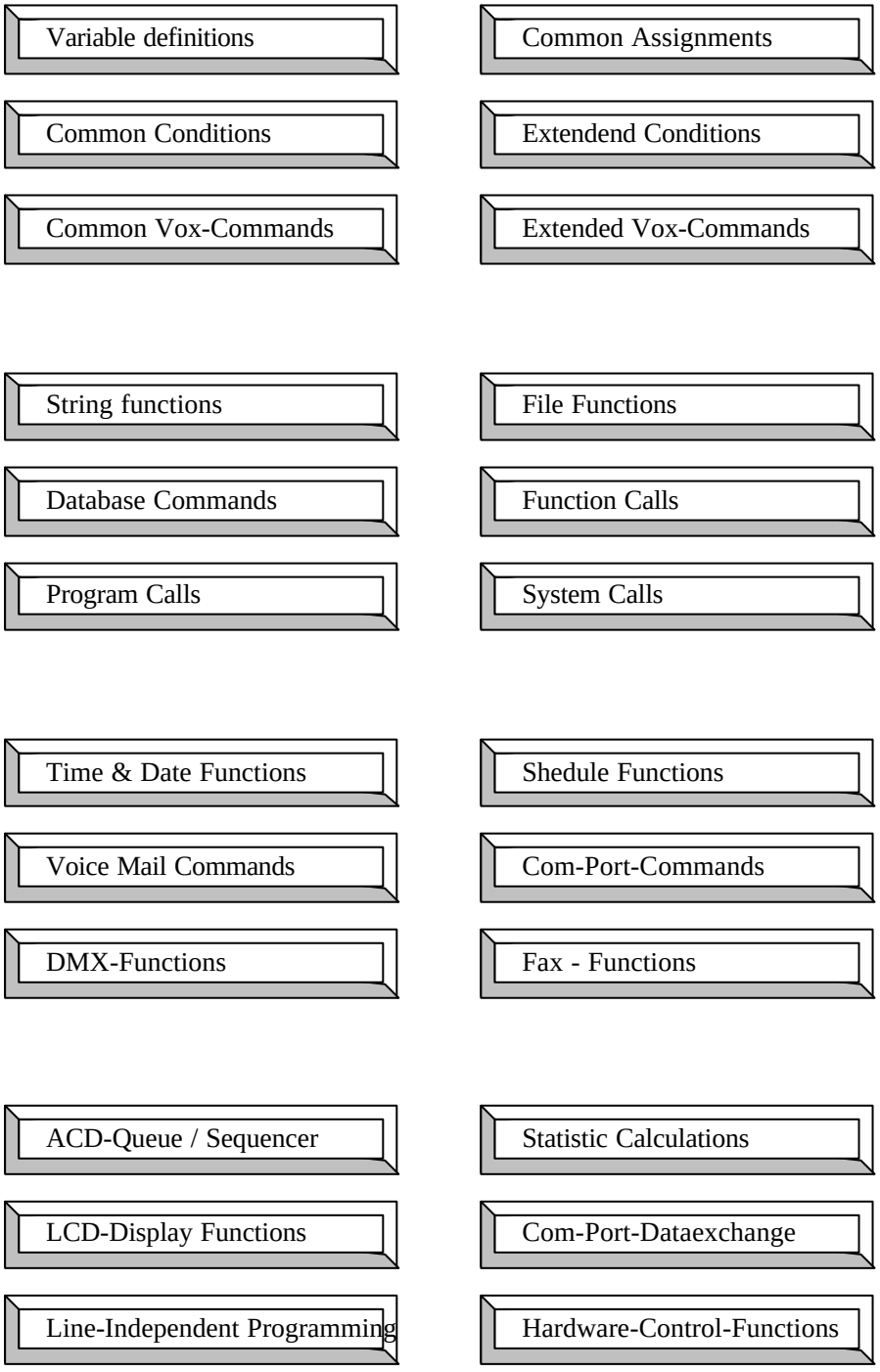
NVC DOCUMENTATION

A	Automated Call Distribution	acdqueue
B	Play Voice-Commands	dial, play, rec, voldet, waitring,..
C	Check Schedule Plan	checkschedule, checkweek
D	Data Base	dbuse, dbget, dbput...
E	File Functions -Ascii	fopen, fwrite, findfile
F	Fax Functions	sendfax, recfax
G	Goto a Sub-Program	do, proc, process, run
H		
IJ	Integration PABX	integration
K	Com Port Command	setdtr, getdtr
L		
M	Voice Mail Functions	getmbx, playmbx, recmbx, setmbx
N	Voice Mail Power Functions	playmbx, recmbx
O	ODBC Commands	odbcuse, odbcselect, odbcsql, odbcput...
PQ	Pipe Commands	openpipe, putpipe, sendpipe closepipe...
R	Random Generator	random
S	Speak Function	speak -string, -amount, -date, -time
Sch	Text-To-Speach Conversion	speak -tts
St	String Functions	strlen, strpart, strstr, strexchange
T	Time Functions	sleep, get time, get date, -difftime
U		
V	Voice Recognition	recognize
W		
X-Z	Conference Functions	conference

The NVT-Functions are classed by the following groups:



Variable Definitions

o integer definition	int number pwd loop
o constant integer definition	cst factor = 123
o string definition	str name filename directory
o constant string definition	str musicfile = "carmina.vox"

Common Assignments

o integer assignement	number = 5
o integer calculations	number = 5* 7 + 3 * (2+ loop)
o string assignement	filename = "carmina.vox"
o string concatenations	filename = directory+ "VM" + name + ".VOX"

Common Conditions

o if - condition

```
if number= 40 [then]
     name= "VM001.VOX"
else
     name= "VM002.VOX"
endif
```

o while - condition

```
while number > 0
     playvox musicfile
     number= number- 1
wend
```

o loop - endless condition

```
loop
     playvox musicfile
lend
```

o possible conditions

if a= b	if a# b
if a< b	if a> b
if a<= b	if a>= b
if not a	true on all numbers <= 0
if odd a	true on odd numbers (-1,1,3)

Extendend Conditions

o extended if - conditions

```
if number> 40 and number < 80 [then]
    name= "VM001.VOX"
else
    name= "VM002.VOX"
fi
```

o extended while - condition

```
while number > 0  and  st = 0
    playvox musicfile
    number= number- 1
wend
```

common vox - commands

o play a voice file	play filename [directory]
o record a voice file	rec filename [directory]
o wait for DTMF - tones	getkey 6
o blind dial a number	bldial dial_string
o dial with call analisys	dial dial_string
o volume (grunt) detection	voldet [-duration delta_time]
o wait for a number of rings	waitring nbr_of_rings
o go onhook	onhook
o go offhook	offhook

exended vox - commands

o speak actual date	speak -date [dd.mm] [-dir vox-dir]
o speak actual time	speak -time [hh:mm] [-dir vox-dir]
o speak actual date and time	speak -datetime [date,time] [vox-dir]
o speak a string literal	speak -literal number [vox_dir]
o speak an amount	speak -amount amount vox_file_name

String functions

o get length of a string	strlen string
o extract a part of a string	strpart string start stop result_string
o string comparison	if string1 = string2
o find a substring in a string	strstr string string_part_to_find

File Functions

o open an ASCII - file	fopen filename [directory] read | write
o close the opened file	fclose
o find a file	ffind filename [-dir directory] [-fast]
o read a line	fread str_variable_to_get_string
o write a line	fwrite string
o open, write and close	fwriteln filename [directory] string

Database Functions

o open a database	dbuse database_name
o read from the database	dbget field_name int_var | str_var
o write to the database	dbput field_name integer | string
o append an entry	dbappend
o seek an entry (needs .ndx)	dbseek
o skip forward | backword	dbskip [number_of_entries]
o goto top of database	dbtop
o goto bottom of database	dbbot
o goto record number	dbgo record_number
o get actual record number	dbrecno int_record_number

Function Calls

o function definition

```
proc getpassword
{
    !"procedure - commands"
}
```

o function call

```
getpassword
```

Program Calls

o enter into *.nvt program

```
do  "operator"
```

System Calls

o start a system call

```
run  "c:\nvt\nvtcmd deletefile e.e"
```
runs any DOS, Windows or OS/2 program!

o start an independent task

```
process  "c:\nvt\nvtcmd copyfile d.d e.e"
```
starts any DOS, Windows or OS/2 program

Note: The process may terminate much later than the caller is on the phone, so it is up to you to verify the result of the function the caller started.

Note: You can start only *.EXE and *.CMD files. To start the DOS-System-Call

```
DIR  > Z.Z
```
use or create the cmd-file ddir.cmd with the command in it.

Time & Date Functions

o wait for a defined time	sleep time_in_1/10_of_seconds

o predefined variables, set, when going offhook

o 1/100 of a second	str nvt_hseconds
o seconds	str nvt_seconds
o minutes	str nvt_minutes
o hours	str nvt_hours
o day of week 1.. 7 (sunday)	str nvt_wday
o day of year 1.. 365/ 366	str nvt_yday
o day of month	str nvt_day
o month	str nvt_month
o four digits 1991	str nvt_year
o five digits hh:mm (min)	str nvt_time
o five digits dd.mm (mt)	str nvt_date

o actual time into variables	GET DATE | TIME | DATETIME

Schedule Functions

o week - schedule	checkweek filename.nvw action
o calendar - schedule	checkholiday filename.nvh action
	checkschedule filename.nvh action

Voice Mail Commands

o play from a mailbox	playmbx mailbox_number
o record to a mailbox	recmbx mailbox_number
o get a mailbox - parameter	getmbx mailbox_number -status status
o set a mailbox - parameter	setmbx mailbox_number -status status
o move a mailbox to another	movembx source_mbx_nr target_mbx_nr
o delete a mailbox	delmbx mbx_nr

Com-Port-Commands

o set DTR of COMx to 1 | 0	setdtr com_adr state
o get DTR of COMx	getdtr com_adr int_state

DMX-Functions

o option, needs a DMX-Board

o connect 2 lines together	crossconnect vox_file_dir group [type]
o connect 2 lines together	conference int_source int_dest. 1 | 0
o connect 2 lines together	routexy source receive_dest transmit_dest

Fax - Functions

o option, needs a Fax-Board / Library

o send a FAX document	sendfax filename
o receive a FAX document	recfax filename

Brand New Functions

o File-Commands	fdelete fcopy frename fmove fopen -file 9
o ODBC-Commands	odbcopen odbcput odbcget odbcsql odbc close
o Pipe-Commands	openpipe putpipe getpipe closepipe
o Socket-Commands	opensocket putsocket getsocket closesocket

Name:	acdqueue	<line_nr> <action> [options]
Inputs:	line_nr	indicates the line
	action	acd-queue-handling
	action= 1	call-in (new caller)
	action= 2	status (re-query position)
	action= 3	exit-ok (caller diverted)
	action= 4	exit-ko (caller left queue)
Options:	-group	acd-queue [1 to 9]
Returns:	st	position in the queue
	st= 0	top, may start outdialling
	st= 1	1., "you are the first caller"
	st= 2	2., "you are the second.. "
	st= 3	3., "you are the third.. "
	st= cont...	continues till maximum (40)
	st= -1	the line is not queued yet (action 2 before action 1)
	st= -8	queue left with action 3 / 4
Includes:	nvtacdis.exe	ACD-Queue-Manager
	nvtacd.dbf	ACD-Queue-DataBase
Category:	ACD - Library	New Voice Tool - Swiss

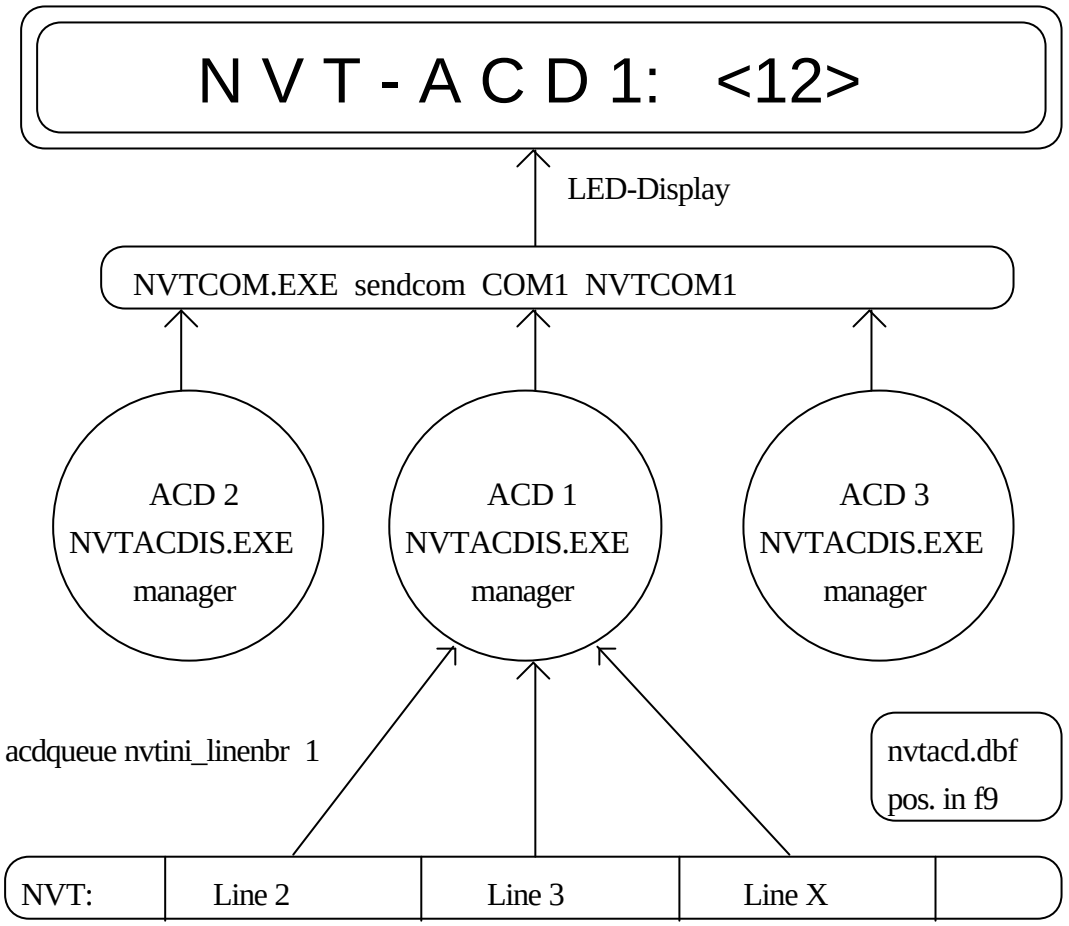
Description

This function will be used, whenever a pabx does not propose integrated sequencing. Imagine 12 persons calling the same phone-number, but only one or two operators can answer the phones.
Instead of transfering all of them every 20 seconds to the operators, and allowing random queue-dispatching, manage them through the acd-queue.
Every line will check-in once and then regularly ask for its position in the queue. When finished, the line gently logs-out to free its place in the queue.

Note

Please note, that one line can only be queued in one queue at the same time. NVTACDIS.EXE uses NVTACD.DBF to return the position in the queue. It is possible to queue a line in different queues by mirroring or renumbering the line.

Design



Note

Please note, that at least one NVTACDIS.EXE has to be lounched, when using the function acdqueue. Also the database NVTACD.DBF / .NDX / .CGP must be present.

[c:\nvt]nvtacdis manager ACD1 [nvtcom spectrumdisplay com1 "NVT - ACD%i %i"]

The option [ ] might be subject of changings, please contact your NVT-representative.
When using the display, do not forget to start the NVTCOM-manager in a .CMD-file:

```
rem acdstart.cmd to lounche the com-manager before the acd-manager
start nvtcom.exe sendcom com1 nvtcom1
nvtacdis.exe acd1 nvtcom spectrumdisplay com1 "NVT-ACD%%i < %%i >"
```

Note

Within a CMD-file you have to use %%i to pass the string %i.

Example

[c:\nvt]nvtacdis manager ACD1 (starts acd-queue-manager on acd-queue 1)

```
str facdpos_x                    // filename keeping position-text
cst login  = 1
cst status = 2
cst logout = 3
cst quit   = 4
int position
{
 !"Wellcome at the helpdesk"
 play "helpdesk.vox"

 !"check-in the line "+ nvtini_linenbr+ " in acd-queue"
 acdqueue nvtini_linenbr login
 postition= st
 while position> 0
  !"the queue position is: "+ position
  facdpos_x= "acdpos"+ position+ ".vox"
  play facdpos_x -cont
  if Hangup
   acdqueue nvtini_linenbr quit
   hangup
  fi
  acdqueue nvtini_linenbr status
  position= st
 wend

 !"your call will be transfered, please hold on the line"
 play "acddevi.vox" -cont
 if Hangup
  acdqueue nvtini_linenbr quit
  hangup
 fi
 dial ",&,100" -rings 16
 if st= 1
  !"connect successfull, logout out of acd-queue"
  acdqueue nvtini_linenbr logout
  hangup
 fi
 !"no connection, return to caller"
 bldial ",&,1"

 acdqueue nvtini_linenbr quit
 !"we can not answer your call now, please call later"
 play "acdquit.vox"
}
```

Category: ACD - Library

Verify the content of the acd-queue database

Launch the program acdinput.nvt either through new voice executer
[\nvt]nve.exe acdinput
or on a running line (Do not forget to exit the program properly!).
[\nvt]nvtque batchpost 1 acdinput

Programming code

```
// acdinput.nvt   program to show and modify all queue status
{
 !"Start of module [acdinput]"
 !"scan all lines from 1 to 8..."
 nvtini_i1= 1
 while nvtini_i1< 9
  acdqueue nvtini_i1 2
  !"line "+ nvtini_i1+ ": "+ st
  nvtini_i1= nvtini_i1+ 1
 wend

 !"enter line number: "+ nvtini_linenbr+ " (actual)  or exit with 0"
 ?nvtini_linenbr
 while nvtini_linenbr> 0
  !"enter status:  (1: acd in, 2: status?, 3: acd logout, 4: acd quit, 0: return"
  ?nvtini_i1
  while nvtini_i1< 5 and nvtini_i1> 0
   acdqueue nvtini_linenbr nvtini_i1
   if st< 0
    !"error position: "+ st
   fi
   if st= 0
    !"you are the first in queue, position: "+ st
   fi
   if st> 0
    !"there are "+ st+ " other persons waiting longer then you"
   fi
   !"enter status:  (1: acd in, 2: status?, 3: acd logout, 4: acd quit, 0: return"
   ?nvtini_i1
  wend
  !"enter line number: "+ nvtini_linenbr+ " (actual) "
  ?nvtini_linenbr
 wend
}
```

Name:	acdqueue <logical_group> <field>	
Inputs:	int logical_group	references the group-entrie (1: 450, 2: 460 etc.)
	int field	selects one information
	field= 11 (1)	(13: Calls in Queue,
	field= 12 (2)	17: Operators Waiting)
	field= 13 (3)	
	:	
	field= 18 (8)	
Options:		none
Returns:	st	value of field
	st= -1	the group does not exist
	st= -9	no access to nvtacd.dbf
Includes:	nvtcom.exe	ACD-Comport-Reader
	nvtacd.dbf	ACD-Queue-DataBase
	nvtacd.exe	to set group in nvtacd.dbf

Category: ACD - Library New Voice Tool - Swiss

Description

There are some pabx (i.e. Meridien) sending information about the acd-groups to a dumb terminal. NVTCOM.EXE ACDINFO is able to read this structure (through a com-port) and stores the information in the corresponding entry-field of nvtacd.dbf.

The function acdqueue allowes to get these informations into the application.

Note

Please note, that the group-numbers must be stored into the field group of nvtacd.dbf (by using nvtacd.exe) before starting NVTCOM.EXE ACDINFO.

Please find the detailed information about the input stream on the next page.

If this structure does not correspond to your needs, please contact your NVT reseller.

Please note, that the field 1 can be accessed through the fieldnumber 11. field 2 with 12, field 3 with 13 and so on!

Category: ACD - Library

Input Stream on Comport from PABX

```
ACD          #CALLS   #POS   #POS #POS #POS  #POS   #VIRTUAL
 DN    TSF ASA IN QUEUE MANNED  DCP  PCP  WTG NON-ACD CALLS QD
430    632  2     0      5     0    4    1    0      0
450    484  14    0      3     0    2    1    0      0
460     0   0     0      1     0    1    0    0      0
530     0   0     0      0     0    0    0    0      0
550     0   0     0      0     0    0    0    0      0

<group> <1> <2>   <3>    <4>    <5>  <6>  <7>  <8>
```

<group> the group number must be defined in nvtacd.dbf by using nvtacd.exe

<3> #calls in queue: number of caller waiting in the queue
<7> #pos wtg: number of free operators

This information is sent from the pabx every 10 or 20 seconds.

Start on nvtcom.exe acdinfo

[c:\nvt] nvtcom.exe acdinfo com1 1 5

This command reads the data from the com-port 1 and goes for the group-numbers stored in the entry 1, 2, 3, 4, and 5 of the database nvtacd.dbf

The entries 1, 2, 3, 4, 5 are corresponding to the groupnumbers 430, 450, 460, 530, 550.
This is set with the command nvtacd.exe .

Set the groupnumbers

[c:\nvt] nvtacd.exe 1 group 430
[c:\nvt] nvtacd.exe 2 group 450
[c:\nvt] nvtacd.exe 3 group 460

This sets the groupnumbers 430, 450, 460, 530, 550 to the entries 1, 2, 3, 4, 5.

Category: ACD - Library

Verify the information comming from the pabx

... Voice Mail System ... ... ACD Integration ...

Usage: nvtacd from_num [to_num] [field_name new_value]

use> nvtacd.exe 1 group 450 to set groupnumber (1) to 450
use> nvtacd.exe 2 group 460 to set groupnumber (2) to 460
use> nvtacd.exe 3 group 470 to set groupnumber (3) to 470

all group-numbers must be set before nvtcom.exe acdinfo is started !

use> nvtacd.exe 1 info1 111 to set info-element 1 to 111
use> nvtacd.exe 1 info8 333 to set info-element 8 to 333

use> nvtacd.exe 1 4 to show information about group 1 to 4

Example

```
// nvtacd.nvt   program to show the queue status  NVT - 9.1995
cst calls_in_queue = 13
cst pos_waiting    = 17

cst group_430 = 1
cst group_450 = 2
cst group_460 = 3
cst group_530 = 4
cst group_550 = 5
{

  !"check-in the group 430: "+ group_430+ " and cis: field 3"
  acdqueue  group_430  calls_in_queue
  !"calls in queue 430: "+ st

  !"check-in the group 430: "+ group_430+ " and pw: field 7"
  acdqueue  group_430  pos_waiting
  !"position waiting 430: "+ st
}
```

bldial

blind dial a sequence

Name: bldial <dial_string>
Inputs: str dial_string — keeps the string to dial

Returns: none

Includes: none

Category: Common Vox - Commands — New Voice Tool - Swiss

❒ Description

This function dials the given string.

Dial-Type	Number	pause	flash	special			set mode
Pulse Mode:	"0"-"9"",","	"&"			"P"	"T"	"M"
DTMF Mode:	"0"-"9"",","	"&"	"*", "#", "a"-"d"		"P"	"T"	"M"

❒ Example

```
/* transfers the call to the operator (int. 111)     */

{
  !"hi, you'll be transferred to the operator"
  bldial  "&, 111"

  sleep 40          /* wait for called person answers */
  !"hi, there is a call for you, hold on the line, please"
  hangup
}
```

❒ ISDN / QSIG

If you integrate this command in an ISDN environment with QSIG protocol,
bldial “!hello world”
starting with ‘!’ will print this information on the display of phone-terminals. Note, that this information will not be displayed on every phone and may be refreshed by the pbx.

