsRecording 307

Copyright © 2010 Digivoice Eletrônica

Índice Remissivo

- L -

LocalBridgeConnect 308
LocalBridgeDisconnect 309

- M -

MakeCall 310
MenuAbort 311
MenuErrorSettings 312
MenuStart 313
Módulos do Kernel 23

- N -

Não Atendimento 48

- O -

Ocupado 48
OnAfterDial 435
OnAfterFlash 436
OnAfterMakeCall 437
OnAfterPickUp 438
OnAnswerDetected 439
OnAudioSignalDetected 440
OnBusyDetected 441
OnCallerID 442
OnCalling 443
OnCallStateChange 444
OnChannel 445
OnDialToneDetected 446
OnDigitDetected 447
OnDigitsReceived 448
OnE1Alarm 449
OnE1FramerResponse 450
OnE1GroupB 451
OnE1StateChange 452
OnErrorDetected 454
OnErrorNodeConfig 455
OnFaxDetected 456
OnGSMError 458
OnGSMMemory 459
OnGSMMemoryFull 460
OnGSMMessage 461
OnGSMOtherCall 462
OnGSMReady 463
OnGSMReturnOK 464
OnGSMSignalQuality 466
OnGSMSIM 465
OnGSMSMSConfirmation 457
OnGSMSMSReceived 467
OnGSMSMSSent 468
OnGSMTimeout 469
OnLineOff 470
OnLineReady 471
OnLoggerEvent 472
OnMailBoxDetected 473
OnMenu 474
OnNameID 475
OnPlayStart 476
OnPlayStop 477

Copyright © 2010 Digivoice Eletrônica

OnPrompt 478
OnR2Received 479
OnRecording 480
OnRecordStart 481
OnRecordStop 482
OnReverseCharging 483
OnRingDetected 484
OnSilenceDetected 485
OnText 486

- P -

PauseInputBuffer 315
PickUp 316
Placas E1 93
Placas FXO 92
PlayBuffer 318
PlayCardinal 320
PlayCurrency 321
PlayDate 322
PlayFile 323
PlayList 325
PlayListAdd 326
PlayListClear 327
PlayListGetCount 328
PlayListRemoveItem 329
PlayNumber 330
PlayTime 331
Portas Virtuais 68
Preparando o Ambiente Linux 22
Primeiros Passos 14
Programacao da Placa E1 97
PromptAbort 332
PromptSettings 333
PromptStart 334
Protocolo CAS Customizado 101
Protocolo ISDN PRI 102
Protocolo R2D MFC 111

- R -

R2AskForGroupII 336
R2AskForID 337
R2SendGroupB 338
ReadDigits 339
Recebendo Chamadas E1 115
Recebendo ChamadasISDN 108
RecordFile 340
RecordPause 342
Recursos para o Desenvolvedor 13
Reproduzindo Mensagens 77
ResetError 343
ResetPortResource 345
ReturnCodeGSMToString 346
ReturnCodeToString 347
ReturnCodeToStringCauseBusy 348

- S -

SendISDNCommand 349
SendR2Command 351
SetAlarmMode 353
SetAudioInputCallback 355
SetBearerCapability 356

Copyright © 2010 Digivoice Eletrônica

Índice Remissivo

SetCallAfterAnswer 358
SetCallAfterPickup 359
SetCallBusyPhrase 360
SetCallBusyReturnFlash 361
SetCalledNumber 362
SetCallFlashTime 366
SetCallingNumber 364
SetCallNoAnswerPhrase 367
SetCallNoAnswerReturnFlash 368
SetCallNoAnswerRingCount 369
SetCallPauseBeforeAnalysis 370
SetCallStartFlash 371
SetCallWaitForDialTone 372
SetCardDetections 373
SetCardSyncMode 375
SetDetectionType 377
SetDialDelays 379
SetDigitFrequency 381
SetDigitGain 383
SetDTMFConfig 385
SetE1CRC4Option 386
SetFastDetection 387
SetFaxFrequencies 389
SetFramerLoop 391
SetFrequency 393
SetFXCardType 395
SetGSMMode 396
SetHangUpCause 397
SetLoggerSilenceThreshold 399
SetNameID 401
SetNextE1RxCount 402
SetPlayFormat 404
SetPortChatLog 406
SetPortGain 408
SetPortID 410
SetRecordFormat 411
SetRecordGain 414
SetReverseCharging 413
SetSilenceThreshold 415
SetStartE1RxCount 417
SetText 419
SetTwist 420
ShutdownVoicerLib 422
StartVoicerLib 423
StopPlayBuffer 425
StopPlayFile 426
StopRecordFile 427
Streaming de Áudio 81
Supervisão de Linha 48

- T -

Testando a Placa Instalada Windows 19
TxRxMixEnable 428

- U -

Utilizando a placa VB0404GSM 84

- W -

Wave49ToGsm 429
Wave49ToGsmRaw 430

Copyright © 2010 Digivoice Eletrônica

WaveToGsm 431
WaveToGsmRaw 432
WriteCode 433

Copyright © 2010 Digivoice Eletrônica

DigiVoice tecnologia em eletrônica Ltda.

Al. Juruá, 159 - Térreo
Alphaville - Barueri - SP
CEP 06455-010
(11) 2191-6363

www.digivoice.com.br