Name:	dial <dial_string> [-options]	
Inputs:	str dial_string	keeps the string to dial
Options:	-rings n	rings before 'no answer'
Returns:	st= 0	no answer
	st= 1 called person answered	
	st= 2 called phone is occupied	
	st= 3	no ringback
	st= 4	operator intercept
	st= 5	error level 1
	st= 6	call analisys stopped
	st= 7	error level 2
Includes:		none
Category:	Common Vox - Commands	New Voice Tool - Swiss

❒ Description

This function waits for dial-tone, dials the given string and starts call analisys.

Dial-Type	Number	pause	flash	special	*	set mode
Pulse Mode:	"0"-"9"",","	"&"		";"	"P" "T" "M"	
DTMF Mode:	"0"-"9"",","	"&"	"*", "#", "a"-"d"	";"	"P" "T" "M"	

Wait for dial-tone* as first character. Alternatively use "L"ocal, "I"nternational or e"X"tra.

❒ Example

```
/* transfers the call to the operator (int. 111) and waits for 4 rings     */
{
  !"hi, you'll be transfered to the operator"
  bldial  "&,"
  dial  "111"  -rings 4
  if  st= 1
    !"hi, there is a call for you, hold on the line, please"
    hangup
  else
    bldial  "&,&"
    !"nobody answered on extention 111"
  fi
}
```

Name:	dial <dial_string> [-options]	
Inputs:	str dial_string	keeps the string to dial
Options:	-rings n	rings before 'no answer'
	-ownphone ownphone	in ISDN (CTR4) environment and pbx, transmitts the own phone, which will lead to a name on the display on the phone (phone-> name table).
	-ownphone “7001 Fire Alarm§Room 506	in ISDN (QSIG) environment and pbx, transmitts the own phone, and a text message ‘Fire Alarm. When answering the call, Room 506 will be showed (dep. phone and pbx)!
	-uui User_2_User_Signalling	This message will be sent and stored as mini-message (sms) in the DECT phone (up to 80 char, dep. phone and pbx)!
Returns:	st= 0	no answer
	st= 1	answer
	st= 2	called phone is occupied
	nvt_st= “220, unknown destination”	ISDN-error code and message
Special:	nvtini_s20	pdd: post dial delay
	nvtini_s19	This function allows to control the quality of the network. [D300] AlertingDelayVar=20 ProceedingDelayVar=19
Includes:		NVT.INI settings!
Category:	Common Vox - Commands	New Voice Tool - Swiss

❒ Description

In ISDN digital environment the proceeding is much faster and detection of answer secure.

❒ Note

Set DropCallOnOnHook=0, if the system should continue on non supported Notifications.
Set CNXtoPBX= 1 to transmits nvt_own_phone
While seting up an outbound call, you should see the following: -uui- -own- -owp- -alr- .

Name:	getkey [sequence] <number> [-options]	
Inputs:	str sequence	stores the entered digits
	int number	number of digits expected
Options:	-cont	continue after hangup
	-t[ime] t	max time to wait (1/10 sec)
Returns:	Hangup= 1	if caller went onhook
	Fastbusy= 1	if fast busy tone detected
	Loopdrop= 1	if loop drop detected
	st= 1	expected keys entered
	st= 4	loop current drop detected
	st= 6	timeout reached
	st= 7	fast busy tone detected
Includes:	key	if no sequence is given
Category:	Common Vox - Commands	New Voice Tool - Swiss

❒ Description

This function waits for a number of keystrokes.

❒ Example

```
/* waits for 4 keystrokes (max 3 sec)              */

str  entry         /* stores the entry of the caller */

{
  !"please enter your 4 - digit password"
  getkey  entry  4  -time  30  -cont
  !"der returncode lautet: <"+ st+ ">"
  if st= 1
    !"the password is: <"+ entry+ ">"
  fi
}
```

Name: offhook
Inputs: none

Returns: none

Includes: none

Category: Common Vox - Commands New Voice Tool - Swiss

❒ Description

This function puts the line offhook.

❒ Example

```
/* program with callback function */

str  phone_nbr       /* string to keep the entered nbr */

{
  !"hi, please enter your phonenumber"
  play "enternbr.vox"

  getkey  phone_nbr  15  -time 30
  !"the entered number is: <"+ phone_nbr+ ">"

  !"you'll be called back immediately"
  play "callback.vox"  -cont

  onhook
  sleep 10          /* wait before offhook          */
  offhook

  sleep 10          /* wait for dialtone            */
  bldial  phone_nbr

  sleep 40          /* wait for called person answers */
  !"hi, we are back"
  play  "back.vox"
}
```

Name: onhook
Inputs: none

Returns: none

Includes: none

Category: Common Vox - Commands New Voice Tool - Swiss

❒ Description

This function puts the line onhook.

❒ Example

```
/* program with callback function */

str  phone_nbr       /* string to keep the entered nbr */

{
  !"hi, please enter your phonenumber"
  play "enternbr.vox"

  getkey  phone_nbr  15  -time 30
  !"the entered number is: <"+ phone_nbr+ ">"

  !"you'll be called back immediately"
  play "callback.vox"  -cont

  onhook
  sleep 10          /* wait before offhook          */
  offhook

  sleep 10          /* wait for dialtone            */
  bldial  phone_nbr

  sleep 40          /* wait for called person answers */
  !"hi, we are back"
  play  "back.vox"
}
```

play

play a vox - file

Name:	play <filename> [directory] [-options]	
Inputs:	str filename	complete filename
	str directory	location of file
Options:	-cont	continue after hangup
	-k[eys] n	number of DTMF Tones
	-p[art] m	vox - part to play
	-a[scii] a	vox - part to play
Returns:	Notfound= 1	if vox-file not found, else 0
	Hangup= 1	if caller went onhook
	Fastbusy= 1	if fast busy tone detected
	Loopdrop= 1	if loop drop detected
	st= -1	if vox-file not found
	st= -1	if vox-file part nonexistent
	st= 1	termination on keystroke
	st= 4	loop current drop detected
	st= 7	fast busy tone detected
	st= 10	normal termination
Includes:		none
Category:	Common Vox - Commands	New Voice Tool - Swiss

❒ Description

This function plays the given vox - file.

❒ Example

/* plays a vox - file and continue after onhook */

```
{
  play  "carmina.vox"  "c:\nvt\"  -cont
  !"the returncode is: <"+ st+ ">"
}
```

❒ Note

Depending the telephony-interface vox and wave-file types are supported. Typical formats are
for vox: VBase 40 Format with 0: 6kHz 4bit, 1: 8kHz 4bit, 2: 6kHz 8bit 3: 8kHz 8 bit.
for wav: 11kHz 8bit.

Name:	play <filename> [directory] [-options]	
Inputs:	str filename	complete filename
	str directory	location of file
Options:	-spec 1	fast forward / backward
	-spec 2	do not delete dtmf buffer
	-spec 3	= -spec 1 and -spec 2
Returns:	Notfound= 1	if vox-file not found, else 0
	Hangup= 1	if caller went onhook
	Fastbusy= 1	if fast busy tone detected
	Loopdrop= 1	if loop drop detected
	st= -1	if vox-file not found
	st= -1	if vox-file part nonexistent
	st= 1	termination on keystroke
	st= 4	loop current drop detected
	st= 7	fast busy tone detected
	st= 10	normal termination
Includes:		none
Category:	Common Vox - Commands	New Voice Tool - Swiss

❒ Description

This function plays the given vox - file. A special mode has been introduced for dictaphone functions. Also it is now possible to start playing a voxfile without deleting the digits previously received in the digit buffer.

❒ Example

```
/* plays a vox - file in a dictaphone mode */

str filename
int i
{
  i= 1000
  while i< 1099
    filename= "memo"+ i+ ".vox"
    play  filename  -spec  1
    i= i+ 1
  wend
}
```

Name:	rec <filename> [directory] [-options]	
Inputs:	str filename	complete filename
	str directory	location of file
Options:	-cont	continue after hangup
	-k[eys] n	number of DTMF Tones
	-q[ality] q	"agc" | "8"
	-silence s	maximum silence (1/10 sec)
	-t[ime] t	max. record time (1/10 sec)
Returns:	Hangup= 1	if caller went onhook
	Fastbusy= 1	if fast busy tone detected
	Loopdrop= 1	if loop drop detected
	st= 1	termination on keystroke
	st= 2	maximum silence reached
	st= 4	loop current drop detected
	st= 6	maximum rec-time reached
	st= 7	fast busy tone detected
	st= 8	fax tone detected
Includes:	nvtini_maxrec	maximum record time
	nvtini_recqu	record quality 0.. 3
Category:	Common Vox - Commands	New Voice Tool - Swiss

❒ Description

This function records into the given vox - file.

❒ Example

/* record a message (max 30 sec) and continue after onhook */

```
{
  !"please speak after the beep"
  rec  "message.vox"  "c:\nvt\"  -time  300  -cont
  !"the returncode is: <"+ st+ ">"
  if Hangup= 0
    play  "message.vox" "c:\nvt\"
  fi
}
```

Name:	voldet [duration] [-options]	
Inputs:	int duration	time to wait for answer in 1/100 seconds
Options:	-nonsil n	minimum length of grunt in 1/10 of sec. (def: 3)
	-time m	maximum time to wait in 1/10 of sec. (def. 40)
Returns:	st= 0	silence / timeout
	st= 1	grunt detected / dtmf detected
	st= 2	error level 1
	st= 4	loop current drop detected
	st= 7	fast busy tone detected
	st= 8	fax tone detected
Includes:		none
Category:	Common Vox - Commands	New Voice Tool - Swiss

❒ Description

This function listens on the line, wether the caller says something or not.

❒ Example

/* ask the caller a question, and measure the answertime */

```
int duration

{
  !"hi, please say yes, if Napoleon was an emperor"
  voldet  duration
  if  st= 1
    !"you are right and answered after "+ duration+ "/100 sec"
  else
    !"sorry, but Napoleon was a french emperor"
  fi
}
```

waitring

wait for a number of rings

Name:	waitring <number> [-options]	
Inputs:	int number	number of rings to wait a negative number is equal to the option -nooffhook
Options:	-nooff[hook]	stay onhook after detection
	-time n	timeout during waitring
Returns:	st= 0	answer after ring detection
	st= 1	start after timeout
	Timeout= 0	Timeout not reached
	Timeout= 1	Timeout reached
Includes:		none
Category:	Common Vox - Commands	New Voice Tool - Swiss

❒ Description

This function waits for a number of rings

❒ Example

/* wait for a caller, or start an outgoing call (rings= 0)*/

```
{
 !"waiting for 2 rings or timeout after 10 minutes"
 waitring 2 -time 6000
 if Timeout= 1
   !"start outgoing call in the program outdial.nvt "
   do "outdial"
 else
   !"start the program playinfo.nvt for the caller"
   do "playinfo"
 fi
}
```

Name:	checkholiday <filename.nvh> [action]	
Name:	checkschedule <filename.nvh> [action]	
Inputs:	str filename.nvh	holiday-schedule
	str action	if matching: result - string
Returns:	st= 0	default value
	st= 1	entry corresponding
	st= -1	schedule file not found
Includes:	filename.nvh	schedule file
Category:	Schedule functions	New Voice Tool - Swiss

❒ Description

This function returns a string in function of the actual date and time.

To be able to play the right message to the caller, depending the day and hour of its call, the function CHECKHOLIDAY has been implemented. The function gets as input a filename, naming an ascii-data-file containing the holiday-shedule. The structure of the file is the following:

```
start-date , time  - stop-date  , time   , action
00.00.  00 , 00:00 - 00.00.  00 , 00:00  , string;
```

The file can be created on every text-editor und must be stored as raw ascii-text with new-lines. The delemitters '.' ',' ':' ';' '-' and the data must be on the right place.

On the control pannel you'll find the possability to create the demonstration-file holidemo.nvh which can be used as starter-file. The content of this file is:

```
rem   New Voice Tool Audiotex & Voicemail Application Generator
rem   Holiday example: holidemo.nvh
rem   Line 1 for default values, other lines: holiday
rem   start-date , time - stop-date , time , action
00.00.  00 , 00:00 - 00.00.  00 , 00:00 , default.vox;
31.12.  94 , 07:45 - 03.01.  95 , 18:00 , newyear.vox;
31.07.     , 16:00 - 02.08.     , 05:00 , august_1.vox;
```

The structures of the start-date and stop-date are: day.month.year
The structures of the start-time and stop-time are: hour:minute
If all values are set to 0, the action-field will be returned as default value, if no date-periode has matched the actual date

If the field of the year is empty, the date-periode is ok for every year, such as 1. of august.
The file is read top-down, that means, if several date-entries match the actual date, the action-field of the first corresponding entry will be returned.
If no date-entry match the actual date, the action-field of the **last** corresponding default-value-entry will be returned. This allows you to modify the default-values after certain entries.

Returncode: st = -1 if no entry matches or the file could not be found
Returncode: st = 0 if defaultvalue does match
Returncode: st = 1 if schedule periode does match

On the controlpannel you'll find the possability to verify the structure of any *.NVH file. This will help you to check your programs before running them.

❒ Example

```
/* play a greeting in function of date and time     */

str filename
{
  checkholiday  "holidemo.nvh"  filename

  play filename
}
```

❒ Example

```
/* play a greeting in function of date and time     */

str filename
{
  checkschedule  "schedul1.nvh"  filename

  play filename
}
```

Name:	checkweek <filename.nvw> [action]	
Inputs:	str filename.nvw	week-shedule
	str action	if matching: result - string
Returns:	st= 0	default value
	st> 0	day of corresponding entry
	st= -1	week file not found
Includes:	filename.nvw	week schedule file
Category:	Schedule functions	New Voice Tool - Swiss

❒ Description

This function returns a string in function of the actual date and time.

To be able to play the right message to the caller, depending the day and hour, the function CHECKWEEK has been implemented. The function gets as input a filename, naming an ascii-data-file containing the weekly-shedule. The structure of the file is the following:

```
day , start-time - stop-time , action
0   , 00:00      - 00:00     , string;
```

The file can be created on every text-editor und must be stored as raw ascii-text with new-lines. The delemitters '.' ,', ':' ';' '-' and the data must be on the right place.

On the control pannel you'll find the possability to create the demonstration-file weekdemo.nvw which can be used as starter-file. The content of this file is:

```
rem   New Voice Tool   Audiotex & Voicemail Application Generator
rem   Week schedule example: weekdemo.nvw
rem   Line 1+ 2 for default values, other lines: working hours
rem   day , start-time , stop-time , action
0 , 00:00 - 00:00 , offhour1.vox;
1 , 09:00 - 11:45 , monday_1.vox;
1 , 13:30 - 16:45 , monday_2.vox;
2 , 07:30 - 11:45 , tuesday1.vox;
2 , 13:30 - 16:45 , tuesday2.vox;
3 , 07:30 - 11:45 , wednes_1.vox;
3 , 13:30 - 16:45 , wednes_2.vox;
4 , 07:30 - 11:45 , thu_day1.vox;
4 , 13:30 - 16:45 , thu_day2.vox;
5 , 07:30 - 11:45 , fri_day1.vox;
5 , 13:30 - 16:45 , fri_day2.vox;
6 , 07:30 - 11:45 , sat_day1.vox;
7 , 00:00 - 23:59 , sunday_1.vox;
```

The day is numbered from 1 to 7 equal to monday to sunday.
The structures of the start-time and stop-time are: hour:minute
If all values are set to 0, the action-field will be returned as default value, if no date-periode has matched the actual date
If several date-entries match the actual date, the action-field of the first corresponding entry will be returned.
If no date-entry match the actual date, the action-field of the **last** corresponding default-value-entry will be returned. This allows you to modify the default-values after certain entries.
Returncode: st = -1 if no entry matches or the file could not be found
Returncode: st = 0 if defaultvalue does match
Returncode: st = 1.. 7 if week periode does match

On the control pannel you'll find the possability to verify the structure of any *.NVH file. This will help you to check your programs before running them.

❒ Example

```
/* play a greeting in function of date and time     */

str filename

{
  checkweek  "weekdemo.nvw"  filename

  play filename

  hangup
}
```

Name:	checkschedule <filename.nvh> [action]	
Inputs:	str filename.nvh	holiday-schedule
	str action	if matching: result - string
Returns:	st= 0	default value
	st= 1	entry corresponding
	st= -1	schedule file not found
Includes:	filename.nvh	schedule file
Category:	Schedule functions	New Voice Tool - Swiss

❒ Description

This function returns a string in function of the actual date and time.

To be able to play the right message to the caller, depending the day and hour of its call, the function CHECKSCHEDULE has been implemented. The function gets as input a filename, naming an ascii-data-file containing the holiday-shedule. The structure of the file is the following:

```
start-date , time  - stop-date  , time   , action
00.00.  00 , 00:00 - 00.00.  00 , 00:00  , string;
```

The file can be created on every text-editor und must be stored as raw ascii-text with new-lines. The delemitters '.' ',', ':' ';' '-' and the data must be on the right place.

On the control pannel you'll find the possability to create the demonstration-file holidemo.nvh which can be used as starter-file. The content of this file is:

```
rem   New Voice Tool Audiotex & Voicemail Applikation Generator
rem   Schedule example: schedule.nvh
rem   Line 1 for default values, other lines: Working days
rem   start-date , time - stop-date , time , action
00.00.  00 , 00:00 - 00.00.  00 , 00:00 , default.vox;
31.12.  94 , 07:45 - 03.01.  95 , 18:00 , specialx.vox;
31.07.     , 16:00 - 02.08.     , 05:00 , special1.vox;
```

The structures of the start-date and stop-date are: day.month.year
The structures of the start-time and stop-time are: hour:minute
If all values are set to 0, the action-field will be returned as default value, if no date-periode has matched the actual date

If the field of the year is empty, the date-periode is ok for every year, such as 1. of august.
The file is read top-down, that means, if several date-entries match the actual date, the action-field of the first corresponding entry will be returned.
If no date-entry match the actual date, the action-field of the **last** corresponding default-value-entry will be returned. This allows you to modify the default-values after certain entries.

Returncode: st = -1 if no entry matches or the file could not be found
Returncode: st = 0 if defaultvalue does match
Returncode: st = 1 if schedule periode does match

On the control pannel you'll find the possability to verify the structure of any *.NVH file. This will help you to check your programs before running them.

❒ Example

```
/* play a greeting in function of date and time     */

str filename

{

  checkschedule  "working.nvh"  filename

  play filename

  hangup
}
```

Name:	dbappend	
Inputs:		none
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:		none
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function appends a new record to the opened database.

❒ Example

/* appends a new record at the end of vm001.dbf */

```
{
 dbuse  "vm001"  "c:\nvt\db\"
 if  dbok
   dbbot
   dbappend
   if  dbok
     dbput  "T02"  1
   else
     !"no new record can not be appended"
   fi
   dbclose
 else
  !"the database vm001.dbf can not be opened"
 fi
}
```

dbbottom

goes to the last record

Name:	dbbot[tom]	
Inputs:		none
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:		none
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function goes to the last record of the database.

❒ Example

```
/* gets the record number of the last record of vm001.dbf */

int  record_number      /* integer to keep the record nbr */

{
  dbuse  "vm001"  "c:\nvt\db\"
  if  dbok
    dbbot
    if  dbok
      dbrecno  record_number
      !"the record number is <"+ record_number+ ">"
    else
      !"error going to the bottom of the database"
    fi
    dbclose
  else
   !"the database vm001.dbf can not be opened"
  fi
}
```

dbclose

close the opened database

Name:	dbclose	
Inputs:		none
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:		none
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function closes the database opened previosly with dbuse.

❒ Example

```
/* gets the status of the top mailbox of vm001.dbf */

int  status          /* integer to keep the status */

{
  dbuse  "vm001"  "c:\nvt\db\"
  if  dbok
    dbtop
    dbget  "T02"  status
    if  dbok
      !"the status is <"+ status+ ">"
    else
      !"error reading the status"
    fi
    dbclose
  else
   !"the database vm001.dbf can not be opened"
  fi
}
```

dbdelete

delete the active record

Name:	dbdel[ete]	
Inputs:		none
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:		none
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function deletes the active record of the opened database.

❒ Example

/* deletes the first record of vm001.dbf */

```
{
  dbuse  "vm001"  "c:\nvt\db\"
  if  dbok
    dbtop
    dbdel
    if  dbok
      !"the top record is deleted"
    else
      !"the top record can not be deleted"
    fi
    dbclose
  else
   !"the database vm001.dbf can not be opened"
  fi
}
```

dbget

get the value of a field

Name:	dbget <fieldname> [str_var | int_var]	
Inputs:	fieldname	fieldname in the database
	str_var	variable to store the string
	int_var	variable to store the integer
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:	dbstr	default result string
	dbint	default result integer
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function gets the value of a field of the opened database. The record is the active record
set with dbtop, dbbot, dbseek, dbgo or dbskip.

❒ Example

```
/* gets the status of the top mailbox of vm001.dbf */

int  status          /* integer to keep the status */

{
  dbuse  "vm001"  "c:\nvt\db\"
  if  dbok
    dbtop
    dbget  "T02"  status
    if  dbok
      !"the status is <"+ status+ ">"
    else
      !"error reading the status"
    fi
    dbclose
  else
   !"the database vm001.dbf can not be opened"
  fi
}
```

dbgo

goes to the specified record

Name:	dbgo <record_nubmer>	
Inputs:	int record_number	specified record number
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:		none
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function goes to the specified record of the database.

❒ Example

```
/* goes to the record number of vm001.dbf */

int  record_number      /* integer to store the record nbr */

{
 dbuse  "vm001"  "c:\nvt\db\"
 if  dbok
   record_number=  111
   dbgo  record_number
   if  dbok
     !"the active record is <"+ record_number+ ">"
   else
     !"error going to the record <"+ record_number+ ">"
   fi
   dbclose
 else
  !"the database vm001.dbf can not be opened"
 fi
}
```

❒ Limits - the record number is the number given during creation time and does not allways correspond with the number of the record in the database.

Name:	dblock	
Inputs:		none
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:		none
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function locks the database opened previosly with dbuse.

❒ Example

```
/* gets the status of the top mailbox of vm001.dbf */

int  status          /* integer to keep the status */

{
  dbuse  "vm001"  "c:\nvt\db\"
  if  dbok
    dbtop
    status= 1
    dblock
      dbput  "T02"  status
      if  dbok
        !"the status is set to 1"
      else
        !"error writing the status"
      fi
    dbunlock
    dbclose
  else
   !"the database vm001.dbf can not be opened"
  fi
}
```

Name: dbput <fieldname> [str_var | int_var]
Inputs: fieldname fieldname in the database
str_var variable keeping the string
int_var variable keeping the integer

Returns: dbok= 1 if successfull
dbok= 0 if failure

Includes: dbstr default string
dbint default integer

Category: Database - Commands New Voice Tool - Swiss

❒ Description

This function puts a value into a field of the opened database. The record is the active record
set with dbtop, dbbot, dbseek, dbgo or dbskip.

❒ Example

```
/* sets the status of the top mailbox of vm001.dbf */

int  status          /* integer to keep the status */

{
 dbuse  "vm001"  "c:\nvt\db\"
 if  dbok
   dbtop
   status= 1
   dbput  "T02"  status
   if  dbok
     !"the status is set to 1"
   else
     !"error writing the status"
   fi
   dbclose
 else
  !"the database vm001.dbf can not be opened"
 fi
}
```

dbrecno

gets the record number

Name:	dbrecno <record_number>	
Inputs:	int record_number	variable to return the value
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:		none
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function gets the record number of the actual record of the database.

❒ Example

```
/* gets the record number of the last record of vm001.dbf */

int  record_number      /* integer to keep the record nbr */

{
  dbuse  "vm001"  "c:\nvt\db\"
  if  dbok
    dbbot
    if  dbok
      dbrecno  record_number
      !"the record number is <"+ record_number+ ">"
    else
      !"error going to the bottom of the database"
    fi
    dbclose
  else
   !"the database vm001.dbf can not be opened"
  fi
}
```

❒ Limits - the record number is the number given during creation time und does not allways correspond with the number of the record in the database.

dbseek

seek an entry through index

Name:	dbseek <index>	
Inputs:	str index	string index to be found
	int index	integer index to be found
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:	autoopen	environment variable
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function seeks a record through the index.

❒ Example

```
/* seeks the record with the given index in vm001.dbf */

int  status          /* integer to store the status */

{
 dbuse  "vm001"  "c:\nvt\db\"
 if  dbok
   dbtop
   dbseek  1000
   if  dbok
     get  "T02"  status
     !"the status is <"+ status+ ">"
   else
     !"error seeking the record index 1000"
   fi
   dbclose
 else
  !"the database vm001.dbf can not be opened"
 fi
}
```

❒ Limits - set autoopen= 1, as only databases opened with index can be 'seeked'.
Use dBase III files with index file <filename>.ndx
Write the <filename> of the index-file into <filename>.cgp

dbsel

select database 1.. 9

Name:	dbsel number	
Inputs:	number	number of database 1.. 9
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:	1	default value
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

New Voice Tool allowes to open several databases in parallel. By selecting a database number previous to any database command, New Voice Tool will apply this function to selected database. By default, the database 1 is active.

❒ Example

```
/* This program copies the field nvtini_s1 of nvtini.dbf to nvtbatch.dbf  */
{
  dbuse  "nvtini.dbf"  "c:\nvt\"
  dbtop
  dbget  “T34“  nvtini_s1
  dbsel 2
  dbuse  "nvtbatch.dbf"  "c:\nvt\"
  dbtop
  dbput  “T34“  nvtini_s1
  dbskip
  while dbok
    dbsel 1
    dbskip
    dbget  “T34“  nvtini_s1
    dbsel 2
    dbput  “T34“  nvtini_s1
    dbskip
  wend
  dbclose
  dbsel 1
  dbclose
}
```

Name:	dbskip [number]	
Inputs:	number	skip number of records
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:	1	default value
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function skips a given number of records. If no number is given, the function will skip to the next record.

❒ Example

```
/* skips through the database of vm001.dbf        */
int  status          /* integer to keep the status */
{
  dbuse  "vm001"  "c:\nvt\db\"
  if  dbok
    dbtop
    dbget  "T02"  status
    while  dbok  and  status < 1
      dbskip  1
      if dbok
        dbget  "T02"  status
      else
        !"end of database reached, status 1 not found"
      fi
    wend
    dbclose
    if status= 1
      !"status 1 found"
    else
      !"status 1 not found"
    fi
  else
   !"the database vm001.dbf can not be opened"
  fi
}
```

dbtop

goes to the top record

Name:	dbtop	
Inputs:		none
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:		none
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function goes to the top record of the database.

❒ Example

```
/* gets the status of the top mailbox of vm001.dbf */

int  status          /* integer to keep the status */

{
  dbuse  "vm001"  "c:\nvt\db\"
  if  dbok
    dbtop
    dbget  "T02"  status
    if  dbok
      !"the status is <"+ status+ ">"
    else
      !"error reading the status"
    fi
    dbclose
  else
   !"the database vm001.dbf can not be opened"
  fi
}
```

dbunlock

unlock the locked database

Name: dbunlock
Inputs: none

Returns: dbok= 1 if successfull
dbok= 0 if failure

Includes: none

Category: Database - Commands New Voice Tool - Swiss

❒ Description

This function unlocks the database locked previosly with dblock.

❒ Example

```
/* gets the status of the top mailbox of vm001.dbf */

int  status          /* integer to keep the status */

{
  dbuse  "vm001"  "c:\nvt\db\"
  if  dbok
    dbtop
    status= 1
    dblock
      dbput  "T02"  status
      if  dbok
        !"the status is set to 1"
      else
        !"error writing the status"
      fi
    dbunlock
    dbclose
  else
   !"the database vm001.dbf can not be opened"
  fi
}
```

dbuse

opens a database

Name:	dbuse <filename> [directory]	
Inputs:	str filename	filename without extention
	str directory	location of filename
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:	autoopen	environment variable
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function opens the given database. If the indexfile and the cgp-file is located in the same directory, as the database, the database will be opened with the index and so, the seek-command will be allowed.

To open several databases in parallel, switch between databases by application of the function dbsel. Up to 9 databases can be opened in parallel.

❒ Example

```
/* gets the status of the top mailbox of vm001.dbf */

int  status          /* integer to keep the status */

{
 dbuse  "vm001"  "c:\nvt\db\"
 if  dbok
   dbtop
   dbget  "T02"  status
   if  dbok
     !"the status is <"+ status+ ">"
   else
     !"error reading the status"
   fi
   dbclose
 else
  !"the database vm001.dbf can not be opened"
 fi
}
```

❒ Limits To open an indexed file, the name of the index must be written in the <filename>.cgp - file.

Name: fclose
Inputs:

Returns: st= 0 if successfull
st= -1 if failure

Includes: none

Category: File Functions New Voice Tool - Swiss

r Description

This function closes the ASCII file previous opened with fopen for read or for write.

r Example

```
/* opens an ascii file, writes and reads to verify  */

str text1 text2

{
  fopen "file.txt" -dir "c:\nvt\" "write"
  fwrite "text line 1"
  fwrite "text line 2"
  fclose
  !"file.txt created"

  text1= ""

  fopen "file.txt" -d "c:\nvt\" "read"
  fread text1
  fread text2
  fclose

  ! text1
  ! text2
}
```

fcopy copies a file into another one

Name:	fcopy <filename> [-option] <target_filename> [-option2]	
Inputs:	str filename str target_filename	source filename target filename
Options:	-dir directory	location of filename
Options2:	-dir directory -time 1/10_sec	location of target-filename timeout on file locked
Returns:	st= 0 st= -1	if successfull if failure
Includes:	none	
Category:	File Functions	New Voice Tool - Swiss

---

r Description

This function copies the given file to another file. The option -time declares the maximum time to wait on file locked or target-file-existence..

r Example

/* copies a file to another file */

str fname fname2

```
{
  !"copies info1001.vox from \dtvox to \frvox
  fcopy  fname -dir nvtini_lg1  fname -dir nvtini_lg2  -time 30
}
```

fdelete

deletes the given file

Name:	fdelete <filename> [-option]	
Inputs:	str filename	filename
Options:	-dir directory	location of filename
	-time 1/10_sec	timeout on file locked
Returns:	st= 0	if successfull
	st= -1	if failure
Includes:	none	
Category:	File Functions	New Voice Tool - Swiss

r Description

This function deletes the given file. The option -time declares the maximum time to wait on file locked or nonexistent.

r Example

/* deletes the vox-file corresponding to the mail-message deleted by the customer */
str fname

```
proc play_recorded_messages{
  while 1
    !“press 4 to delete the message, press 2 to replay"
    getkey key 1 -time 30
    if key= “#“
      return
    fi
    if key= 4
      fdelete fname -dir nvtini_dbv1 -time 30
      playmbx mbx -delete fname
      if st< 0
        !“this was the last message“
        return
      fi
      !“play next message“
      play fname nvtini_dbv1
    fi
    if key= 2
      play fname nvtini_dbv1
    fi
  wend
}
```

Name:	ffind <filename> [-option]	
Inputs:	str filename	filename
Options:	-dir directory	location of filename
	-fast	no open, just tests presence
	-size str_size	return the filesize in Bytes
	-time str_time	returns create/modif-date
Returns:	st>= 0	if present
	st= -1	if not found
Includes:		none
Category:	File Functions	New Voice Tool - Swiss

r Description

This function seeks a file in the local or given directory.

r Example

```
/* checks the presence of a file, then plays the message  */

str filename

{
  filename= "personal.vox"

  ffind filename

  if st= 0
     play  filename
  else
     play  "standart.vox"
  fi
}
```

Name:	ffind <filename> [-option]	
Inputs:	str filename	filename
Options:	-dir directory	location of filename
	-fast	no open, just tests presence
	-size str_size	return the filesize in Bytes
	-time str_time	returns create/modif-date
Returns:	st>= 0	if present
	st= -1	if not found
Includes:		none
Category:	File Functions	New Voice Tool - Swiss

r Description

This function seeks a file and returns file size and creation and modification date.
To get the date of creation, just store in str_time the value “nvt_creation_time”

r Example

```
str s0 s1 s2
int d h m s
{
  !"-------------------file creation and modification--------"
  s1= "nvt_creation_time"
  ffind "difftime.nvt" -time s1
  !"creation date and time: "+ s1
  s2= ""
  ffind "difftime.nvt" -time s2
  !"  modify date and time: "+ s2
  get -difftime s1 s2
  !"difference between creation and last modif: "+ st
  st= st* -1
  m= st/ 60
  s= st- (m* 60)
  h= m/ 60
  d= h/ 24
  h= h- (d* 24)
  m= st/60 - (h* 60)- (d*24*60)
  !"delta: "+d+" days "+h+" hours "+m+" minutes "+s+" seconds"
  st= d*24*60*60+h*60*60+m*60+s
  !"difference in seconds: "+ st+ " seconds"
```

```
  !"-------------------time calculation-------------------"
  get -time s0 s1
  !"time since 1970: "+ s1+  "  sleep 2 seconds..."
  sleep 20
  get -difftime s1
  !"difference time to now: "+ st+  "    (expected: 2 seconds)"
  get -time s0 s1
  !"time since 1970: "+ s1+  "  sleep 3 seconds"
  sleep 30
  get -time s0 s2
  !"get time since 1970: "+ s2
  get -difftime s1 s2
  !"difference time s1- s2: "+ st+  "    (expected: -3 seconds)"
}
```

Name:	fmove <filename> [-option] <target_filename> [-option2]	
Inputs:	str filename str target_filename	source filename target filename
Options:	-dir directory	location of filename
Options2:	-dir directory -time 1/10_sec	location of target-filename timeout on file locked
Returns:	st= 0 st= -1	if successfull if failure
Includes:	none	
Category:	File Functions	New Voice Tool - Swiss

r Description

This function moves the given file to another location with optionally another name. The option -time declares the maximum time to wait on file locked or target-file-existence..

r Example

/* moves a file to another location */

str fname

```
{
  !“move info1001.vox from \dtvox to \frvox
  fmove  fname -dir nvtini_lg1  fname -dir nvtini_lg2  -time 30
}
```

fopen

opens an ASCII file

Name:	fopen <filename> [-option] <read | write>	
Inputs:	str filename	filename
	str "read"	open the file for read
	str "write"	open the file for write
Options:	-dir directory	location of filename
	-time 1/10_sec	retry on nonexist or locked
Returns:	st= 0	if successfull
	st= -1	if failure
Includes:		none
Category:	File Functions	New Voice Tool - Swiss

r Description

This function opens the given ASCII file for read or for write.

r Example

```
/* opens an ascii file, writes and reads to verify  */

str text1 text2

{
  fopen "file.txt" -dir "c:\nvt\" "write"
  fwrite "text line 1"
  fwrite "text line 2"
  fclose
  !"file.txt created"

  text1= ""

  fopen "file.txt" -d "c:\nvt\" "read"
  fread text1
  fread text2
  fclose

  ! text1
  ! text2
}
```

fread

reads from an ASCII file

Name:	fread <text>	
Inputs:	str text	read line written into text
Returns:	st= 0	if successfull
	st= -1	if failure
Includes:		none
Category:	File Functions	New Voice Tool - Swiss

r Description

This function reads from an opened ASCII file (mode must be 'read').

r Example

```
/* opens an ascii file, writes and reads to verify  */

str text1 text2

{
  fopen "file.txt" -dir "c:\nvt\" "write"
  fwrite "text line 1"
  fwrite "text line 2"
  fclose
  !"file.txt created"

  text1= ""

  fopen "file.txt" -d "c:\nvt\" "read"
  fread text1
  fread text2
  fclose

  ! text1
  ! text2
}
```

frename

renames file to another name

Name:	frename <filename> [-option] <target_filename> [-option2]	
Inputs:	str filename str target_filename	source filename target filename
Options:	-dir directory	location of filename
Options2:	-dir directory -time 1/10_sec	location of target-filename timeout on file locked
Returns:	st= 0 st= -1	if successfull if failure
Includes:	none	
Category:	File Functions	New Voice Tool - Swiss

r Description

This function renames the given file to another anme with optionally another directory. The option -time declares the maximum time to wait on file locked or target-file-existence..

r Example

/* renames a file to another name */

str fname fnewname

```
{
  !“renames info1001.tmp in \dtvox into info1001.vox
  fmove  fname -dir nvtini_lg1  fnewname -dir nvtini_lg1  -time 30
}
```

Name:	fwrite <text>	
Inputs:	str text	line to write to the file
Returns:	st= 0	if successfull
	st= -1	if failure
Includes:		
	none	
Category:	File Functions	New Voice Tool - Swiss

r Description

This function writes to an opened ASCII file (mode must be 'write').

r Example

```
/* opens an ascii file, writes and reads to verify  */

str text1 text2

{
  fopen "file.txt" -dir "c:\nvt\" "write"
  fwrite "text line 1"
  fwrite "text line 2"
  fclose
  !"file.txt created"

  text1= ""

  fopen "file.txt" -d "c:\nvt\" "read"
  fread text1
  fread text2
  fclose

  ! text1
  ! text2
}
```

fwriteln

append a textline to a file

Name:	fwriteln <filename> [-option] <text>	
Inputs:	str filename	file to append line
	str text	line to append to the file
Returns:	st= 0	if successfull
	st= -1	if failure
Includes:		none
Category:	File Functions	New Voice Tool - Swiss

r Description

This function writes a text-string to an ASCII file. The line is appended at the end of the file.

r Example

```
/* writes a text-string to an ascii-file (mode: append) */

str text1 text2

{
  fopen "file.txt" -d "c:\nvt\" "read"
  fread text1
  fread text2
  fclose

  ! text1
  ! text2

  fwriteln "filecopy.txt" text1
  fwriteln "filecopy.txt" text2
}
```

Name:	sendfax <fax-filename> [options]	
Inputs:	str fax-filename	filename of fax document
	-dir str_directory	directory of fax-file
	-poll	send document on pollrequest
	-rate int_rate	2400, 4800, 7200, 9600...
Returns:	st>= 1	number of transmitted page
	st= -1	error sending document
Includes:		Fax Hardware / Software
Category:	Fax - Commands	New Voice Tool - Swiss

❒ Description

This function sends a fax-document. The document format depends to the hardware and software installed on your system. The option -poll allowes to let the caller poll a document. The option -rate allowes to slow down the transmitting speed for high quality information on pay-per-call lines.

❒ Example

```
/* NVT Fax broadcast              New Voice Fax 1995 */
/* NVF gets the phonenumber out of a database        */

str phonenumber
cst ok=  1
cst ko= -1
{
 dbuse  "client.dbf"
 dbtop
 dbget  "PHONE"   phonenumber
 offhook
 dial  phonenumber -rings 8
 if  st= 1
   sendfax  "infotext.fax"
   dbput  "STATUS"  ok
 else
   dbput  "STATUS"  ko
 fi
 dbclose
}
```

❒ Note

Whenever the caller sends a document while yur application tries to send a document on poll-request (fax-polling) the document will be stored in “.\RECFXnn.TIF“ . Where nn is the actual linenumber (nvtini_linenbr). Rename the file if you need the document later on.

Name:	recfax <fax-filename> [options]	
Inputs:	str fax-filename	filename of fax document
	-dir str_directory	directory of fax-file
	-poll	the document will be polled
Returns:	st>= 1	number of received pages
	st= -1	error receiving document
Includes:		Fax Hardware / Software
Category:	Fax - Commands	New Voice Tool - Swiss

❒ Description

This function receives a fax-document. The document format depends to the hardware and software installed on your system. The option -poll allowes to receive a document as call initiator.

❒ Example

```
/* NVT Fax poll - receive a fax        New Voice Fax 1995 */
/* This program will be started in Batch-mode            */

str phonenumber

{
 strlen  nvtini_s1
 if st > 5
   phonenumber= nvtini_s1
   !“the phone number to dial is: „+ nvtini_s1
 else
   phonenumber=  "0041,1,3641991"
   !“dial alternate number, as „+ nvtini_s1+ „ is too short“
 fi
 offhook
 dial  phonenumber -rings 8
 if  st= 1
   recfax  "polltext.fax"  -poll
 else
   !"the fax-sender-station is busy or out of order"
 fi
}
```

Name:	sendfax <fax-filename> [options]	
Inputs:	str fax-filename	filename of fax document
	-first int_pagenumber	indicates the first page sent valid for multipage tif-files
	-special...str_var_control	pass aditional information
Returns:	st>= 1	number of transmitted page
	st= -1	error sending document
Includes:		Fax Hardware / Software NVTFAX.INI
Category:	Fax - Commands	New Voice Tool - Swiss

❒ Description

The file NVTFAX.INI helps you to define the headder line or a headder part. This information can be overwritten by passing the information through the string variable and the option -special

❒ NVTFAX.INI

```
<LOCALID=0041,1,364,2652>
<TXHEADDER32=NVT-FAX/CH-TX>
<TXHEADDER80=>
<RXHEADDER32=NVT-FAX/CH-RX>
<RXHEADDER80=>
<HEADDERFILE=NEWVOICE.TIF>
```

❒ Example

```
/* NVT Fax broadcast              New Voice Fax 1995 */
/* NVF adds an ascii-file as headder to page 3 of info.tif*/

cst  phone = "0041,1,242 11 78"
str  special
{
  offhook
  dial  phone  -rings  6
  if st = 1
    special= "<headderfile=newvoice.txt><localid=013642652>"
    sendfax  "info.tif"   -first 3   -special  special
    !st
  else
    !"no answer on "+ phone
  fi
}
```

Name:	sendfax <fax-filename> [options]	
Inputs:	str fax-filename	filename of fax document
	-first int_pagenumber	indicates the first page sent valid for multipage tif-files
	-special...str_var_control	pass aditional information
Returns:	st>= 1	number of transmitted page
	st= -1	error sending document
Includes:		Fax Hardware / Software NVTFAX.INI
Category:	Fax - Commands	New Voice Tool - Swiss

❒ Description

The file NVTFAX.INI helps you to define the headder line or a headder part. This information can be overwritten by passing the information through the string variable and the option -special

❒ NVTFAX.INI

```
<LOCALID=0041,1,364,2652>
<TXHEADDER32=NVT-FAX/CH-TX>
<TXHEADDER80=>
<RXHEADDER32=NVT-FAX/CH-RX>
<RXHEADDER80=>
<HEADDERFILE=NEWVOICE.TIF>
```

❒ Example

```
/* NVT Fax broadcast                New Voice Fax 2000 */
/* NVF configures the fax header for gamma-fax board */
cst  phone = "0041,1,242 11 78"
{
  offhook
  dial  phone  -rings  6
  if st = 1
    if nvtini_i1= 1
      sendfax  "info.tif"   -first 3   -special  “<HEADER=[none]“
    else
      sendfax  "info.tif"   -first 3   -special  “<HEADER=0041 1 242 11 78><OVLHDR=Y>“
    fi
    !st
  else
    !"no answer on "+ phone
  fi
}
```

Name:	do <program>	
Inputs:	str program	name of the program to start (without extention).
Returns:	st= 0	program.nve present
	st= -1	program.nve not found
Includes:		none
Category:	Program Calls	New Voice Tool - Swiss

❒ Description

Programs allowes you to groupe your code into small units. Programs also allowes you to reuse a part of your code at different moments without rewriting it.

❒ Example

```
/* readfile.nvt: reads lines from a given file */
{
  fopen nvtini_s4 "read"
  fread nvtini_s1
  fread nvtini_s2
  fclose
}

/* main.nvt: calls readfile.nvt to read lines from a file */
{
  !"please enter 1 for gold or 2 for silver"
  getkey  1

  if key= "1"
    nvtini_s4= "gold.txt"
  else
    nvtini_s4= "silver.txt"
  fi
  do  "readfile"
  !"line1: <"+ nvtini_s1+ "> line2: <"+ nvtini_s2+ ">"
}
```

❒ **Limits:** Only the predefined variables nvtini_* and nvt_* are passed between the programs.

Name: proc <procedure>
Inputs: str procedure — name of the procedure

Returns: none

Includes: none

Category: Function Calls — New Voice Tool - Swiss

❒ Description

Procedures allowes you to groupe your code into small units. Procedures also allowes you to reuse a part of your code at different moments without rewriting it.

❒ Example

```
/* the procedure reads lines from a given file */

str text1 text2
str filename

proc  readfile
{
  fopen filename "read"
  fread text1
  fread text2
  fclose
}

{
  !"please enter 1 for gold or 2 for silver
  getkey  1

  if key= "1"
    filename= "gold.txt"
    readfile
  else
    filename= "silver.txt"
    readfile
  fi
}
```

Name:	process <command>	
Inputs:	str command	DOS / OS2 - Command + Windows - Command
Returns:		none
Includes:		none
Category:	System Calls	New Voice Tool - Swiss

❒ Description

This function starts an external process. This can be a DOS - program, Windows or OS/2 command. The NVT-program will continue, while the process continues in parallel.

❒ Note

This uses a lot of CPU-time and memory. Whenever a process terminates within a few seconds, use the RUN command instead of PROCESS.

To start COPY and / or other System-Commands, create a *.CMD file.

```
rem    copybeep.cmd       copies a file and plays one beep ( ascii 8 )
copy   %1   %2
@echo

/* copybeep.nvt    use: [\nvt]nve.exe copybeep  */
{
 process  "copybeep.cmd  d.d  e.e"
}
```

❒ Example

```
/* this program copies the recorded file to another file */
str command  fname
{
 !"please speak after the beep"
 fname= "message"
 rec  fname  -cont
 command= "nvtcmd.exe copyfile "+ fname+ " "+ fname+".vox"
 !"do:> "+ command
 process  command
}
```

Name:	run <command>	
Inputs:	str command	DOS / OS2 - Command + Windows - Command
Returns:		none
Includes:		none
Category:	System Calls	New Voice Tool - Swiss

❒ Description

This function starts an external process. This can be a DOS-, Windows- or OS/2-.program. The nvt-program will wait until the process has been terminated.

❒ Note

This is a fast and easy way to communicate with external programs. It is much faster than the command -> PROCESS and recommended, if a lot of processes will be started.

To start COPY and / or other System-Commands, create a *.CMD file.

```
rem   copybeep.cmd      copies a file and plays one beep ( ascii 8 )
copy  %1  %2
@echo

/* copybeep.nvt   use: [\nvt]nve.exe copybeep  */
{
 run "copybeep.cmd  d.d  e.e"
}
```

❒ Example

```
/* this program copies the recorded file to another file */
str command  frec  fplay
{
 !"please speak after the beep"
 frec=  "message.rec"
 fplay= "message.vox"
 rec  frec  -cont
 command= "nvtcmd.exe copyfile "+ frec+ " "+ fplay
 !"do:> "+ command
 run  command
}
```

integration

reads pabx-integration data

Name:	integration	
Inputs:		none
Returns:	st> 0	internal phone number
	st= 0	no internal phone number
	st= -1	error reading data
Modifies:	str nvt_int_phone	internal phone number
	str nvt_ext_phone	external phone number
	int nvt_phone_type	type of phone call
	= 0	external direct call
	= 1	external diverted call
	= 2	internal direct call
Includes:	nvtcom.exe <com-port-process>	independent process
Category:	Integration - Commands	New Voice Tool - Swiss

❒ Description

This function reads the integration data receiving from the pabx. Usually, the pabx sends the information about the call through a serial interface. Set up the appropriate process to read from the com-port and to write to the NVT-Interface-Manager.

❒ Note

The function integration should not be started, before the data has been sent from the pabx, else old data will be read. Either set the number of rings before answering the call to 2 or 3 or play a standard vox-file before using integration to read the integratiion-data.

❒ Example

```
/* read the integration data and start the actual program    */
/* answers the call after 2 rings                            */

{
  sleep 5
  integration
  !"Start the program "+ nvt_int_phone+ ".nvt"
  do  nvt_int_phone
}
```

Category: Integration - Commands New Voice Tool - Swiss

❒ Description

To give an idea to the programmer, how the function integration works, a demonstration program has been written in the New Voice Tool -language.

❒ Example

```
int linenbr
str own_phone_number
{
  sleep 5
  dbuse  "nvtini"
  if dbok
     'go to the top first record of nvtini.dbf
     dbtop
     linenbr= nvtini_linenbr-1
     if linenbr> 0
        dbskip linenbr
        !"skiped "+ linenbr+ " with "+ st
     fi

     'T58 is the fieldname of nvtini.dbf, where the relevant data
     ' has been written to.
     dbget  "T58"  nvt_int_phone

     dbget  "T59"  nvt_ext_phone
     dbget  "T57"  nvt_phone_type
     dbget  "T60"  own_phone_number

     dbclose
  fi

  !"diverted call from the extention: "+ nvt_int_phone
  'save statistic data
  nvtini_s1=nvt_int_phone
  nvtini_i4=nvtini_linenbr

  !"start the program "+ nvt_int_phone+ ".nvt"
  do  nvt_int_phone
  if st< 0
    !"program  "+ nvt_int_phone+ " could not be found"
    play  "infotext.vox"
  fi
}
```

Name:	getdtr -comadr <adr>	
Inputs:	str adr	physical adress of comport
Returns:	0 | 1	the state of the dtr-signal
Includes:		none
Category:	Com-Port Commands	New Voice Tool - Swiss

❒ Description

This function gets the state of the dtr-signal (0 or 1).

❒ Example

```
/* gets the state of the comport and lounches an alarm on 0 */
cst WINDOWS= 1
{
  loop
     if WINDOWS then
       getdtr   -comadr  "COM1"
     else
       getdtr   -comadr  "2f8"
     fi
     if st= 0
       ofhook
       sleep 10
       dial  "0041 1 364 19 91"   -rings 12
       if st= 1
         play  "comalarm.vox"
       else
         setdtr  1  -comadr  "3e8"
       fi
     fi

          sleep 100

  lend
}
/* For alarm systems, check alarm-system-roules attentively  */

/* 25 pin: connect pin 4 to 5 to get dsr: on, open 4 to 5 or connect 7 to 5 to get dsr: off */
/*   9 pin: connect pin 7 to 8 to get dsr: on, open 7 to 8 or connect 5 to 8 to get dsr: off */
```

Name:	setdtr <state> -comadr <adr>	
Inputs:	int state	set dtr to 1 or 0
	str adr	physical adress of comport
Returns:	0	on no error
Includes:		none
Category:	Com-Port Commands	New Voice Tool - Swiss

❒ Description

This function sets the dtr-signal to 0 or 1

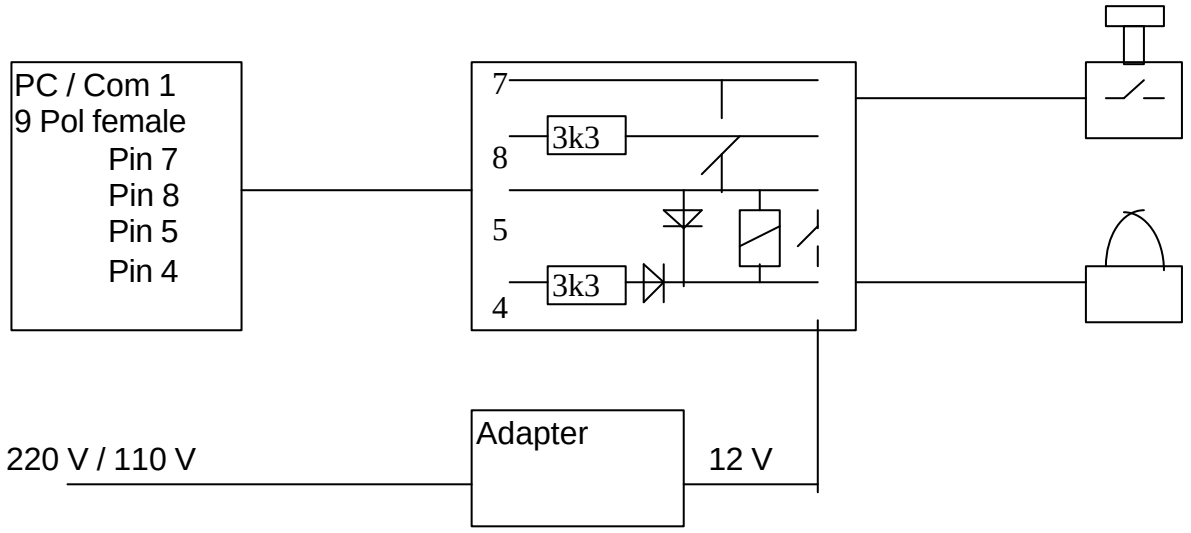
❒ Example

```
/* changes the state of the comport from 0 to 1 to 0 to 1... */
/* connect a modem for world most expensive blinking light.  */
cst WINDOWS= 1
{
  loop
    if WINDOWS then
      setdtr  1  -comadr  "COM1"
    else
      setdtr  1  -comadr  "2f8"
    fi

    !"set comport  2f8  to 1"
    Sleep 10

    if WINDOWS then
      setdtr  0  -comadr  "COM1"
    else
      setdtr  0  -comadr  "2f8"
    fi
    !"set comport  2f8  to 0"
         sleep 10

  lend
}
/* This program can also be started with   nve.exe  setdtr   */

/* 25 pin: if dtr on, a relais will be on between 20 to 7 ( 20: + 5 V - +9 V ) */
/*   9 pin: if dtr on, a relais will be on between   4 to 5 ( 20: + 5 V - +9 V ) */

/* insert a diode between 20 and the relais and 7, as on dtr: of, the voltage */
/“ is only inverted !  */
```

Category: Com-Port Commands New Voice Tool - Swiss

❒ Example

An alarm-contact will be connected to the comport 1 and a flashing light will be connected to the same comport. NVSPCTRL.EXE controls the comport. See next page !



Programs: NVSPCTRL.EXE controls the comport and opens the pipe „nvspctrl“
comp_on.nvt sets the flashing light to active
comp_off.nvt desactivates the flashing light
cts_on.bat is started if cts changes from 0 to 1
cts_off.bat is started if cts changes from 1 to 0

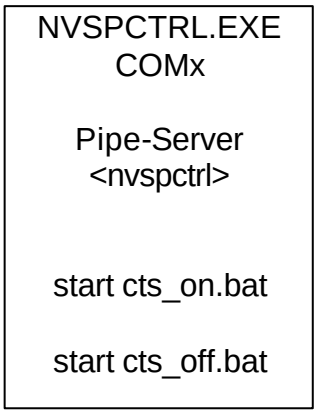
possible connection plan of zero-modem cable

	9 pin female		9 pin female		25 pin male
	7	<-	1	->	8
	3	<-	2	->	3
	2	<-	3	->	2
	6 + 8	<-	4	->	20
	5	<-	5	->	7
	4	<-	6	->	6
	1	<-	7	->	4
	4	<-	8	->	5
	9	<-	9	->	-

Category: Com-Port Commands New Voice Tool - Swiss

❒ Concept

The program nvspctrl.exe controls the selected comport, waits for changes on CTS to start batch-programs and waits for commands on the pipe <nvspctrl>.



//========== comp_on.nvt	//========== comp_off.nvt

str cmd_on
{
cmd_on= "Control*on*"
strxchg cmd_on "*" "nvt_line_feed"
!"on: "+ cmd_on
openpipe "nvspctrl"
!"open pipe: "+ st
putpipe cmd_on
!"put pipe on: "+ st
closepipe
 	
nvtini_s1= "****** beep ******“
strxchg nvtini_s1 "*" "nvt_beep_tone"
!nvtini_s1
}
	str cmd_off
	{
	cmd_off= "Control*off*"
	strxchg cmd_off "*" "nvt_line_feed"
	!"off: "+ cmd_off
	openpipe "nvspctrl"
	!"open pipe: "+ st
	putpipe cmd_off
	!"put pipe on: "+ st
	closepipe
	
	nvtini_s1= "****** beep ******“
	strxchg nvtini_s1 "*" "nvt_beep_tone"
	!nvtini_s1
	}

//========== cts_on.bat	//========== cts_off.bat
echo cts_on.bat \nvt\nve.exe comp_on rem pause	echo cts_off.bat \nvt\nve.exe comp_off rem pause

Name:	delmbx <mbx_number>	
Inputs:	int mbx_number	a number from 10 to 9999 using the dbase vm001.dbf
Includes:	nvtini_db1	dbase directory vm001.dbf
Category:	Voice Mail Commands	New Voice Tool - Swiss

❒ Description

This function deletes all stored messages of the given mailbox. Each message status will be reseted to 0 an the corresponding vox-file will be deleted. The system-database vm001.dbf contains about 10000 mailboxes. Mailboxes with the numbers 10 to 9999 are accessible.
Every mailbox can store an unlimitted number of messages. The physical limit of the total number of messages in the database vm001.dbf is 2000 (default), 4000 or 10000.
Every mailbox has its own password (range from 1000 to 9999) to protect the stored messages against unauthorized access.
Every mailbox has its status (range from 0 to 99) to keep an information about the state of the mailbox. i.e. 0: out of service. >0: active. 10: outdial 0 allowed. 20: outdial 00 allowed

❒ Example

```
/* Answering machine, deleting and moving messages ok */
{
  !"the messages stored in mbx 100 will be moved to 101 "/
  !"it is important to delete stored messages in 101 first"
  delmbx  101
  if  st# 0            // check result of delmbx
    !"the messages from mailbox: <101> could not be deleted"
    playmbx  111
  fi
  movembx  100  101
}
```

Name:	getmbx <mbx_number> [-option] <return_value>	
Inputs:	int mbx_number	a number from 10 to 9999 using the dbase vm001.dbf
Option:	-status (int) status	returns the mailbox-status
Returns:	int status	range: 0 < status < 100
Option:	-pwd (str) password	compairs password
Returns:	st= -1	password does not match
	st= >0	password does match
Option:	-msgnbr (int) msgnbr	returns the number of stored messages
Returns:	int msgnbr	range: 0<= nbrmsg < 10000
Includes:	nvtini_db1	dbase directory vm001.dbf
Category:	Voice Mail Commands	New Voice Tool - Swiss

❒ Description

This function gets information about the given mailbox. The system-database vm001.dbf contains about 10000 mailboxes. Mailboxes with the numbers 10 to 9999 are accessible.
Every mailbox can store an unlimitted number of messages. The physical limit of the total number of messages in the database vm001.dbf is 2000 (default), 4000 or 10000.
Every mailbox has its own password (range from 1000 to 9999) to protect the stored messages against unauthorized access.
Every mailbox has its status (range from 0 to 99) to keep an information about the state of the mailbox. i.e. 0: out of service. >0: active. 10: outdial 0 allowed. 20: outdial 00 allowed

❒ Example

```
/* Answering machine, playing the messages if password ok */
str password
{
  !"Please enter your password"
  play  "enterpwd.vox"  nvtini_lg1
  getkey  password  4 -time 20
  getmbx  111  -pwd  password
  if  st= 0            // check operator-password
    !"play the recorded messages from mailbox: <111>"
    playmbx  111
  fi
}
```

Demonstration

Category: Voice Mail Commands New Voice Tool - Swiss

❒ Example

```
/* Playing the number of stored messages in a mailbox */
int msgnbr
int mbxnbr
{
  !"Please enter your mailbox number"
  play  "entermbx.vox"  nvtini_lg1
  getkey  keys  4 -time 20
  if  Timeout
    !"Please enter your mailbox number"
    play  "entermbx.vox"  nvtini_lg1
    getkey  keys  4 -time 20
    if  Timeout
      !"Thank you and good bye"
      play  "thankyou.vox"  nvtini_lg1
      hangup
    fi
  fi
  mbxnbr= keys
  getmbx  111  -msgnbr  msgnbr
  if  msgnbr= 0
    !"there are no messages recorded"
        play  "no_msg.vox"  nvtini_lg1
    return
  fi
  if  msgnbr> 1 and msgnbr < 10
    !"there are "+ mbxnbr+ " messages stored"
    play  "nbr_msg.vox"  nvtini_lg1
    msgnbr= msgnbr+ 48
    play  "alphabet.vox"  nvtini_lg1  -part msgnbr
    msgnbr= msgnbr- 48
    play  "nbr_msg2.vox"  nvtini_lg1
  fi
  if  msgnbr> 9
    !"there are many messages stored"
    play  "vielemsg.vox"  nvtini_lg1
  fi

  playmbx  mbx_nr
}
```

Name:	getmbx <mbx_number> [-option] <return_value>	
Inputs:	int mbx_number	a number from 10 to 9999 using the dbase vm001.dbf
Option:	-ogm str ogm	gets the default ogm
Option:	-ogm1 str ogm	gets ogm 1
Option:	-ogm2 str ogm	gets ogm 2
Return:	st= 4, 1, 2	for actual ogm: default, 1, 2
	st= -1	ogm 1 or 2 not assigned
Includes:	nvtini_db1	dbase directory vm001.dbf
Category:	Voice Mail Commands	New Voice Tool - Swiss

❒ Description -cont

This function gets the actual out going message (ogm) with -ogm, ogm 1 with -ogm1 and ogm 2 with -ogm2 - from the given mailbox.
Every call of getmbx -ogm will return the actual ogm-name.

❒ Note:

o Every ogm is treated as a message and uses one of the 2000 (or 4000 or 10000) message
records from vm001.dbf.
o A once declaired ogm will occupie its record in vm001.dbf until delmbx is started.
o The ogm-type 0 is stored in vm001.dbf as type 4.
o The option -ogm will return either standard.vox or standa_s.vox if nvtini_din= 1 and if the actual ogm is neither 1 nor 2. standard.vox would be a text with company-name announcement and standa_s.vox without. This is not important, as in case of ogm= 0 every voicefile can be played, not only the returned name in ogm.

❒ Example

```
/* Answering machine, playing the actual out going message */
str ogm filename
{
  getmbx 111 -ogm ogm
  !"Play actual ogm of mailbox 111 to caller"
  play ogm nvtini_dbv1
  if Notfound
    play "standard.vox" nvtini_lg1
  fi
  recmbx 111 -norec filename
  rec filename nvtini_dbv1
}
```

Name:	getmbx <mbx_number> [-option] <return_value>	
Inputs:	int mbx_number	a number from 10 to 9999 using the dbase vm001.dbf
Option:	-date str date	gets the creation date
Option:	-time str time	gets the creation time
Option:	-tel str phone	gets the phone - number
Option:	-pager str pager	gets the pager - number
Option:	-mbxf int targetmailbox	gets the target mailbox
Option:	-mbxr int infomailbox	gets the information mailbox
Includes:	nvtini_db1	dbase directory vm001.dbf
Category:	Voice Mail Commands	New Voice Tool - Swiss

❒ Description -cont

This function gets the date and time - previous set with setmbx - from the given mailbox. The phone and pager numbers can be used to call the owner of the mailbox on reception of a message.

❒ Note:

o The format of the date is: yy.ddd
  optionally: dd.mm
o The format of the time is: hh.mm
o The phone and pager numbers are strings with up to 20 characters

❒ Example

```
/* Answering machine, dialling the pagernumber of the mbx */
/* uses the program calloper.nvt to call the operator     */
str filename  pager  command
{
  play  "speaknow.vox"  nvtini_dbv1
  recmbx  111  -norec  filename
  rec  filename  nvtini_dbv1
  getmbx  111  -pager  pager
  command= "nvtque.exe batchpost calloper 1 -s1"+ pager
  !command
  run command
}
```

Name:	getmbx <mbx> [-field str_fieldname <return_value>]	
Inputs:	int mbx_number	a number from 10 to 9999 using the dbase vm001.dbf
Fieldname:	"ogmx" int type_ogm	ogm type set to 1, 2.. 9 or 0
Fieldname:	"vacation" int p_ogm1	mailbox / ogm during vacation
Fieldname:	"close" int p_ogm2	mailbox / ogm while closed
Fieldname:	"open" int p_ogm3	mailbox / ogm in working hours
Fieldname:	"schedule" int type_answer	status of schedule configuration
Fieldname:	"nbrmsg" int n_msg	the number of stored messages
Fieldname:	"service21" str text_opt1	divert the call to this number
Includes:	nvtini_db1	dbase directory vm001.dbf
Category:	Voice Mail Commands	New Voice Tool - Swiss

❒ Description -cont

This function plays the selected outgoing-message of the given mailbox. If the outgoing-message can not be found, a standard message of the mailbox will be played..

❒ Note:

Please be carefully when choosing the type of the result-variable.

❒ Example

```
/* Answering machine, playing the given out-going-message */
int mbx
int i_ogmx
str filename
{
  mbx= 111
  getmbx mbx -field "ogmx" i_ogmx
  filename= nvtini_aab1+ mbx+ "_"+ i_ogmx+ ".vox"
  play filename nvtini_dbv1
  if Notfound
    filename= nvtini_aab1+ mbx+ "_"+ ".vox"
    play filename nvtini_dbv1
  fi
  if Notfound
    play "standard.vox" nvtini_dbv1
  fi
  recmbx mbx -norec filename
  rec filename nvtini_dbv1
}
```

Name: getmsg <msg_number> [-option] <return_value>
Inputs: int msg_number a number from 10 to 11999 using the dbase vm001.dbf
str filename_with_msg_number one voxfile of the mailbox

Option: -date str date gets the creation date
Option: -time str time gets the creation time
Option: -tel str phone gets the phone - number
Option: -pager str pager gets the pager - number
Option: -state str state gets the state of the msg
Option: -pwd str password gets the pwd of the msg

Includes: nvtini_db1 dbase directory vm001.dbf

Category: Voice Mail Commands New Voice Tool - Swiss

❒ Description -cont

This function gets addidtional information about a message stored in a mailbox. So the date and time - previous set with setmbx - will be returned from the given mailbox.
The phone and pager numbers can be used to call the owner of the mailbox on reception of a message.
The password can be used to protect private messages to be heard by unauthorized persons.

❒ Note:

o The format of the date is: yy.ddd
optionally: dd.mm
o The format of the time is: hh.mm
o The phone and pager numbers are strings with up to 20 characters

❒ Example

```
/* Answering machine, playing the callers identification   */
/* The message-nr 10037 will be extracted from the filename */
str filename  callers_phone
{
  filename= "vm10037.vox"
  getmsg  filename  -tel  callers_phone
  !"The callers phone-number is: "
  play  "speakcli.vox"  nvtini_dbv1
  speak  -string  callers_phone
}
```

Name:	movembx <mbx_number> <mbx_target_number>	
Inputs:	int mbx_number	a number from 10 to 9999 using the dbase vm001.dbf int mbx_target_number a number from 10 to 9999 using the dbase vm001.dbf
Returns:	int status	0: ok, -1: target mailbox invalid -2: target mailbox not empty
Includes:	nvtini_db1	dbase directory vm001.dbf
Category:	Voice Mail Commands	New Voice Tool - Swiss

❒ Description

This function moves all stored messages of the given mailbox to the given Target-mailbox. The first (latest) message will be moved to the target mailbox. The target mailbox must be empty. The corresponding vox-files are unchanged The system-database vm001.dbf contains about 10000 mailboxes. Mailboxes with the numbers 10 to 9999 are accessible.
Every mailbox can store an unlimitted number of messages. The physical limit of the total number of messages in the database vm001.dbf is 2000 (default), 4000 or 10000.
Every mailbox has its own password (range from 1000 to 9999) to protect the stored messages against unauthorized access.
Every mailbox has its status (range from 0 to 99) to keep an information about the state of the mailbox. i.e. 0: out of service. >0: active. 10: outdial 0 allowed. 20: outdial 00 allowed

❒ Example

```
/* Answering machine, deleting and moving messages ok */
{
  !"the messages stored in mbx 100 will be moved to 101 "/
  !"it is important to delete stored messages in 101 first"
  delmbx  101
  if  st# 0            // check result of delmbx
    !"the messages from mailbox: <101> could not be deleted"
    play  “err_move.vox“
  fi
  movembx  100  101
}
```

Name:	playmbx <mbx_number> [-options]	
Inputs:	int mbx_number	a number from 10 to 9999 using the dbase vm001.dbf
Options:	-first	get first message
Returns:	str filename	name of the first message
Returns:	st= 0	no messages are stored
	st= >0	this is not the last message
Options:	-next	get the next message
Returns:	str filename	name of the next message
Returns:	st= 0	no more messages
	st= >0	this is not the last message
Options:	-delmsg	delete the actual message
Returns:	str filename	name of the next message
Returns:	st= 0	no more messages
	st= >0	this is not the last message
Includes:	nvtini_db1	dbase directory vm001.dbf
Category:	Voice Mail Commands	New Voice Tool - Swiss

❒ Description

This function returns the demanded message of the given mailbox. The system-database vm001.dbf contains about 10000 mailboxes. Mailboxes with the numbers 10 to 9999 are accessible. Every Mailbox can store an unlimitted number of messages. The physical limit of the total number of messages in the database vm001.dbf is 2000 (default), 4000 or 10000.
The first command must be lounched with the option -first to get the newest message, then all messages will be returned with the option -next.
The option -delmsg will delete the message and return the next message. -delmsg will not delete the message on the harddisk. This should be done with run "nvtcmd deletefile filename".

❒ Example

```
/* Answering machine, playing the first message */
str filename
{
  playmbx 111 -first filename
  play filename nvtini_dbv1
}
```

Demonstration

Category: Voice Mail Commands New Voice Tool - Swiss

❒ Example

```
/* Playing all messages stored in a mailbox */
str  filename
cst  mbxnbr= 111
int  msgnbr
str  entry
str  command              //string to delete the message
{
  playmbx  mbxnbr  -first  filename
  msgnbr= st
  while  msgnbr> 0
    play  filename  nvtini_dbv1
    entry= ""
    while  eingabe # "3" and eingabe # "4"
      !"press 3 for next, 4 to delete, # to exit"
      play  "play_cmd.vox"  nvtini_lg1
      getkey  entry  1  -time 40
      if  entry= "2"
        play  filename  nvtini_dbv1
      fi
      if  entry= "3"
        playmbx  mbxnbr  -next  filename
        msgnbr= st
      fi
      if  entry= "4"
        command= "nvtcmd deletefile "+ nvtini_dbv1+filename
        !"run message delete process: "+ command
        run  command
        playmbx  mbxnbr  -delmsg  filename
        msgnbr= st
      fi
      if  entry= "#"
        play  "thankyou.vox"  nvtini_lg1
        hangup
      fi
    wend
  wend
  play  "lastmsg.vox"  nvtini_lg1
  play  "thankyou.vox"  nvtini_lg1
}
```

recmbx

record message to mailbox

Name:	recmbx <mbx_number> [-options]	
Inputs:	int mbx_number	a number from 10 to 9999 using the dbase vm001.dbf
Options:	-norec <filename>	do not record immediately
Returns:	str filename	filename to record message
Returns:	st= -1	if vm001.dbf not found
	st= >0	normal termination
	st= 10001.. 11999	contains the message number
Includes:	nvtini_db1	dbase directory vm001.dbf
Category:	Voice Mail Commands	New Voice Tool - Swiss

❒ Description

This function allocates a messages into the given mailbox: The message will be recorded later with the standard function **rec**. The recorded messages will be played with the function **playmbx**. The system-database vm001.dbf contains about 10000 mailboxes. Mailboxes with the numbers 10 to 9999 are accessible. Every Mailbox can store an unlimitted number of messages. The physical limit of the total number of messages in the database vm001.dbf is 2000 (default), 4000 or 10000.

❒ Example

```
/* Answering machine, playing the messages if password ok */
str filename
{
  !"Wellcome to New Voice. Please leave your msg after bip"
  play  "wellcome.vox"  nvtini_lg1
  recmbx  111  -norec  filename
  if st > 0
    !"record message "+ nvtini_dbv1+ filename+ "into mbx 111"
    rec  filename  nvtini_dbv1
  else
    !"no message can be recorded"
    play  "sysfull.vox"  nvtini_lg1
  fi
}
```

Name:		recmbx <mbx_number> [-options]
Inputs:	int mbx_number	a number from 10 to 9999 using the dbase vm001.dbf
Option:	-ogm str ogm	record actual ogm
Option:	-ogm1 str ogm	record ogm 1
Option:	-ogm2 str ogm	record ogm 2
Returns:	str ogm	ogm-name to record
Returns:	st= -1 st= >0	if vm001.dbf not found normal termination
Includes:	nvtini_db1	dbase directory vm001.dbf
Category:	Voice Mail Commands	New Voice Tool - Swiss

❒ Description

This function allocates the given out going message (ogm) into the given mailbox: The ogm will be recorded later with the standard function **rec**. The recorded ogm will be refound with the function **getmbx -ogm**.

❒ Example

```
/* Answering machine, recording the first ogm */
str ogm
{
  recmbx 111 -ogm1 ogm
  !"Plase speak after the beep"
  play "speaknow.vox" nvtini_lg1
  rec ogm nvtini_dbv1
}
```

Name:	setmbx <mbx_number> [-option] <data_value>	
Inputs:	int mbx_number	a number from 10 to 9999 using the dbase vm001.dbf
Option:	-status int status	sets the mailbox-status range: 0 < status < 100
Option:	-pwd str password	sets the mailbox password 999 < password < 10000
Includes:	nvtini_db1	dbase directory vm001.dbf
Category:	Voice Mail Commands	New Voice Tool - Swiss

❒ Description

This function sets data in the given mailbox. The system-database vm001.dbf contains about 10000 mailboxes. Mailboxes with the numbers 10 to 9999 are accessible.
Every mailbox can store an unlimitted number of messages. The physical limit of the total number of messages in the database vm001.dbf is 2000 (default), 4000 or 10000.
Every mailbox has its own password (range from 1000 to 9999) to protect the stored messages against unauthorized access.
Every mailbox has its status (range from 0 to 99) to keep an information about the state of the mailbox. i.e. 0: out of service. >0: active. 10: outdial 0 allowed. 20: outdial 00 allowed

❒ Example

```
/* Answering machine, setting the password to a new value */
str password
{
  !"Please enter your password"
  play  "enterpwd.vox"  nvtini_lg1
  getkey  password  4 -time 20
  setmbx 111 -pwd password
  !"the password has been changed"
  play  "changpwd.vox"  nvtini_lg1
}
```

Name:	setmbx <mbx_number> [-option] <data_value>	
Inputs:	int mbx_number	a number from 10 to 9999 using the dbase vm001.dbf
Option:	-ogm str ogm	sets ogm-type to standard
Option:	-ogm1 str ogm	sets ogm-type to ogm 1
Option:	-ogm2 str ogm	sets ogm-type to ogm 2
Includes:	nvtini_db1	dbase directory vm001.dbf
Category:	Voice Mail Commands	New Voice Tool - Swiss

❒ Description

This function sets the actual ogm to 0 for standard out going message (ogm), to 1 for ogm 1 and to 2 for ogm - in the given mailbox.
Every call of setmbx -ogm will return the actual ogm-name.
Please note:
o Every ogm is treated as a message and uses one of the 2000 (or 4000 or 10000) message records from vm001.dbf.
o A once declaired ogm will occupie its record in vm001.dbf until delmbx is started.
o The ogm-type 0 will be stored in vm001.dbf as type 4.
o The option -ogm will return either standard.vox or standa_s.vox if nvtini_din= 1. standard.vox would be a text with company-name announcement and standa_s.vox without. This is not important, as in case of ogm= 0 every voicefile can be played, not only the returned name in ogm.

❒ Example

```
/* Answering machine, setting the ogm to the standard ogm */
str ogm
{
  !"Please press 1 for ogm1, 2 for ogm2 3 for standard"
  play "enterogm.vox" nvtini_lg1
  getkey key 1 -time 40
  if key = "1" setmbx 111 -ogm1 ogm fi
  if key = "2" setmbx 111 -ogm2 ogm fi
  if key = "3" setmbx 111 -ogm ogm fi
  !"the out going message has been changed"
  play "changogm.vox" nvtini_lg1
}
```

Name:	setmbx <mbx_number> [-option] <return_value>	
Inputs:	int mbx_number	a number from 10 to 9999 using the dbase vm001.dbf
Option:	-date str date	sets the creation date
Option:	-time str time	sets the creation time
Option:	-tel str phone	sets the phone - number
Option:	-pager str pager	sets the pager - number
Option:	-mbxf int targetmailbox	sets the target mailbox
Option:	-mbxr int infomailbox	sets the information mailbox
Includes:	nvtini_db1	dbase directory vm001.dbf
Category:	Voice Mail Commands	New Voice Tool - Swiss

❒ Description -cont

This function sets the date and time of the given mailbox.
The phone and pager numbers can be used to call the owner of the mailbox on reception of a message.

❒ Note:

o The format of the date is: yy.ddd
  optionally: dd.mm
o The format of the time is: hh.mm
o The phone and pager numbers are strings with up to 20 characters

❒ Example

```
/* Program to store the pager number of the mbx */
str pager
{
  !"please enter your pager number now"
  play "enterpag.vox" nvtini_dbv1

  getkey pager 20 -time 30
  setmbx 111 -pager pager
}
```

Name:	setmbx <mbx> [-field str_fieldname <return_value>]	
Inputs:	int mbx_number	a number from 10 to 9999 using the dbase vm001.dbf
Fieldname:	"ogmx" int type_ogm	ogm type set to 1, 2.. 9 or 0
Fieldname:	"vacation" int p_ogm1	mailbox / ogm during vacation
Fieldname:	"close" int p_ogm2	mailbox / ogm while closed
Fieldname:	"open" int p_ogm3	mailbox / ogm in working hours
Fieldname:	"schedule" int type_answer	status of schedule configuration
Fieldname:	"nbrmsg" int n_msg	the number of stored messages
Fieldname:	"service21" str text_opt1	divert the call to this number
Includes:	nvtini_db1	dbase directory vm001.dbf
Category:	Voice Mail Commands	New Voice Tool - Swiss

❒ Description -cont

This function defines the active outgoing message of the given mailbox.

❒ Note:

Please be carefully when choosing the type of the result-variable.

❒ Example

```
/* Answering machine, setting the active out-going-message */
int mbx
int i_ogmx
str filename
{
  mbx= 111
  setmbx mbx -field "ogmx" 0
  !"to activate your ogm1, press 1, for ogm2 press 2"
  getkey key 1 -time 30
  if key= "1"
    setmbx mbx -field "ogmx" 1
  fi
  if key= "2"
    setmbx mbx -field "ogmx" 2
  fi
}
```

Name: setmsg <msg_number> [-option] <return_value>
Inputs: int msg_number a number from 10 to 11999 using the dbase vm001.dbf
str filename_with_msg_number one voxfile of the mailbox

Option: -date str date gets the creation date
Option: -time str time gets the creation time
Option: -tel str phone gets the phone - number
Option: -pager str pager gets the pager - number
Option: -state str state gets the state of the msg
Option: -pwd str password gets the pwd of the msg

Includes: nvtini_db1 dbase directory vm001.dbf

Category: Voice Mail Commands New Voice Tool - Swiss

❒ Description -cont

This function sets addidtional information about a message stored in a mailbox. So the date and time stamp can be modified and stored.
The phone-number of the caller can be stored with the recorded message. When ever the caller does not leave a message, you can store at least it's number (if known by the pabx).
The password can be used to protect private messages to be heard by unauthorized persons.

❒ Note:

o The format of the date is: yy.ddd
optionally: dd.mm
o The format of the time is: hh:mm
o The phone and pager numbers are strings with up to 20 characters

❒ Example

```
/* Answering machine, playing the callers identification */
/* The pabx sends the cli immediately after going offhook */
str filename callers_phone
{
  getkey callers_phone 20 -time 10
  recmbx 111 -norec filename
  setmsg filename -tel callers_phone
  play "speaknow.vox"
  rec filename
}
```

Name:	playmbx <mbx_number> [-options]	
Inputs:	int mbx_number	a number from 10 to 9999 using the dbase vm001.dbf
Options:	-cont	continue after hangup
Returns:	st= -1	if vm001.dbf not found
	st= >0	normal termination
Includes:	nvtini_lg1	prerecorded vox-prompts
	nvtini_db1	dbase directory vm001.dbf
	nvtini_dbv1	directory of rec. messages
	nvtini_aab1	default: "vm" for naming messages: vm10001.vox...
Category:	Voice Mail Commands	New Voice Tool - Swiss

❒ Description

This function plays the messages stored in the given mailbox, which has been previously recorded with **recmbx**. The system-database vm001.dbf contains about 10000 mailboxes. Mailboxes with the numbers 10 to 9999 are accessible. Every Mailbox can store an unlimitted number of messages. The physical limit of the total number of messages in the database vm001.dbf is 2000 (default), 4000 or 10000.
Depending the presense of the prerecorded voice-prompts, different functions are supported. Please read the annex with the voice prompts carefully.

❒ Example

```
/* Answering machine, playing the messages if password ok */
{
  !"Wellcome to New Voice. Please leave your msg after pip"
  play  "wellcome.vox"  nvtini_lg1
  getkey  key  4 -time 20
  if  key # "1234"            // check operator-password
    !"record the message to mailbox: <111>"
    recmbx  111
  else
    !"play the recorded messages from mailbox: <111>"
    playmbx  111
  fi
}
```

-voice prompts
-functions

Category: Voice Mail Commands New Voice Tool - Swiss

Please note the different voice-prompts with text-examples for the different options:

Basic prompts:

PLAY_CMD.VOX "*To repeat the message, please press Two,*
to play the next message, press Three,
to delete this message, press Four,
(optional) *to copy the message to another mail-box, press Five,*
(optional) *to get the record date and time, press Six,*
(optional) *to get the phone number of the caller, press Seven,*
to exit the function, press the pound key.
Please make your selection now."

This menu-text will be played after every message and after every option (point 5, 6, 7).

NO_MSG.VOX "*Sorry, there is no message for you*"
If no message has been stored, the function will return immediately after this text.

LASTMSG.VOX "*This was the last message*"
Until you do not hear this text, you have not reached the last message in the mailbox.

Confirm the deletion of a message

MSGDELOK.VOX "*The msessage was deleted*"
MSGDELKO.VOX "*The msessage was not deleted*"
This text will be played after deletion or non-deletion of the actual message.

System informations

RETURN.VOX "*The mailbox number is invalid*"
This text will be played if the mailboxnumber is too low.

ERRMBX.VOX "*There is no corresponding mailbox*"
This text will be played if the mailboxnumber is too big, or no database has been found.

SERV_KO.VOX "*The mailbox number is invalid*"
This text will be played on an invalid entry (i.e. key 8).

-voice prompts
-functions

Category: Voice Mail Commands New Voice Tool - Swiss

Please note the different voice-prompts with text-examples for the different options:

Announce the number of stored messages (optional)

NO_MSG.VOX	"*Sorry, there is no message for you*"
EINEMSG.VOX	"*There is one message for you*"
NBR_MSG.VOX	"*There are*"
ALPHABET.VOX	part 48: "*zero*", part 49: "*one*", part 50: "*two*", ... part 57: "*nine*"
NBR_MSG2.VOX	"*messages for you*"
VIELEMSG2.VOX	"*there are many messages for you*"

Speak the recording date and time of the message (optional)

NO_DATE.VOX	"*The date was not recorded*"
RECDATE.VOX	"*This message has been recorded on the ...*"
DATE_DAY.VOX	part 1: "*First*", part 2: "*Second*", .. part 31: "*Thirty first*"
DATE_MT.VOX	part 1: "*January*", part 2: "*February*",... part 12: "*December*"
TIME_HR.VOX	part 1: "*at Zero..*", part 2: "*at One..*", .. part 24: "*at Twentythree*"
TIME_MIN.VOX	part 1: "*Zero*", part 6: "*Five*", part 11: "*Ten*", ... part 56: "*Fiftyfive*"

Copy a stored message to another mailbox in the same database (optional)

MBX_NBR.VOX	"*Enter the number of the mailbox*"
TB_OFF.VOX	"*This answering machine is out of service and the message*" "*can not be copied*"
COPY_OK.VOX	"*The message is copied*"
COPY_KO.VOX	"*The message is not copied*"
SPEAKNOW.VOX	"*Please speak after the beep and end your message by*" "*pressing the pound key*"

Speak the callers phone-number, if stored with recmbx (optional)

EXTNR.VOX	"*The external caller's number is:*"
INTNR.VOX	"*The internal caller's number is:*"
TRANR.VOX	"*The diverted caller's number is:*"
ALPHABET.VOX	part 48: "*zero*", part 49: "*one*", part 50: "*two*", ... part 57: "*nine*"
NOEXTNR.VOX	"*The external caller's number is not known*"
NOINTNR.VOX	"*The internal caller's number is not known*"
NOTRANR.VOX	"*The diverted caller's number is not known*"

Name:	recmbx <mbx_number> [-options]	
Inputs:	int mbx_number	a number from 10 to 9999 using the dbase vm001.dbf
Options:	-cont	continue after hangup
Returns:	st= -1 st= >0	if vm001.dbf not found normal termination
Includes:	nvtini_lg1 nvtini_db1 nvtini_dbv1 nvtini_aab1	prerecorded vox-prompts dbase directory vm001.dbf directory of rec. messages default: "vm" for naming messages: vm10001.vox...
Category:	Voice Mail Commands	New Voice Tool - Swiss

❒ Description

This function records a messages into the given mailbox: This message will be played with the function **playmbx**. The system-database vm001.dbf contains about 10000 mailboxes. Mailboxes with the numbers 10 to 9999 are accessible. Every Mailbox can store an unlimitted number of messages. The physical limit of the total number of messages in the database vm001.dbf is 2000 (default), 4000 or 10000.
Depending the presense of the prerecorded voice-prompts, different functions are supported. Please read the annex with the voice prompts carefully.

❒ Example

```
/* Answering machine, playing the messages if password ok */
{
  !"Wellcome to New Voice. Please leave your msg after pip"
  play  "wellcome.vox"  nvtini_lg1
  getkey  key  4 -time 20
  if  key # "1234"            // check operator-password
    !"record the message to mailbox: <111>"
    recmbx  111
  else
    !"play the recorded messages from mailbox: <111>"
    playmbx  111
  fi
}
```

-voice prompts
-functions

Category: Voice Mail Commands New Voice Tool - Swiss

Please note the different voice-prompts with text-examples for the different options:

Basic prompts:

SPEAKNOW.VOX "*Please speak after the beep and end your message by*"
"*pressing the pound key*"

Menu to rerecord the message:

MSGCMD.VOX "*If you with to hear your message, press Two*"
"*To confirm your message press Three*"
"*To rerecord your message, press One*"

STORED.VOX "*Your message has been recorded. Thank you*"

System informations

RETURN.VOX "*The mailbox number is invalid*"
This text will be played if the mailboxnumber is too low.

ERRMBX.VOX "*There is no corresponding mailbox*"
This text will be played if the mailboxnumber is too big, or no database has been found.

SYSFULL.VOX "*Unfortunately no messages can be recorded at the moment*"
This text will be played if the number of recorded messages exceeds the maximum number of messages preallocated in the database vm001.dbf.

odbcappend

appends a record

Name:	odbcappend	
Inputs:		none
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:		Data Source Admininstrator
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function appends a new record to the opened database.

❒ Example

/* appends a new record at the end of "Test database" "Personal"*/

```
{
  odbcuse  "Test database" "Personal"
  if  dbok
    odbcbot
    odbcappend
    if  dbok
      odbcput  "Company"  "New Voice Zurich"
    else
      !"no new record can not be appended"
    fi
    odbcclose
  else
    !"the database can not be opened"
  fi
}
```

odbcbottom

goes to the last record

Name:	odbcbot[tom]	
Inputs:		none
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:		Data Source Admininstrator
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function goes to the bottom of the opened database.

❒ Example

/* appends a new record at the end of "Test database" "Personal"*/

```
{
  odbcuse  "Test database" "Personal"
  if  dbok
    odbcbot
    odbcappend
    if  dbok
      odbcput  "Company"  "New Voice Zurich"
    else
      !"no new record can not be appended"
    fi
    odbcclose
  else
    !"the database can not be opened"
  fi
}
```

odbcclose

close the opened database

Name: dbcclose
Inputs: none

Returns: dbok= 1 if successfull
dbok= 0 if failure

Includes: Data Source Admininstrator

Category: Database - Commands New Voice Tool - Swiss

❒ Description

This function closes the database opened previosly with dbuse.

❒ Example

/* appends a new record at the end of "Test database" "Personal"*/

```
{
  odbcuse  "Test database" "Personal"
  if  dbok
    odbcbot
    odbcappend
    if  dbok
      odbcput  "Company"  "New Voice Zurich"
    else
      !"no new record can not be appended"
    fi
    odbcclose
  else
    !"the database can not be opened"
  fi
}
```

odbcdelete

delete the active record

Name: odbcdel[ete]
Inputs: none

Returns: dbok= 1 if successfull
obok= 0 if failure

Includes: Data Source Admininstrator

Category: Database - Commands New Voice Tool - Swiss

❒ Description

This function deletes the active record of the opened database.

❒ Example

/* deletes the first record of vm001.dbf */

/* deletes the toplevel record of "Test database" "Personal"*/

```
{
  odbcuse  "Test database" "Personal"
  if  dbok
    odbctop
    odbcdel
    odbcclose
  else
    !"the database can not be opened"
  fi
}
```

odbcfind

find a value in a field

Name:	odbcfind <fieldname> <str_var | int_var> -next	
Inputs:	fieldname	fieldname in the database
	str_var	variable keeping the string
	int_var	variable keeping the integer
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Option:	-next	finds next entry
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function finds the given value in the selected field of the opened database.

❒ Example

```
/* finds the given expression */

str  name          /* string to keep the name */

{
 odbcuse  "Test database" "Personal"
 if  dbok
  name= "New Voice Zurich"
  odbcfind  "Name"  name
  if  dbok
   !"the name <"+ name+ "> found"
   odbcfind  "Name"  name  -next
   if  dbok
    !"the name <"+ name+ "> found a second time"
   else
    !"the name <"+ name+ "> not found a second time"
   fi
  else
   !"the name <"+ name+ "> not found"
  fi
  odbcclose
 else
  !"the database can not be opened"
 fi
}
```

odbcget

get the value of a field

Name:	odbcget <fieldname> [str_var | int_var]	
Inputs:	fieldname	fieldname in the database
	str_var	variable to store the string
	int_var	variable to store the integer
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function gets the value of a field of the opened database. The record is the active record
set with odbctop, odbcbot, odbcseek, odbcgo or odbcskip.

❒ Example

```
/* gets the name of the top level record */

str  name          /* string to keep the name */

{
  odbcuse  "Test database" "Personal"
  if  dbok
    odbctop
    odbcget  "Name"  name
    if  dbok
      !"the name is <"+ name+ ">"
    else
      !"error reading the name"
    fi
    odbcclose
  else
   !"the database can not be opened"
  fi
}
```

odbcput

put a value into a field

Name:	odbcput <fieldname> [str_var | int_var]	
Inputs:	fieldname	fieldname in the database
	str_var	variable keeping the string
	int_var	variable keeping the integer
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function puts a value into a field of the opened database. The record is the active record
set with odbctop, odbcbot, odbcseek, odbcgo or odbcskip.

❒ Example

```
/* sets the name of the top level record */

int  name          /* string to keep the name */

{
  odbcuse  "Test database" "Personal"
  if  dbok
    odbctop
    name= "New Voice Zurich"
    odbcput  "Name"  name
    if  dbok
      !"the name is set to <"+ name+ ">"
    else
      !"error writing the name: "+ name
    fi
    odbcclose
  else
   !"the database can not be opened"
  fi
}
```

Name:	odbcrecno <record_number>	
Inputs:	int record_number	variable to return the value
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:		Data Source Admininstrator
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function gets the record number of the actual record of the database.

❒ Example

```
/* gets the record number of the last record */

int  record_number      /* integer to keep the record nbr */

{
 odbcuse  "Test database" "Personal"
 if  dbok
   odbcbot
   if  dbok
     odbcrecno  record_number
     !"the record number is <"+ record_number+ ">"
   else
     !"error going to the bottom of the database"
   fi
   odbcclose
 else
  !"the database can not be opened"
 fi
}
```

❒ Limits - the record number is the number given during creation time und does not allways correspond with the number of the record in the database.

Name:	odbcsel < table>	
Inputs:	str table	name of table to select
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:	ODBC32 Server	odbc32-environment
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function selects the given table of the previous opened datasource

❒ Example

```
/* sets the name of the top level record */

int  name          /* string to keep the name */
{
  odbcuse  "Test database"
  if  dbok= 0
    !"the database can not be opened"
    hangup
  fi
  odbcsel  "personal"
  if  dbok
    odbctop
    name= "New Voice Zurich"
    odbcput  "Name"  name
    if  dbok
      !"the name is set to <"+ name+ ">"
    else
      !"error writing the name: "+ name
    fi
    odbcclose
  else
    !"the database table can not be selected"
  fi
}
```

❒ Limits - only one database can be opened at the same time
- do not forget to declare the name of the datasource in the odbc32-manager.
- these functions are not supported under OS/2

Name:	odbcskip [number]	
Inputs:	number	skip number of records
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Default:	1	
Includes:		Data Source Admininstrator
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function skips a given number of records. If no number is given, the function will skip to the next record.

❒ Example

```
/* skips through the database                    */
str  name          /* integer to keep the status */
{
 odbcuse  "Test database" "Personal"
 if  dbok
  odbctop
  odbcget  "Name"  name
  while  dbok  and  name # "New Voice Zurich"
   odbcskip  1
   if dbok
    odbcget  "Name"  name
   else
    !"end of database reached, name not found"
   fi
  wend
  odbcclose
  if name= "New Voice Zurich"
   !"entry found"
  else
   !"entry not found"
  fi
 else
  !"the database can not be opened"
 fi
}
```

odbctop

goes to the top record

Name:	odbctop	
Inputs:		none
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:		Data Source Admininstrator
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function goes to the top record of the database.

❒ Example

```
/* sets the name of the top level record */

int  name          /* string to keep the name */

{
  odbcuse  "Test database" "Personal"
  if  dbok
    odbctop
    name= "New Voice Zurich"
    odbcput  "Name"  name
    if  dbok
      !"the name is set to <"+ name+ ">"
    else
      !"error writing the name: "+ name
    fi
    odbcclose
  else
   !"the database can not be opened"
  fi
}
```

Name:	odbcuse <datasource> [table]	
Inputs:	str datasource	name of datasource
	str table	name of table to open
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:	ODBC32 Server	odbc32-environment
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function opens the given data source.

❒ Example

```
/* sets the name of the top level record */
int  name          /* string to keep the name */
{
 odbcuse  "Test database" "Personal"
 if  dbok
   odbctop
   name= "New Voice Zurich"
   odbcput  "Name"  name
   if  dbok
     !"the name is set to <"+ name+ ">"
   else
     !"error writing the name: "+ name
   fi
   odbcclose
 else
  !"the database can not be opened"
 fi
}
```

❒ Limits - only one database can be opened at the same time
- do not forget to declare the name of the datasource in the odbc32-manager.
- these functions are not supported under OS/2

❒ General information

ODBC functions allows to access various types of databases with only one command set. The only thing you have to do, is to declare the database in the ODBC32 manager of your system.

To add a new datasource, simply open the system control folder and click on the ODBC32bit icon. Then select ADD and follow the instructions.

Name:	odbcappend	
Inputs:		none
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:		Data Source Admininstrator
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function appends a new record to the opened database.

❒ Example

/* appends a new record at the end of "Test database" "Personal"*/

```
{
  odbcuse  "Test database" "Personal"
  if  dbok
    odbcbot
    odbcappend
    if  dbok
      odbcput  "Company"  "New Voice Zurich"
    else
      !"no new record can not be appended"
    fi
    odbcclose
  else
    !"the database can not be opened"
  fi
}
```

odbcbottom

goes to the last record

Name:	odbcbot[tom]	
Inputs:		none
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:		Data Source Admininstrator
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function goes to the bottom of the opened database.

❒ Example

/* appends a new record at the end of "Test database" "Personal"*/

```
{
  odbcuse  "Test database" "Personal"
  if  dbok
    odbcbot
    odbcappend
    if  dbok
      odbcput  "Company"  "New Voice Zurich"
    else
      !"no new record can not be appended"
    fi
    odbcclose
  else
    !"the database can not be opened"
  fi
}
```

odbcclose

close the opened database

Name: dbcclose
Inputs: none

Returns: dbok= 1 if successfull
dbok= 0 if failure

Includes: Data Source Admininstrator

Category: Database - Commands New Voice Tool - Swiss

❒ Description

This function closes the database opened previosly with dbuse.

❒ Example

/* appends a new record at the end of "Test database" "Personal"*/

```
{
  odbcuse  "Test database" "Personal"
  if  dbok
    odbcbot
    odbcappend
    if  dbok
      odbcput  "Company"  "New Voice Zurich"
    else
      !"no new record can not be appended"
    fi
    odbcclose
  else
    !"the database can not be opened"
  fi
}
```

odbcdelete

delete the active record

Name:	odbcdel[ete]	
Inputs:		none
Returns:	dbok= 1	if successfull
	obok= 0	if failure
Includes:		Data Source Admininstrator
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function deletes the active record of the opened database.

❒ Example

/* deletes the first record of vm001.dbf */

/* deletes the toplevel record of "Test database" "Personal"*/

```
{
  odbcuse  "Test database" "Personal"
  if  dbok
    odbctop
    odbcdel
    odbcclose
  else
    !"the database can not be opened"
  fi
}
```

odbcfind

find a value in a field

Name:	odbcfind <fieldname> <str_var | int_var> -next	
Inputs:	fieldname	fieldname in the database
	str_var	variable keeping the string
	int_var	variable keeping the integer
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Option:	-next	finds next entry
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function finds the given value in the selected field of the opened database.

❒ Example

```
/* finds the given expression */

str  name          /* string to keep the name */

{
 odbcuse  "Test database" "Personal"
 if  dbok
  name= "New Voice Zurich"
  odbcfind  "Name"  name
  if  dbok
   !"the name <"+ name+ "> found"
   odbcfind  "Name"  name  -next
   if  dbok
    !"the name <"+ name+ "> found a second time"
   else
    !"the name <"+ name+ "> not found a second time"
   fi
  else
   !"the name <"+ name+ "> not found"
  fi
  odbcclose
 else
  !"the database can not be opened"
 fi
}
```

odbcget

get the value of a field

Name:	odbcget <fieldname> [str_var | int_var]	
Inputs:	fieldname	fieldname in the database
	str_var	variable to store the string
	int_var	variable to store the integer
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function gets the value of a field of the opened database. The record is the active record
set with odbctop, odbcbot, odbcseek, odbcgo or odbcskip.

❒ Example

```
/* gets the name of the top level record */

str  name          /* string to keep the name */

{
  odbcuse  "Test database" "Personal"
  if  dbok
    odbctop
    odbcget  "Name"  name
    if  dbok
      !"the name is <"+ name+ ">"
    else
      !"error reading the name"
    fi
    odbcclose
  else
   !"the database can not be opened"
  fi
}
```

Name:	odbcput <fieldname> [str_var | int_var]	
Inputs:	fieldname	fieldname in the database
	str_var	variable keeping the string
	int_var	variable keeping the integer
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function puts a value into a field of the opened database. The record is the active record
set with odbctop, odbcbot, odbcseek, odbcgo or odbcskip.

❒ Example

```
/* sets the name of the top level record */

int  name          /* string to keep the name */

{
  odbcuse  "Test database" "Personal"
  if  dbok
    odbctop
    name= "New Voice Zurich"
    odbcput  "Name"  name
    if  dbok
      !"the name is set to <"+ name+ ">"
    else
      !"error writing the name: "+ name
    fi
    odbcclose
  else
   !"the database can not be opened"
  fi
}
```

odbcrecno

gets the record number

Name: odbcrecno <record_number>
Inputs: int record_number — variable to return the value

Returns: dbok= 1 — if successfull
dbok= 0 — if failure

Includes: Data Source Admininstrator

Category: Database - Commands — New Voice Tool - Swiss

❒ Description

This function gets the record number of the actual record of the database.

❒ Example

```
/* gets the record number of the last record */

int  record_number      /* integer to keep the record nbr */

{
 odbcuse  "Test database" "Personal"
 if  dbok
   odbcbot
   if  dbok
     odbcrecno  record_number
     !"the record number is <"+ record_number+ ">"
   else
     !"error going to the bottom of the database"
   fi
   odbcclose
 else
  !"the database can not be opened"
 fi
}
```

❒ Limits - the record number is the number given during creation time und does not allways correspond with the number of the record in the database.

Name:	odbcsel < table>	
Inputs:	str table	name of table to select
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:	ODBC32 Server	odbc32-environment
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function selects the given table of the previous opened datasource

❒ Example

```
/* sets the name of the top level record */

int  name          /* string to keep the name */
{
  odbcuse  "Test database"
  if  dbok= 0
    !"the database can not be opened"
    hangup
  fi
  odbcsel  "personal"
  if  dbok
    odbctop
    name= "New Voice Zurich"
    odbcput  "Name"  name
    if  dbok
      !"the name is set to <"+ name+ ">"
    else
      !"error writing the name: "+ name
    fi
    odbcclose
  else
    !"the database table can not be selected"
  fi
}
```

❒ Limits - only one database can be opened at the same time
- do not forget to declare the name of the datasource in the odbc32-manager.
- these functions are not supported under OS/2

Name: odbcskip [number]
Inputs: number skip number of records

Returns: dbok= 1 if successfull
dbok= 0 if failure

Default: 1

Includes: Data Source Admininstrator

Category: Database - Commands New Voice Tool - Swiss

❒ Description

This function skips a given number of records. If no number is given, the function will skip to the next record.

❒ Example

```
/* skips through the database                    */
str  name          /* integer to keep the status */
{
 odbcuse  "Test database" "Personal"
 if  dbok
  odbctop
  odbcget  "Name"  name
  while  dbok  and  name # "New Voice Zurich"
   odbcskip  1
   if dbok
    odbcget  "Name"  name
   else
    !"end of database reached, name not found"
   fi
  wend
  odbcclose
  if name= "New Voice Zurich"
   !"entry found"
  else
   !"entry not found"
  fi
 else
  !"the database can not be opened"
 fi
}
```

odbctop

goes to the top record

Name:	odbctop	
Inputs:		none
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:		Data Source Admininstrator
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function goes to the top record of the database.

❒ Example

```
/* sets the name of the top level record */

int  name          /* string to keep the name */

{
  odbcuse  "Test database" "Personal"
  if  dbok
    odbctop
    name= "New Voice Zurich"
    odbcput  "Name"  name
    if  dbok
      !"the name is set to <"+ name+ ">"
    else
      !"error writing the name: "+ name
    fi
    odbcclose
  else
    !"the database can not be opened"
  fi
}
```

odbcuse

opens a database

Name:	odbcuse <datasource> [table]	
Inputs:	str datasource	name of datasource
	str table	name of table to open
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:	ODBC32 Server	odbc32-environment
Category:	Database - Commands	New Voice Tool - Swiss

❒ Description

This function opens the given data source.

❒ Example

```
/* sets the name of the top level record */
int  name          /* string to keep the name */
{
  odbcuse  "Test database" "Personal"
  if  dbok
    odbctop
    name= "New Voice Zurich"
    odbcput  "Name"  name
    if  dbok
      !"the name is set to <"+ name+ ">"
    else
      !"error writing the name: "+ name
    fi
    odbcclose
  else
   !"the database can not be opened"
  fi
}
```

❒ Limits - only one database can be opened at the same time
- do not forget to declare the name of the datasource in the odbc32-manager.
- these functions are not supported under OS/2

❒ General information

ODBC functions allows to access various types of databases with only one command set. The only thing you have to do, is to declare the database in the ODBC32 manager of your system.

To add a new datasource, simply open the system control folder and click on the ODBC32bit icon. Then select ADD and follow the instructions.

Category: Database - Commands New Voice Tool - Swiss

❒ **Tip**

With ODBC not only a Database, but a datasource is opened. Therefore you have to add a table. In Access or Excel, this is the name of the table or map. In dBase files this will be the name of the database.

❒ **Example 1**

```
/* reads the name of the language directory kept in the top level record */
{
 odbcuse  "nvtini.dbf " "nvtini"
 if  dbok
   odbctop
   odbcget  "T02"  nvtini_lg1
   if  dbok
     !"the language direcotry 1 is: <"+ nvtini_lg1+  ">"
   fi
   odbcclose
fi
}
```

❒ **Example 2**

```
/* reads the name of the language directory kept in the top level record */
{
 odbcuse  "nvtini.dbf "
 if  dbok
   odbcsel  “nvtini”
   odbctop
   odbcget  "T02"  nvtini_lg1
   if  dbok
     !"the language direcotry 1 is: <"+ nvtini_lg1+  ">"
   fi
   odbcclose
 fi
}
```

❒ **Limits** - only one database can be opened at the same time
- do not forget to declare the name of the datasource in the odbc32-manager.
- these functions are not supported under OS/2

❒ **General information**

ODBC functions allows to access various types of databases with only one command set. The only thing you have to do, is to declare the database in the ODBC32 manager of your system.

To add a new datasource, simply open the system control folder and click on the ODBC32bit icon. Then select ADD and follow the instructions.

odbcuse

opens a database

Name:	odbcuse <datasource> [table] -name name –pwd password	
Inputs:	str datasource	name of datasource
	str table	name of table to open
	str name	name of operator
	str password	password of operator
Returns:	dbok= 1	if successfull
	dbok= 0	if failure
Includes:	ODBC32 Server	odbc32-environment
Category:	Database - Commands	New Voice Tool - Swiss

❒ Note

With ODBC not only a Database, but a datasource is opened. Therefore you have to add a table. To open an SQL database, you need a name and password of an operator to be able to access the datasource.

❒ Example 1

```
/* reads the name of the language directory kept in the top level record */
{
 odbcuse  "DataEngine " "Accounts"  -name "MrBond" –pwd "007"
 if  dbok
   odbctop
   odbcget  "Account"  nvtini_s1
   if  dbok
     !"the account name 1 is: <"+ nvtini_s1+  ">"
   fi
   odbcclose
fi
}
```

❒ Limits

- only one database can be opened at the same time
- do not forget to declare the name of the datasource in the odbc32-manager.
- these functions are not supported under OS/2

❒ General information

ODBC functions allows to access various types of databases with only one command set. The only thing you have to do, is to declare the database in the ODBC32 manager of your system.

To add a new datasource, simply open the system control folder and click on the ODBC32bit icon. Then select ADD and follow the instructions.

closepipe

close the opened pipe

Name: closepipe
Inputs: none

Returns: st= 1 if successfull
st= 0 if failure

Includes:

Category: Pipe- Commands New Voice Tool - Swiss

❒ Description

This function closes the pipe opened previosly with openpipe.

❒ Example

```
/* sends a pagerinformation to the pagerserver */

{
  !"Enter the pager number"
  getkey  nvtini_s1  10  -time 30

!"Enter the message"
  getkey  nvtini_s2  20  -time 30

  openpipe  "NVPAGER"
  if  st= 0
    !"The pipe NVPAGER is not present or busy"
    hangup
  fi

  nvtini_s1= nvtini_s1+ nvtini_s2
  putpipe  nvtini_s1

  ‘sleep  50
  getpipe  nvtini_s2  -time 50
  closepipe

  !"The result is "+ nvtini_s2
}
```

Name:	getpipe <str_var>	
Inputs:	str_var	variable to store the data
Returns:	st= 1	if successfull
	st= 0	if failure
Category:	Pipe- Commands	New Voice Tool - Swiss

❒ Description

This function gets data from the opened pipe.

❒ Example

```
/* sends a pagerinformation to the pagerserver */

{
  !"Enter the pager number"
  getkey  nvtini_s1  10  -time 30

!"Enter the message"
  getkey  nvtini_s2  20  -time 30

  openpipe  "NVPAGER"
  if  st= 0
    !"The pipe NVPAGER is not present or busy"
    hangup
  fi

  nvtini_s1= nvtini_s1+ nvtini_s2
  putpipe  nvtini_s1

  ‘sleep  50
  getpipe  nvtini_s2  -time 50
  closepipe

  !"The result is "+ nvtini_s2
}
```

openpipe

opens a pipe

Name:	openpipe <pipe_name> [machine]	
Inputs:	str pipe_name	name of the pipe
	str machine	name of the machine
Server:	openpipe <pipe_name> “nvt_pipe_server“ [-time 200]	
Returns:	st= 1	if successfull
	st= 0	if failure
Category:	Pipe- Commands	New Voice Tool - Swiss

❒ Description

This function opens the given pipe as client. If the pipe is not existent, the function returns -1. Whenever the name of the machine is set to “nvt_pipe_server“, the function opens the pipe as server. The timeout allows to close the pipe and return after the given time in 1/10 of seconds.

❒ Example

```
/* sends a pagerinformation to the pagerserver */
{
  !"Enter the pager number"
  getkey  nvtini_s1  10  -time 30
  !"Enter the message"
  getkey  nvtini_s2  20  -time 30
  openpipe  "NVPAGER"
  if  st= 0
    !"The pipe NVPAGER is not present or busy"
    hangup
  fi
  nvtini_s1= nvtini_s1+ nvtini_s2
  putpipe  nvtini_s1
  ‘sleep  50
  getpipe  nvtini_s2  -time 50
  closepipe
  !"The result is "+ nvtini_s2
}
```

❒ Note

When ever the pipe-server is on a remote host, add the name of the machine in the network in the second parameter.
Network / Identification / computer name: NewVoice1
Network / Identification / working group: ...

```
openpipe  “NVPAGER“  “NewVoice1“
```

Name: pipestate

Returns: st= 1 if existent
st= 0 if failure

Category: Pipe- Commands New Voice Tool - Swiss

❒ Description

This function gets the state of the opened pipe.

❒ Example

```
/* sends a pagerinformation to the pagerserver */

{
  !"Enter the pager number"
  getkey  nvtini_s1  10  -time 30

!"Enter the message"
  getkey  nvtini_s2  20  -time 30

  openpipe  "NVPAGER"
  if  st= 0
    !"The pipe NVPAGER is not present or busy"
    hangup
  fi

  nvtini_s1= nvtini_s1+ nvtini_s2
  putpipe  nvtini_s1

  sleep  50
  getpipe  nvtini_s2
  closepipe

  !"The result is "+ nvtini_s2
}
```

Name:	putpipe < str_var>	
Inputs:	str_var	variable keeping the string
Returns:	st= 1	if successfull
	st= 0	if failure
Category:	Pipe- Commands	New Voice Tool - Swiss

❒ Description

This function puts data on the opened pipe.

❒ Example

```
/* sends a pagerinformation to the pagerserver */

{
  !"Enter the pager number"
  getkey  nvtini_s1  10  -time 30

!"Enter the message"
  getkey  nvtini_s2  20  -time 30

  openpipe  "NVPAGER"
  if  st= 0
    !"The pipe NVPAGER is not present or busy"
    hangup
  fi

  nvtini_s1= nvtini_s1+ nvtini_s2
  putpipe  nvtini_s1

  ‘sleep  50
  getpipe  nvtini_s2  -time 50
  closepipe

  !"The result is "+ nvtini_s2
}
```

Category: Pipe- Commands New Voice Tool - Swiss

❒ Demo-Program

/***\	/***\
* *	* *
* Script : pipeserv *	* Script: pipeclie CLIENT *
* *	* *
***/	***/
{	{
!"Start of module [pipesrv]"	!"Start of module [pipeclt]"
openpipe "PAGER" "nvt_pipe_server" -time 200	‘ wait until pipe created by server
	sleep 50
!"open"+st	openpipe "PAGER"
if st<1	!"open pipe"+ st
!"Unable to open pipe PAGER"	if st<1
hangup	!"Unable to open pipe PAGER"
fi	hangup
getpipe nvtini_s1	fi
!"get pipe data "+st	
!"data received: "+ nvtini_s1	putpipe "hallo"
putpipe "hello"	!"put pipe data "+ st
!"put pipe data: "+st	
‘wait until client got sent data	getpipe nvtini_s1
sleep 50	
closepipe	!"get pipe data "+ st
!"pipe closed "+st	!"received data: "+ nvtini_s1
}	
	closepipe
	!"close pipe "+ st
	}

As client, try to open the pipe during several seconds, as the pipe might be occupied by another client ! (while st< 1 and loopcount< 10)

Whenever the pipe-server is located on a distant machine, add in the second parameter of the open command the name of the machine:
Network / Identification / computer name: NewVoice1
Network / Identification / working group: ...

open “PAGER“ “NewVoice1“

getrandom

returns a random number

Name:	getrandom [filename] [options]	
Inputs:	str filename	complete vox-filename
Options:	-d[ir] str directory	location of file
Returns:	st= 0 .. 32767	if no [vox-file] given
	st= one_part_of_vox_file	if [vox-file] passed
Includes:		none
Category:	Common Vox - Commands	New Voice Tool - Swiss

❒ Description

This function generates a random number between 0 and 32767. If a multipart voxfile is given, getrandom returns the number of one part in the given voxfile.
It is important, that all parts of the multipart voxfile between part one and the higest part contain a message, as the random number will be a number between 1 and the higest part found in the vox-file.

❒ Example

```
/* plays random greetings to the caller                */
int random

{
  getrandom  "greeting.vox"  -dir "c:\nvt\"
  random= st
  !"30 different ways to say hello. actual text: "+ random
  play  "greeting.vox"  "c:\nvt\"  -part random

  getrandom
  random= st
  if random= 9999
    !"congratulations, you won the big... "
    play  "youwin.vox"  "c:\nvt\"
  else
    !"sorry, next time you'll...
    play  "nexttime.vox"  "c:\nvt\"
  fi
}
```

speak -amount

speak a given amount

Name:	speak -amount <value> [-options]	
Inputs:	str value	keeps the string to speak
Options:	-file "betrag.vox"	filename with vox-parts
	-type "dt"	language / syntax
Returns:	st= 0	normal termination
	st= -1	error in vox-file
	st= -2	language not supported
	st= 4	loop current drop detected
	st= 7	fast busy tone detected
Includes:	betrag.vox	specific vox-file
Category:	Extended Vox - Commands	New Voice Tool - Swiss

❒ Description

This function speaks the given amount in the given language. If the specific vox-file contains the substring "betrag", the german syntax-structure / -type "dt" is expected automatically.

❒ Example

```
//Program to test the function speak -amount     spktst.nvt
//start: nvt <liniennummer> spktst            new voice 94

str betrag
{
  betrag= "1"
  while betrag# "0"
          !"please enter the amount (0 for exit):" ?betrag
      'betrag.vox is indicating German Syntax
          speak -amount betrag -file "betrag.vox"
  wend
}
```

❒ Note

If specific languages are needed, please contact your NVT-representative.

Specific Vox-File BETRAG.VOX

To speak an amount in German language, the following Vox-File-Structure is used:

part	text				
1	Eins	2	Zwei	3	Drei
4	Vier	5	Fünf	6	Sechs
7	Sieben	8	Acht	9	Neun
10	Zehn	11	Elf	12	Zwölf
13	Dreizehn	14	Vierzehn	15	Fünfzehn
16	Sechzehn	17	Siebzehn	18	Achtzehn
19	Neunzehn	20	und		
21	Einun	22	Zweiun	23	Dreiun
24	Vierun	25	Fünfun	26	Sechsun
27	Siebenun	28	Achtun	29	Neunun
30	Ein	31	Eine	32	Zwanzig
33	Dreissig	34	Vierzig	35	Fünfzig
36	Sechzig	37	Siebzig	38	Achzig
39	Neunzig	40	Hundert		
41	Tausend	42	Million	43	Millionen
44	Milliarde	45	Milliarden	46	zu gross
47	Punkt	48	minus	49	plus
50	null	51	komma	52	prozent
53	Franken	54	Rappen		

Sample program: R_BETRAG.NVT

1) to Record the vox-file-parts B1 to B54

```
//Programm to record B1 to B54 for speak -amout -f BETRAG.VOX
//Filename: R_BETRAG.NVT                    new voice 1994
int i
str fname
{
  i= 1
  while i< 55
          fname= "b"+ i
          !fname
          rec fname
          i= i+ 1
  wend
}
```

The function NVTCMD VOXMERGE will pack the voxparts B1-B54 into the voxfile BETRAG.VOX.

First create the ascii-file BETRAG.LST containing all filenames B1..B54 on 54 lines. Ex.:
[c:\nvt]dir B3. /b > betrag.lst

Then start [c:\nvt]nvtcmd voxmerge betrag.vox betrag.lst 1 3

Standard Vox-Files for SPEAK -amount "1234.65" -file BETRAG.VOX

To speak an amount in German language, the following Vox-File-Structure is used:

1	Eins	2	Zwei	3	Drei	4	Vier	5	Fünf
6	Sechs	7	Sieben	8	Acht	9	Neun	10	Zehn
11	Elf	12	Zwölf	13	Dreizehn	14	Vierzehn	15	Fünfzehn
16	Sechzehn	17	Siebzehn	18	Achtzehn		19Neunzehn	20	und
21	Einund	22	Zweiund	23	Dreiund	24	Vierund	25	Fünfund
26	Sechsund	27	Siebenund	28	Achtund		29 Neunund	30	Ein
31	Eine	32	Zwanzig	33	Dreissig	34	Vierzig	35	Fünfzig
36	Sechzig	37	Siebzig	38	Achzig	39	Neunzig	40	Hundert
41	Tausend	42	Million	43	Millionen	44	Milliarde	45	Milliarden
46	zu gross	47	Punkt	48	minus	49	plus	50	null
51	komma	52	prozent	53	Franken		54 Rappen		

The commands options are:

speak -amount "1234.65" -file "betrag.vox" -type "dt"

-amount indicates the way the number "1234.65" will be spoken
-file adds the vox-file containig the parts 1 to 54
-dir indicates the directory-location of the vox-file
-type "structure" indicates the structure of the filename

The types "ch" and "dt" are indicating the german filestructur betrag.vox.

This indication is not needed, if the filename contains the substring "betrag".

Interpretation roules:

Cents with 2 digits will be spoken in one number (with leading zero, if less than 10), others will be spoken litteral, one digit after the other.

The range of amounts is +/-2'147'483'000.00 down to .001

Note: The positions komma, Prozent, Franken, Rappen are for future use. Franken will be played together with Punkt, and Rappen at the End of the cents, if any.

speak -string

speak a given string literal

Name:	speak -string <value> [-options]	
Inputs:	str value	keeps the string to speak
Options:	-file "alphabet.vox"	filename with vox-parts
	-dir "c:\dtvox\"	directory of vox-file (-file)
Returns:	st= 0	normal termination
	st= -1	error in vox-file
	st= -2	language not supported
	st= 4	loop current drop detected
	st= 7	fast busy tone detected
Includes:	alphabet.vox	specific vox-file
Category:	Extended Vox - Commands	New Voice Tool - Swiss

❒ Description

This function speaks the given character by character in the language of the voxfile. Use the option -dir to define the location of the voxfile containing the vox-parts.

❒ Example

```
//spktst.nvt:  speaks a phone-number with speak -string
//start: nvt <liniennummer> spktst            new voice 94

str phone_nr
{
  if nvtini_lg1= ""
    nvtini_lg1= "c:\frvox\"
  fi
  phone_nr= "0041 1 364 1991"
  speak -string phone_nr -dir nvtini_lg1
}
```

❒ Note

Record your vox-file alphabet.vox as described on the next page. For specific needed, please contact your NVT-representative.

To speak a string character by character, the following Vox-File-Structure is used:

1		2		3		4		5	
6		7		8		9		10	
11		12		13		14		15	
16		17		18		19		20	
21		22		23		24		25	
26		27		28		29		30	
31		32		33		34		35	
36		37		38		39		40	
41		42		43		44		45	
46		47		48	null	49	eins	50	zwei
51	drei	52	vier	53	fuenf	54	sechs	55	sieben
56	acht	57		58	neun	59		60	61
62		63		65	A	66	B	67	C
68	D	69	E	70	F	71	G	72	H
73	I	74	J	75	K	76	L	77	M
78	N	79	O	80	P	81	Q	82	R
83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91		92	

The commands options are:

speak -string "0041 1 364 19 91" -dir "c:\frvox\"

Name:	speak -date [date] [-dir directory]	speak -time [time] speak -datetime [datetime]
Inputs:		If no date is given, the actual date and/or time is spoken
Options:	str date	date of the form DD.MM
	str time	time of the form hh:mm
	str datetime	date and time of the form DD.MM,hh:mm
	str directory	vox-file directory
Returns:	st= 0	normal termination
	st= -1	error in vox-file
Includes:	date_day.vox date_day.vox	date specific vox-file time_hr.vox time_min.vox time specific vox-file
Category:	Extended Vox - Commands	New Voice Tool - Swiss

❒ Description

This function speaks the given date and/or time in the language of the voxfile. If no values are given, the actual date and time are spoken. Find the structure of the vox-files on the next page.

❒ Example

```
//spktst.nvt:  speaks the actual date and time and the
// recording date and time of a message   new voice tool 95

str  recdate  rectime
str  filename
{
  // speak actual date and time
  speak  -datetime

  // get the first message of mailbox 100 and play date & time
  playmbx  100  -first  filename
  getmsg  filename  -date  recdate  -time  rectime
  speak  -date  recdate
  speak  -time  rectime
}
```

Standard Vox-Files for SPEAK -date -time -datetime

To speak a date and / or time, the following Vox-Files and Structures are used:

date_day.vox: and its parts:

1	First	2	Second	3	Third	4	Fourth
5	Fifth	6	Sixth	7	Seventh	8	Eighth
9	Ninth	10	Tenth	11	Eleventh	12	Twelfth
13	13th	14	14th	15	15th	16	16th
17	17th	18	18th	19	19th	20	20th
21	21th	22	22th	23	23th	24	24th
25	25th	26	26th	27	27th	28	28th
29	29th	30	30th	31	31th		

date_mt.vox: and its parts:

1	January	2	February	3	March	4	April
5	May	6	June	7	July	8	August
9	September	10	October	11	November	12	December

time_hr.vox and its parts:

1	at Zero..	2	at One..	3	at Two..	4	at Three..
5	at Four..	6	at Five..	7	at Six..	8	at Seven
9	at Eight..	10	at Nine..	11	at Ten..	12	at Eleven..
13	at Twelve..	14	at Thirteen..	15	at Fourteen..	16	at Fifteen..
17	at Sixteen..	18	at Seventeen..	19	at Eightteen..	20	at Nineteen..
21	at Twenty..	22	at Twentyone..	23	at Twentytwo..	24	at Twentythree

time_min and its parts:

1	Zero	6	Five	11	Ten	16	Fifteen
21	Twenty	26	Twentyfive	31	Thirty	36	Thirtyfive
41	Fourty	46	Fourtyfive	51	Fifty	56	Fiftyfive

speak -amount

speak a given amount

Name:	speak -amount <value> [-options]	
Inputs:	str value	keeps the string to speak
Options:	-file "betrag.vox"	filename with vox-parts
	-type "dt"	language / syntax
Returns:	st= 0	normal termination
	st= -1	error in vox-file
	st= -2	language not supported
	st= 4	loop current drop detected
	st= 7	fast busy tone detected
Includes:	betrag.vox	specific vox-file
Category:	Extended Vox - Commands	New Voice Tool - Swiss

❒ Description

This function speaks the given amount in the given language. If the specific vox-file contains the substring "betrag", the german syntax-structure / -type "dt" is expected automatically.

❒ Example

```
//Program to test the function speak -amount     spktst.nvt
//start: nvt <liniennummer> spktst            new voice 94

str betrag
{
  betrag= "1"
  while betrag# "0"
        !"please enter the amount (0 for exit):" ?betrag
        'betrag.vox is indicating German Syntax
        speak -amount betrag -file "betrag.vox"
  wend
}
```

❒ Note

If specific languages are needed, please contact your NVT-representative.
If specific formats are needed, please contact your NVT-representative.

Specific Vox-File BETRAG.VOX

To speak an amount in German language, the following Vox-File-Structure is used:

part	text	part	text	part	text
1	Eins	2	Zwei	3	Drei
4	Vier	5	Fünf	6	Sechs
7	Sieben	8	Acht	9	Neun
10	Zehn	11	Elf	12	Zwölf
13	Dreizehn	14	Vierzehn	15	Fünfzehn
16	Sechzehn	17	Siebzehn	18	Achtzehn
19	Neunzehn	20	und		
21	Einun	22	Zweiun	23	Dreiun
24	Vierun	25	Fünfun	26	Sechsun
27	Siebenun	28	Achtun	29	Neunun
30	Ein	31	Eine	32	Zwanzig
33	Dreissig	34	Vierzig	35	Fünfzig
36	Sechzig	37	Siebzig	38	Achzig
39	Neunzig	40	Hundert		
41	Tausend	42	Million	43	Millionen
44	Milliarde	45	Milliarden	46	zu gross
47	Punkt	48	minus	49	plus
50	null	51	komma	52	prozent
53	Franken	54	Rappen		

❒ Note

If specific languages are needed, please contact your NVT-representative.

The positions komma, Prozent, Franken, Rappen are for future use. Franken will be played together with Punkt, and Rappen at the End of the cents, if any.

❒ Explication

As explained before, all elements of a spoken number between +1999999.99 and -1999999.99 are stored in one vox-file. One way to record your own specific vox-file is to use a Voice-Editor. Simply take a copy of a vox-file recorded with New Voice Tool and delete its content. Then copy / paste or record all parts you need for your application. Please check the recordquality you desire.

Another way is, to record each part into a separate file and then merging the parts into one file using NVTDLG.EXE VOXMERGE:

Sample Program to record Specific Vox-File BETRAG.VOX

```
//Programm to record B1 to B54 for speak -amout -f BETRAG.VOX
//Filename: R_BETRAG.NVT                new voice 1994
int i
str fname
{
 i= 1
 while i< 55
        fname= "b"+ i
        !"record into: "+ fname
        sleep 30
        rec fname
        i= i+ 1
 wend
}
```

The function NVTDLG VOXMERGE will pack the voxparts B1-B54 into the voxfile BETRAG.VOX.
--> First create the ascii-file BETRAG.LST containing all filenames
B1..B54 on 54 lines. Ex.: [c:\nvt]dir B*. /b > betrag.lst
--> Then start [c:\nvt]nvtdlg voxmerge betrag.vox betrag.lst 1 3

Sample Program to play Specific Vox-File BETRAG.VOX

```
//Programm to play all parts of BETRAG.VOX (from 1 to 54)
//Filename: P_BETRAG.NVT                new voice 1994
int i
{
 i= 1
 while i< 55
        !"play part: "+ i
        play "betrag.vox"  -part i
        i= i+ 1
 wend
 }
```

Name:	speak -ttsstr <text>	
Inputs:	str text	keeps the string to speak
Returns:	st= 0	normal termination
	st= -1	error in convertion
Includes:	BeSTspeech T-T-S	Text-To-Speech Board
Category:	Extended Vox - Commands	New Voice Tool - Swiss

❒ Description

This function speaks the given string using an additional board. The TTS board is a complementary board to your voice-boards, converting ASCII information to computer-generated speech on demand. The TTS board is connected through the PEB bus and provides your system with (actually) one language, English, Germany, French or Italien.

Text-To-Speech quickly and efficiently relays information because it digitally synthesizes the voice. There is no need to wait for a human recording. TTS capabilities can be used with a variety of applications:

o Name and Address: - Reverse Directory Assistance
o Frequently updated information - Email and News Summary
o Infrequently accessed information: - Archives
o Frequently added information - Lists of inventory items
o Particulary Information - Alarming Systems

❒ Example

```
//Program to test the function speak -ttsstr      ttsstr.nvt
//start: nvt <linenummer> ttsstr              new voice 94

str text
{
  text=  "please write your message and hit the enter key"
  strlen text
  while st> 2
        speak  -ttsstr  text
        !"please enter the message (' ' for exit):"
        ?text
        strlen text
  wend
}
```

Specific Control-Sequences

To speak a message slower, faster or just different to the standard settings, use the following sequences:

Voice Tables, base:	**~v1]** (~v1] .. ~v3])	"~v1] average sized person" "~v2] small ~v3] large person"

Note: ~vx] resets the previously defined ~fy] settings.

Speaking rate:	**~r0]** (~r-100] .. ~r200])	"~r-100] fast, ~r50] slow"
Fundamental frequency:	**~f80]** (~f39] .. ~f200])	"~f50] low pitch, ~f150] high"
Overall Gain:	**~g0]** (~g-50] .. ~g10])	"~g-5] low, ~g5] louder"
Unvoiced Gain:	**~u0]** (~u-50] .. ~u20])	"~u-5] low's', ~g5] louder 's'"
Voiced Exitation function:	**~e0]** (~e0] .. ~e2])	"~e2] whispering, ~e0] normal"

Special punctuation marks

To control accents in pronunciation, use the following marks:

accents the following word:	**~+**	"without accent, with ~+accent"
removes accents from following word:	**~-**	"with accent, without ~-accent"
accents with low pitch:	**~_**	"without accent, ~_with accent!"
special emphasis the following word:	**~^**	"normal, with ~^special accent"
Question rising intonation:	~?	"What color is your ~?apple"

❒ Note

If other changes and pronounciations are needed, please contact your NVT-representative.

Table of Resets

Alphabetic and numerical text resets:

alpha literal reset:	~n1,0]	" ~n1,0] ~n1,1] ~n1,2] "
arithmetic pronunciation reset:	~n5,0]	" ~n5,0] ~n8,1] "
control character pronunciation reset:	~n8,0]	" ~n8,0] ~n8,1] "
digit literal reset:	~n2,0]	" ~n2,0] ~n2,1] "
full number reset:	~n6,0]	" ~n6,0] ~n6,1] "
prosodic punctuation literal reset:	~n3,0]	" ~n3,0] ~n3,1] "
stop abbreviation rponunciation reset:	~n10,0]	" ~n10,0] ~n10,1] "
stop capital letter pronunciation reset:	~n11,0]	" ~n11,0] ~n11,1] "
stop time pronunciation reset:	~n9,0]	" ~n9,0] ~n9,1] "
uppercase text reset:	~n7,0]	" ~n7,0] ~n7,1] "

Field resets:

personal name:	~z1,0]	" ~z1,0] ~z1,1] "
personal title:	~z2,0]	" ~z2,0] ~z2,1] "
organization name:	~z3,0]	" ~z3,0] ~z3,1] "
street address:	~z4,0]	" ~z4,0] ~z4,1] "
City name:	~z5,0]	" ~z5,0] ~z5,1] "
state or province, zip code:	~z6,0]	" ~z6,0] ~z6,1] "
country:	~z7,0]	" ~z7,0] ~z7,1] "
time:	~z8,0]	" ~z8,0] ~z8,1] "

❒ **Note**

If other changes and pronounciations are needed, please contact your NVT-representative.

Name:	strexchange <in_string> <old_part> <new_part> [result]	
Inputs:	str in_string	string to modify
	str old_part	string part to exchange
	str new_part	string part to insert
	[str_var result_string]	to write the result in (option)
Returns:	st>= 0	number of elements exchanged
	st= -1	old_string longer then in_string
	st= -2	in_string constant and no result
Special:	str new_part = “nvt_comment_sign“	replaces all ‘old_part’ against “
	nvt_carriage_return / nvt_line_feed / nvt_form_feed / nvt_back_space / nvt_ring_bell / nvt_hex_file / nvt_char_file	
Includes:		none
Category:	String Functions	New Voice Tool - Swiss

❒ Description

This function seeks for old_part in the given in_string and replaces all old_part against new_part.
If no result string (must be of type string variable) is given, the in_string will be modified. If so, make shure, that the in_string is of type string variable.
If new_string contains the word “nvt_comment_sign“ the old_part will be automatically exchanged against the text delemiter sign “

❒ Example

```
/* generate.nvt generates the program play.nvt       */
str  company      /* keeps the company name          */
str  new_company   /* keeps the modified company name */
{
  company= “/* new voice zurich */“
  strexchange company “zurich“ “swiss“ new_company
  ‘writes “/* new voice swiss */“ into result_string new_company

  fwriteln  “play.nvt“  new_company
  fwriteln  “play.nvt“  “{“
  strxchg “  play ~hello.vox~“ “~“ “nvt_comment_sign“ nvtini_s1
  fwriteln  “play.nvt“  nvtini_s1
  ‘adds the line ’  play “hello.vox“’ to the program play.nvt
  fwriteln  “play.nvt“  “}“
}
‘ run   [\nvt]nve.exe generate
‘ compile the generated program [\nvt]nvc.exe play.nvt
‘ run the program on a line [\nvt]nvt.exe 1 play
```

Name: strexchange <in_string> <file_name> <file_type> [result]

Inputs:	str in_string	string to modify
	str file_name	converting list
	str file_type	converting rouls
	[str_var result_string]	to write the result in (option)
Returns:	st>= 0	number of elements exchanged
	st= -1	old_string longer then in_string
	st= -2	in_string constant and no result
Special:	see strxchange 1. part	
Includes:		conversion file
Category:	String Functions	New Voice Tool - Swiss

❒ Description

This function converts all characters of the given string against others, by consulting a list file. This is very usefull, when converting characters for paging or printer-sets.

❒ Example

```
{
nvtini_s1= "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  strxchg  nvtini_s1  "strxchg.lst" "nvt_hex_file"  nvtini_s2
  !"in: "+ nvtini_s1+ "  out: "+ nvtini_s2
}
```

❒ Content of strxchg.lst (type nvt_hex_file)

list	explication	list (cont)	explication	list (cont)	explication
61>41	a-> A	30>39	0->9	68>08	h->BS
62>42	b-> B	31>38	1->8	69>09	i->HT
		32>37	2->7	6A>0A	j->LF
6E>4E	n->N	33>36	3->6	6B>0B	k->VT
6F>4F	o->O	34>35	4->5	6C>0C	l->FF
70>50	p->P	35>34	5->4	6D>0D	m->CR
71>51	q->Q	36>33	6->3	6E>0E	n->SO
		37>32	7->2	6F>0F	o->SI
79->59	y->Y	38>31	8->1		
7A>5A	z-> Z	39>30	9->0		
21>07	!-> BEL	2C>2E	,->.	7E->2D	~->-
		2E>2C	.->,	7F->2D	¦->-

Special: Change Character to Character

Includes: conversion file

Category: String Functions New Voice Tool - Swiss

❒ Description

This function converts all characters of the given string against others, by consulting a list file. This is very usefull, when converting characters for paging or printer-sets.

❒ Example

```
‘  Exchange Ascii-Characters to Telephone keys !
{
  nvtini_s1= "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  strxchg  nvtini_s1  "astrxchg.lst" "nvt_char_file"  nvtini_s2
  !"in: "+ nvtini_s1+ "  out: "+ nvtini_s2
  !""
  strxchg  nvtini_s2  "astrxchg.lst" "nvt_char_file"  nvtini_s1
  !"in: "+ nvtini_s2+ "  out: "+ nvtini_s1
  !""
  nvtini_s1= "abcdefghijklmnopqrstuvwxyz"
  strxchg  nvtini_s1  "astrxchg.lst" "nvt_char_file"  nvtini_s2
  !"in: "+ nvtini_s1+ "  out: "+ nvtini_s2
  !""
}
```

❒ Content of astrxchg.lst (type nvt_char_file)

list	list (cont)	list (cont)	list (cont)
A>2	N>6	a>2	n>6
B>2	O>6	b>2	o>6
C>2	P>7	c>2	p>7
D>3	Q>7	d>3	q>7
E>3	R>7	e>3	r>7
F>3	S>7	f>3	s>7
G>4	T>8	g>4	t>8
H>4	U>8	h>4	u>8
I>4	V>8	i>4	v>8
J>5	W>9	j>5	w>9
K>5	X>9	k>5	x>9
L>5	Y>9	l>5	y>9
M>6	Z>9	m>6	z>9

strlen

get the length of a string

Name:	strlen <string>	
Inputs:	str string	string to evaluate the length
Returns:	st>= 0	length of the given string
Includes:		none
Category:	String Functions	New Voice Tool - Swiss

❒ Description

This function evaluates the number of characters in the given string.

❒ Example

```
/* program to evaluate the number of digits entered */

str  phone_number       /* keeps the entered digits */
{
  !"hi, please enter your phonenumber"

  getkey  phone_number  15  -time 30
  !"the entered number is: <"+ phone_number+ ">"

  strlen  phone_number
  if st> 5
    !"this is an internal phone number"
  else
    !"this is an external phone number"
  fi
}
```

Name:	strpart <string> <start> <length> <result_string>	
Inputs:	str string	base string
	int start	first octet to extract
	int length	number of octet to extract
	str result_string	string containing the part
Returns:	st> 0	length of the result-string
	st= -1	start > lenght of string
Includes:		none
Category:	String Functions	New Voice Tool - Swiss

❒ Description

This function evaluates the number of octets of the given string.

❒ Example

```
/* program extracts a number of octets out of an alarmtext */

str  alarm_text          /* keeps the alarmtext     */
str  text_part           /* receives the extraction  */

{
  !"hi, please enter your alarm-message (form: A1234: urgent"

  ? alarm_text
  !"the entered message is: <"+ alarm_text+ ">"

  strpart  alarm_text  2  4  text_part
  !"the alarm-number is: <"+ text_part+ ">"

  strpart  alarm_text  5  10  text_part
  !"the alarm-message is: <"+ text_part+ ">"
}
```

Name:	strstr <string> <sub_string>	
Inputs:	str string	base string
	str sub_string	substring to seek
Returns:	st> 0	found, start of the substring
	st= -1	substring not found
Includes:		none
Category:	String Functions	New Voice Tool - Swiss

❒ Description

This function seeks a substring in the given string.

❒ Example

```
/* program to seek for 00 in the entered digit-string*/

str  phone_number       /* keeps the entered digits */

{
  !"hi, please enter the phonenumber to dial"

  getkey  phone_number  15  -time 30
  !"the entered number is: <"+ phone_number+ ">"

  strstr  phone_number  "00"
  if st= 1
    !"this is an international number"
  else
    !"this is a local number"
  fi
}
```

sleep

wait during a defined time

Name: sleep <time>
Inputs: int time — time in 1/10 of seconds

Returns: none

Includes: none

Category: Time & Date Functions — New Voice Tool - Swiss

❒ Description

The programm is stoped for a given number of tenth of seconds.

❒ Example

```
/* program with callback function */

str  phone_nbr        /* string to keep the entered nbr */

{
  !"hi, please enter your phonenumber"
  play "enternbr.vox"

  getkey  phone_nbr  15  -time 30
  !"the entered number is: <"+ phone_nbr+ ">"

  !"you'll be called back immediately"
  play "callback.vox"  -cont

  onhook
  sleep 10            /* wait before offhook            */
  offhook

  sleep 10            /* wait for dialtone              */
  bldial  phone_nbr

  sleep 40            /* wait for called person answers */
  !"hi, we are back"
  play  "back.vox"
}
```

Name:	get <type> [data] [data2]	
Inputs:	-time	get actual time
	-date	get actual date
	-datetime	get actual date and time
Returns:	str string_variable	receives date and/or time
Includes:		none
Category:	Time & Date Functions	New Voice Tool - Swiss

❒ Description

The programm gets the actual date and time into the given string variable. If no variable is given, the default variables nvt_time, nvt_date, nvt_hours, nvt_minutes, nvt_seconds are reloaded..

❒ Example

```
/* program with time functions */
{
  !"this program started at seconds: "+ nvt_seconds
  sleep 20
  get  -time
  !"two seconds later we are at second: "+ nvt_seconds

  sleep 20
  get  -time nvtini_s1
  !"this does not change nvt_seconds: "+ nvt_seconds
  !"  but loads the actual time (15:33): "+ nvtini_s1

  get  -time nvtini_s1  nvtini_s2
  !"this loads the actual time in sec from 1970: "+ nvtini_s2

  get  -datetime
  !"another two seconds later we are at second: "+ nvt_seconds

  sleep 20
  get  -datetime nvtini_s2
  !"this does not change nvt_seconds: "+ nvt_seconds
  !" but loads the date (22.12-15:33) to nvtini_s2: "+ nvtini_s2

  get  -difftime  nvtini_s1
  !“the difference “+ nvtini_s1+ “ - “+ nvt_time+ “= “+ st
}
```

Name:	get <type> data [data_2]	
Inputs:	-difftime	get delta time
	-diffdate	get delta date
Returns:	str string_variable	receives date and/or time
	st:	int difference
Includes:		none
Category:	Time & Date Functions	New Voice Tool - Swiss

❒ Description

The programm calculates the number of days between a date to the date of today. It also calculates the difference between hours, minutes or seconds.

❒ Example

```
/* program with time functions */
{
  playmbx  100  -first  fname
  !"get the recording date of a message"
  getmsg  fname  -date  nvtini_s1
  get  -diffdate  nvtini_s1
  !“age of the message: “+st+“(“+nvtini_s1+“ - “+ nvt_date+“)“
  get  -difftime  “18:00“ “07:30“
  !“the difference of 18:00 - 07:30 is: “+ st + “ minutes“
  get  -difftime  “18:00:00“ “17:55:45“
  !“the difference of 18:00:00 - 17:55:45 is: “+ st + “ seconds“
}
```

❒ Alternative

```
int start stop delta
{
  start= nvt_seconds+ nvt_minutes* 60
  sleep 70
  get -time
  stop= nvt_seconds+ nvt_minutes* 60
  if stop> start
    delta= stop- start
    !"dt="+delta+"
("+nvt_hours+":"+nvt_minutes+":"+nvt_seconds+")+"
  else
    delta= stop- start+ 3600
    !"dt= "+ delta+ "
("+nvt_hours+":"+nvt_minutes+":"+nvt_seconds+")-"
  fi
}
```

Name:	get <type> data [data_2]	
Inputs:	-difftime	get delta time
Returns:	str string_variable	receives date and/or time
	st:	int difference
Includes:		none
Category:	Time & Date Functions	New Voice Tool - Swiss

❒ Description

The programm calculates the number of days between a date to the date of today. It also calculates the difference between hours, minutes or seconds.

❒ Example

```
/* difftime.nvt program with time functions */
str s0 s1 s2
{
  get -time s0 s1
  !"get time since 1970: "+ s1
  !"sleep 4 seconds"
  sleep 40
  get -time s0 s2
  !"get time since 1970: "+ s2

  get -difftime s1 s2
  !"diff-time: "+ st+  "    (expected: -4 seconds)"
}
```

❒ Output

```
NVT Optimizing 1-Pass Compiler/Interpreter 5.0 MT
-load code from file <difftime.nve>  - 104 instructions,
 version: 6.0

get time since 1970: 39544E5A
sleep 4 seconds
get time since 1970: 39544E5E
diff-time: -4    (expected: -4 seconds)

free code for difftime
NORMAL TERMINATION
```

Name:	get <type> data [data_2]	
Inputs:	-difftime	get delta time
Returns:	str string_variable	receives date and/or time
	st:	int difference
Includes:		none
Category:	Time & Date Functions	New Voice Tool - Swiss

❒ Description

The programm calculates the number of days between a date to the date of today. It also calculates the difference between hours, minutes or seconds.

❒ Example

```
str s0 s1 s2
int d h m s
{
  !"-------------------file creation and modification--------"
  s1= "nvt_creation_time"
  ffind "difftime.nvt" -time s1
  !"creation date and time: "+ s1
  s2= ""
  ffind "difftime.nvt" -time s2
  !"  modify date and time: "+ s2
  get -difftime s1 s2
  !"difference between creation and last modif: "+ st
  st= st* -1
  m= st/ 60
  s= st- (m* 60)
  h= m/ 60
  d= h/ 24
  h= h- (d* 24)
  m= st/60 - (h* 60)- (d*24*60)
  !"delta: "+d+" day(s) "+h+" hour(s) "+m+" minute(s) "+s+" sec"
  st= d*24*60*60+h*60*60+m*60+s
  !"difference in seconds: "+ st+ " seconds"
```

```
  !"-------------------time calculation-------------------"
  get -time s0 s1
  !"time since 1970: "+ s1+  "  sleep 2 seconds..."
  sleep 20
  get -difftime s1
  !"difference time to now: "+ st+  "    (expected: 2 seconds)"
  get -time s0 s1
  !"time since 1970: "+ s1+  "  sleep 3 seconds"
  sleep 30
  get -time s0 s2
  !"get time since 1970: "+ s2
  get -difftime s1 s2
  !"difference time s1- s2: "+ st+  "    (expected: -3 seconds)"
}
```

Name:	get -ini filename section keyword strvar	
Inputs:	-ini	get data from ini-file
	filename	name of ini-file
	section	[section]-name
	keyword	desired information
	strvar	default value
Returns:	str strvar	receives date and/or time
	st:	stringlength (strvar)
Includes:		none
Category:	Management Functions	New Voice Tool - Swiss

❒ Description

The programm reads a number of keywords to initialize an application. Reading ini-files allows you to read configuration data and settings previously configured through a graphic user interface. The ini-file should be located as every ini-file in your system directory i.e. \windows or \winnt

❒ Example

```
//   filename: read_ini.nvt                          19. 8. 1997
//   purpose:  loads configuration data from ini-file
//
//   nvt.ini   [company]
//             name= New Voice Switzerland
//             tel= 004113641991
//             [support]
//             web= www.NewVoice.ch
//             fax= 004113642652
{
  nvtini_s1= "New Voice Zurich"
  get -ini  "nvt.ini" "company"  "name"  nvtini_s1
  !"company name: "+ nvtini_s1+ " length: "+ st

  nvtini_s2= "111"
  get -ini  "nvt.ini" "company"  "tel"  nvtini_s2
  !"phone number: "+ nvtini_s2+ " length: "+ st

  nvtini_s3= "in construction"
  get -ini  "nvt.ini" "support"  "web"  nvtini_s3
  !"web page: "+ nvtini_s3+ " length: "+ st

  nvtini_s4= "no more info"
  get -ini  "nvt.ini" "support"  "more_info" nvtini_s4
  !"there are "+ nvtini_s4
}
```

Name:	get -license strvar	
Inputs:	-license	get license from hardlock key
Returns:	str strvar	license code
	st:	number of lines licensed
Includes:		none
Category:	Management Functions	New Voice Tool - Swiss

❒ Description

The programm reads the license number and prints the number of valid lines to the screen.

❒ Example

```
//   filename: read_ini.nvt                                    19. 8.
1997
{
}
```

Name:	`recognize <voc-file> [-options] <res_1> <res_2>`	
Inputs:	`str voc-file`	keeps the vocabulary
Options:	`-dir directory`	voc-file location
Returns:	`st= 0`	function returns normally
	`st< 0`	an error occured
	`str <res_1>`	string with best match
	`str <res_2>`	string with second match
Includes:		none
Category:	Voice-Recognition-Commands	New Voice Tool - Swiss

❒ Description

This function plays a beep to the caller and analizes the word, spoken by the caller. It then finds the best match with the given vocabulary.

If the extention of the vocabulary is .voc, scott-voice-recognition is used.

❒ Example

```
/* asks the callers choice                    */

str  result    // word with the best match
str  resalt    // word with the second match
{
  !"hi, please say any digit between 0 and 9"
  recognize  "digits.voc"  -d "c:\voc\"   result  resalt
  if  st>= 0
    !"we recognized the following: "+ resalt+ " or "+ result
  else
    !"we didn't recognize anything"
  fi
}
```

Name:	`conference first_line second_line mode type`	
Inputs:	`int first_line`	own line nuber
	`int second_line`	partners line number
	`int mode = 1 / 0`	connect / disconnect
	`int type = 1`	connect with D/4xx
	`2`	connect with D/300
	`3`	connect with MSI
	`4`	connect with GFax
	`5`	connect Voice over IP
Returns:	`st= 0`	if connection
	`st= -1`	if no connections
Includes:		none
Category:	Extended Vox - Commands	New Voice Tool - Swiss

r Description

This function connects two callers together by switching the two channels together. A log-file is generated in the log-directory: `confNNN_YYYYMMDD.log, (in NNN: the conference number).`

r Note

This function is limited to boards supporting switching. Please contact your NVT-representative in your country or New Voice in Switzerland for further information.

r Example

```
// New Voice Tool Conference Demonstration Program      11.96
// Start Line 1 and Line 2 with the program call_out.nvt
// Then call the Line 1 to be diverted to New Voice.

str  command
cst  connect    1
cst  disconnect 2
{
  !"Hi, your call will be transfered"
  play  "hello.vox"
  ‘start dial_s1 as batch-program on line 2. dial_s1.nvt
  ‘is delivered with the standard package of New Voice design
  command= "nvtque.exe batchpost 2 dial_s1 -s1004113641991“
  !command
  run  command
  !"establish connection with second caller"
  conference  1  2  connect
  !"wait for keystroke # to exit or a loop current drop"
  while key # "#"
    getkey  key  1  -time 1000  -cont
    if  Loopdrop  then  hangup   fi
```

```
  wend
  !"disconnect and hangup"
  conference  nvtini_linenbr  2  disconnect
}
```

r Example

```
// New Voice Tool Conference Demonstration Program      11.96
// Start Line 1 and Line 2 with the program connect.nvt

cst first_caller  = 1 // this number is identical with line 1
cst second_caller = 2 // this number is identical with line 2
cst connect=    1
cst disconnect= 0
{
  if  nvtini_linenbr > second_caller  then
    !"this demo program works only on line 1 and 2, sorry"
    hangup
  fi

  if  nvtini_linenbr =  first_caller
    !"wait for start / press any key to start connection"
    getkey  key  1 -time 1000
    if  Timeout
      !“you did not press any key to launch connect-process“
      hangup
    fi

    !"establish connection with second caller"
    conference  nvtini_linenbr  second_caller  connect
    !"connected: "+ st
    !"wait for a loop current drop or a keystroke # to exit"
    while key # "#"  and  key # "X"
      getkey  key  1  -time 1000  -cont
      if  Loopdrop  then  key = "X"   fi
    wend
    !"disconnect and hangup"
    conference  nvtini_linenbr  2  disconnect
    hangup
  fi

  !"We are on line 2 waiting to be connected"
  play  "hello.vox"
  !"wait for a loop current drop or a keystroke * to exit"
  while key # "*"
    getkey  key  1  -time 1000  -cont
    if  Loopdrop  then hangup  fi
  wend
  !"hangup after keystroke * received"
  !"bye"
}
```

r Example DIVERT.NVT

```
// New Voice Tool Conference Demonstration Program      4.97
// Start Line 1 and Line 2 with the program divert.nvt
str cmd
int i
str fname
str line
int iline

//============================ WAITFORCONNECT
proc waitforconnect{
  i= 20
  fname= "cnx"+nvtini_linenbr+ ".inf"
  fopen fname "write"
  fwrite ""
  fclose

  while i> 0
    fopen fname "read"
    fread cmd
    !"cmd: "+ cmd
    fclose
    //1xx: connect 2xx: busy 0xx: noanswer 5xx: error
    if 0= cmd
      getkey 1 -time 30 -cont
      if Hangup then
        !"caller went onhook"
        fname= "cnx"+ iline+ ".inf"
        fopen fname  "write"
        fwrite "-1"
        fclose
        hangup
      else
        i= i-1
      fi
    else
      return
    fi
  wend
}
```

Example DIVERT.NVT cont

```
proc waitfordisconnect{
  i= 60
  fname= "cnx"+nvtini_linenbr+ ".inf"
  while i> 0
    fopen fname "read"
    fread cmd
    !"cmd: "+ cmd
    fclose
    if -1 # cmd
      getkey 1 -time 30 -cont
      if Hangup then
        !"caller went onhook"
        fname= "cnx"+ iline+ ".inf"
        fopen fname  "write"
        fwrite "-1"
        fclose
        hangup
      else
        i= i-1
      fi
    else
      !"called personne disconnected"
      return
    fi
  wend
}
{//=========================================MAIN
  play "hello.vox"
  cmd= "nvtque postjob 1 div_s1 -s1022161 -i1"+ nvtini_linenbr
  run cmd
  waitforconnect
  if i<= 0
    !"no connection"
    hangup
  fi
  iline= cmd
  if iline<= 100
    !"no answer"
    hangup
  fi
  if iline>= 200
    !"busy or error"
    hangup
  fi
  iline= iline- 100
  conference nvtini_linenbr iline 1
  waitfordisconnect
  conference nvtini_linenbr iline 0
}
```

r Example DIV_S1.NVT started from DIVERT.NVT

```
str cmd
int i
str fname
str line
int iline
int state

//============================ WAITFORDISCONNECT
proc waitfordisconnect{
  i= 60
  fname= "cnx"+nvtini_linenbr+ ".inf"
  while i> 0
    fopen fname "read"
    fread cmd
    !"cmd: "+ cmd
    fclose
    if -1# cmd
      getkey 1 -time 30 -cont
      if Hangup then
        !"caller went onhook"
        fname= "cnx"+ nvtini_i1+ ".inf"
        fopen fname  "write"
        fwrite "-1"
        fclose
        hangup
      else
        i= i-1
      fi
    else
      return
    fi
  wend
}
```

r Example DIV_S1.NVT cont. (started from DIVERT.NVT)

```
{//=====================================MAIN
  fname= "cnx"+nvtini_linenbr+ ".inf"
  fopen fname "write"
  fwrite ""
  fclose
  dial  nvtini_s1 -rings 6
  state= st
  !"call state: "+ state
  cmd= ""
  fopen fname "read"
  fread cmd
  !"cmd: "+ cmd
  fclose
  if -1= cmd
    !"initial caller disconnected"
    hangup
  fi
  if nvtini_linenbr> 9
    cmd= state+ nvtini_linenbr
  else
    cmd= state+ "0"+ nvtini_linenbr
  fi
  fname= "cnx"+ nvtini_i1+ ".inf"
  if state= 1
    fopen fname  "write"
    fwrite cmd
    fclose
    !"start conference with "+ nvtini_i1
    conference nvtini_linenbr nvtini_i1 1
    waitfordisconnect
    conference nvtini_linenbr nvtini_i1 0
  fi
  if state= 2
    fopen fname  "write"
    fwrite cmd
    fclose
  fi
  if state= 0
    fopen fname  "write"
    fwrite cmd
    fclose
  fi
  if  state# 0 and state# 1 and state# 2
    !"error while dialling out"
    fopen fname  "write"
    fwrite cmd
    fclose
  fi
}
```

r Example INFOCONF.NVT

```
/************************************************************
  program: infoconf.nvt

  purpose: one can speak and all other can listen

  concept: on call, the program checks, whether there is
           one on line. file: master.in
           if not, he'll create it ( content: nvtini_linenbr )
           and starts speaking  ( radio, or other )
           if not, he starts listening to the master


************************************************************/
str fname
str line
int iline
int timelimit


//======================================== MASTERCALL
proc mastercall
{
  line= nvtini_linenbr
  fwriteln fname line
  play "himaster.vox" -cont
  if Hangup
    fdelete fname
    hangup
  fi
  timelimit= 15
  while timelimit> 0
    getkey  key  1 -time 200 -cont
    if Hangup
      fdelete fname
      hangup
    fi
    !key
  wend
}
```

r Example INFOCONF.NVT cont.

```
//====================================== CLIENTCALL
proc clientcall
{
  !"you'll be connected to the master"
  play "hiclient.vox"

  fopen fname "read"
  if st>= 0
    fread  nvtini_s4
    fclose
    iline= nvtini_s4
  else
    !"file open error, stop communication"
    hangup
  fi
  conference nvtini_linenbr iline 1
  timelimit= 15
  while timelimit> 0
    getkey  key  1 -time 200 -cont
    if Hangup
      conference nvtini_linenbr iline 0
      hangup
    fi
    !key
  wend
}


//============================================= MAIN
{
  fname= "master.in"
  ffind fname
  if st>= 0
    clientcall
  else
    mastercall
  fi

}
```

r Example RADIOREC.NVT

```
/************************************************************
  program: radiorec.nvt

  purpose: one can speak and all other can listen on radioply

concept: on call, the program records a message in the file
           recplay.vox

************************************************************/
str fname

//=========================================== MAIN
{
  fname= "recplay.vox"
  rec  fname  -time 300

}



/************************************************************
  program: radioply.nvt

  purpose: one can speak on radiorec.nvt and all other can
listen

concept: on call, the program plays the file
           recplay.vox with a special option

************************************************************/
str fname

//=========================================== MAIN
{
  fname= "..\9821\recplay.vox"
  play  fname  -spec 12
}
```

Name:	get -ini ”filename” ”section” ”field” stringvar	
Inputs:	str filename	name of the ini-file
	str section_in_brackets	name of the section
	str fieldname_in_section	name of the field
Returns:	str str_variable	contains the value, if found, else the content will not change
Includes:	ini-file	
Category:	z-other - commands	New Voice Tool - Swiss

r Description

This function reads the value of an initialization variable stored in the ini-file [section]

r Example

```
/*  filename: read_ini.nvt located in your system directory
[company]
name= New Voice Switzerland
tel= 004113641991
[support]
web= www.NewVoice.ch
fax= 004113642652
*/
str sInfo
{
  nvtini_s1= "New Voice Zurich"
  get -ini  "nvt.ini" "company"  "name"  nvtini_s1
  !"company name: "+ nvtini_s1
  !"number of digits: "+ st
  nvtini_s2= "111"
  get -ini  "nvt.ini" "company"  "tel"  nvtini_s2
  !"phone number: "+ nvtini_s2
  nvtini_s3= "in construction"
  get -ini  "nvt.ini" "support"  "web"  nvtini_s3
  !"web page: "+ nvtini_s3
  !"number of digits: "+ st
  nvtini_s4= "unknown"
  get -ini  "nvt.ini" "support"  "fax"  nvtini_s4
  !"fax number: "+ nvtini_s4
  sInfo= "no more info"
  get -ini  "nvt.ini" "support"  "more_info" sInfo
  !"there are "+ sInfo
}
```

Name:	getsocket str_data -time time	
Inputs:	str str_data	string to receive data
	int time	wait-timeout in 1/10 sec
Returns:	int st	0, if port address ok
Includes:	tcp-ip protocol	
Category:	z-other - commands	New Voice Tool - Swiss

r Description

This opens a chat-session as server and gets data from client

r Example

```
// Start with  [nvt]nve.exe socket_s -s24001
proc socketcommunication
{
  !"open socket "+ nvtini_s2+ " as server"
  opensocket  nvtini_s2   "server"
  if st< 0
    !"Error: Socket can not be opened on port "+ nvtini_s2+ " (nvtini_s2)"
    return
  fi
  !"("+ st+ ") socket is open on port "+ nvtini_s2
  !"the socket server is now waiting for a connection on port "+ nvtini_s2
  connectsocket  "" ""
  !"we are connected: "+ st+ "- now get data from the socket client connected with us"
  getsocket  nvtini_s3  -time 100
  !"("+ st+ ") we got the following data: <"+ nvtini_s3+ ">"
  nvtini_s4= "successfull "+nvtini_s3
  !"we answer with the following data: <"+ nvtini_s4+ ">"
  putsocket  nvtini_s4
  !"data sent: "+ st
  !"close socket after 0.5 seconds, end communication with client"
  sleep 5
  closesocket
  !"close: "+ st
}
// ============================MAIN
{
  nvtini_i3= 6
  nvtini_s3= ""
  while nvtini_i3> 0 and nvtini_s3# "exit"
     socketcommunication
     nvtini_i3= nvtini_i3- 1
  wend
}
```

opensocket

as client

Name: opensocket
Inputs: str port — client port address

Returns: int st — 0, if port address ok

Includes: tcp-ip protocol

Category: z-other - commands — New Voice Tool - Swiss

r Description
This opens a chat-session as client

r Example

```
{
 !"opensocket 4003 as client"
 opensocket  "4003"
 !"open: "+ st

 connectsocket  "192.168.1.202"  "4002"
 !"connect: "+ st

 while 1
  !"enter text message: "
  nvtini_s1= "  "
  ?nvtini_s1
  if nvtini_s1= "exit"
    putsocket  nvtini_s1
    sleep 10
    closesocket
    hangup
  fi
  putsocket  nvtini_s1
  '!"put: "+ st

  getsocket  nvtini_s1
  if nvtini_s1= "exit"
    !"server went onhook. Bye"
    closesocket
    hangup
  fi
  !"get: "+ nvtini_s1
 wend

 closesocket
}
```

Name: opensocket
Inputs: str port — server port address

Returns: int st — 0, if port address ok

Includes: tcp-ip protocol

Category: z-other - commands — New Voice Tool - Swiss

r Description
This opens a chat-session as server

r Example

```
{
  !"opensocket 4002 / server"
  opensocket "4002"   "server"
  !"open: "+ st

  !"connect socket / wait for connection"
  connectsocket  "" ""

  while 1
    getsocket  nvtini_s1
    !"data: "+ nvtini_s1
    if nvtini_s1= "exit"
      !"caller went onhook. Bye"
      closesocket
      hangup
    fi

    !"enter text message: "
    nvtini_s1= "  "
    ?nvtini_s1
    if nvtini_s1= "exit"
      putsocket  nvtini_s1
      sleep 10
      closesocket
      hangup
    fi

    putsocket  nvtini_s1
  wend

  closesocket
}
```

putsocket

send data on socket link

Name: putsocket data
Inputs: str data — data to send

Returns: int st — 0, if data successfully sent

ICategory: z-other - commands — New Voice Tool - Swiss

r Description
This program started as client sends data to the server and gets answer

r Example

```
// Start with [nvt]nve.exe socket_c -s1192.169.1.200 -s24001 -s34002
//   or [nvt]nve.exe socket_c -s1192.169.1.200 -s24001 -s34002 -s4text
//   where the ip-address is the address of the socket-server machine
{
  !"Open socket on port "+ nvtini_s3+ " as client"
  opensocket  nvtini_s3
  if st< 0
    !"Error opening socket on port "+ nvtini_s3+ " (nvtini_s3)"
    return
  fi
  !"("+ st+ ") the socket client is open on port "+ nvtini_s3

  !"Now connect client with server on "+ nvtini_s1+ " port "+ nvtini_s2
  connectsocket  nvtini_s1  nvtini_s2
  if st< 0
    !"connection with server on ip "+ nvtini_s1+ " port "+ nvtini_s2+ " failed"
    closesocket
    return
  fi
  !"We are connected on ip "+ nvtini_s1+ " port "+ nvtini_s2
  if nvtini_s4= "" then
     nvtini_s4= "new voice tool- socket link"
  fi
  !"send data to the server: <"+ nvtini_s4+ " >"
  sleep 10
  putsocket  nvtini_s4
  !"("+ st+ ") data sent"

  !"Now get the answer from the server:"
  getsocket  nvtini_s5  -time 100
  !"("+ st+ ") Data received from the server: <"+ nvtini_s5+ ">"

  !"close socket client after 0.5 seconds"
  sleep 5
  closesocket
}
```