NICOTRA

DLL for SELECTION of BELT DRIVEN FANS

For integration into OEM CAE/design programs

running under 32 bit PC operating systems

DLL Version 2

(Release 2.5.4)

Archive version 2.9.0

USER MANUAL

Rev. 13

1. INDEX

2. GENERAL DESCRIPTION

2.1 Scope

The NICOTRA Belt Driven Fan (BDF) selection DLL is a set of precompiled software functions, intended for integration with OEM automated design (Computer Aided Engineering - CAE) programs.

These functions allow interrogation of proprietary archives to obtain a list of available products and, for each selected product, operating point identification, belt drive life and design parameters, polynomial functions for plotting performance graph, dimensional drawing, picture and list of available accessories.

2.2 General properties of programming with DLLs of subroutines

Under Microsoft Windows, programs can employ DLLs (Dynamic Link Libraries), which allow truly modular programming. They are like black boxes, within which all required functions from a main program can be separately stored with a predefined interface. Therefore a large program can be divided into several modules, which can be developed by different programmers. Furthermore there are advantages in testing separately the single modules.

This universal interface also offers the possibility of using these modules within other programs, so that separate CAE programs, for different product types or ranges, can use the same supplier-furnished functions for calculation components.

Product and/or calculation methods updates can also be easily integrated into the CAE programs because only the affected modules (DLL's) have to be updated.

The Nicotra dll for calculation of belt driven fans was also conceived with a modular archive structure, with the objective of making as simple as possible any update or extension of the product databases.

2.3 DLL Versions

Different minor variants of the Nicotra.dll version 2 file have been made available to the users:

- Version 2.0.0 (633 KB, 27-07-01) is a stand-alone dll compatible with Win32 operating systems (Win'98, WinNT, Win2000, WinME) but not with Windows 95. It was distributed with RDH Upgrade packages 2.5 to 2.6.
- Version 2.1.0 (237 KB, 26-10-01) is the same code, recompiled with different settings, to achieve compatibility also with Windows 95. The smaller DLL file requires three additional dlls to be located together with the nicotra.dll file (alternatively they can be located in the \Windows\Sistem32 directory); they are: Dformd.dll (418 KB, 17-11-98), Dforrt.dll (411 KB, 17-11-98) and Msvcrt.dll (261 KB, 01-03-99). It was included in the updated Ventil v. 2.0.0 distribution CD-Rom.
- Version 2.1.1 (237 KB, 27-11-01) is similar to version 2.1.0, but was recompiled with a character string length, of the "List" variable used by GET_PRODUCTS, reduced to 20000 characters, as used by the DLL ver. 1.0.0, to achieve compatibility with existing main programs. This string was extended, in the former two versions, to 30000 characters to compensate for the increased number of fans dealt with by Ventil. This special version is available from Nicotra to assist software developers who wish to keep existing software while updating to the revised and extended mathematical models employed by the new dll ver. 2. See also chapter 4.2.

- Version 2.2.0 (237 KB, 28-01-02) is the same code as version 2.1.0, apart from the correction of a bug which produced a slight overestimate (of roughly 1 dB) of the sound power levels of twin fans only, when calculated according to the BS 848 Part 2, appendix G mathematical model. It has been distributed as part of the ADH CP 2.0 upgrade package.
- Version 2.2.1 (237 KB, 28-01-02) is the same code as version 2.2.0, with the length of the variable List in GET_PRODUCTS reduced again to 20000 characters, to achieve compatibility with code written for the older dll ver. 1.0.0 .
- Version 2.3.0 (245 KB, 19-06-02) is an improved variant of version 2.2.0, with corrections of a couple of bugs which formerly prevented proper calls to the two graphical routines GET_WMF and GET_WMF_DIM_LIST. A “SETDLLPATH” routine was added, with slight changes to the archive search procedures. This version was distributed as part of the RDH Upgrade package 2.6 and of the ADH Upgrade package 2.7.
- Version 2.3.1 (245 KB, 09-10-02) is identical to version 2.2.0, apart from a correction to a mistyped value for a motor shaft diameter. This version is distributed together with the DLL technical documentation for SW developers, as part of the ADH Upgrade package 2.3b and as integral part of Ventil 2.0.2.
- Version 2.3.2 (245 KB, 28-01-03) is identical to version 2.3.1, apart from correction of a bug which sometimes prevented a successful call to the function GET_PICTURE_NAME. This version is distributed together with the DLL technical documentation for SW developers, and as part of the redistributable archive package for ADH and RDH fans.
- Version 2.3.3 was just a development version, not distributed.
- Version 2.4.0 (268 KB, 01-07-04) was modified to introduce support for AMCA 301 noise calculation model. This version is distributed together with the DLL technical documentation for SW developers, and as part of the redistributable archive package for ADH, RDH and AT fans.
- Version 2.5.0 and 2.5.1 were just development versions, not distributed.
- Version 2.5.2 (292 KB, 20-06-05) was modified to improve support for AMCA 301 noise calculation model, including handling outlet-side licensed data. Functions to provide extended noise information (GET_NOISE_DATA), to provide the constants for parabolic margins delimiting the useful operation range (GET_GRAPH_K) and to provide flags stating the type of third-party performance certification of the of a specified fan (GET_CERT_DATA) have been included. This version is distributed together with Ventil Version 3.1.2, with the DLL technical documentation for SW developers, and as part of the redistributable archive package for ADH, RDH, RDA and AT fans.
- Version 2.5.3 was a purely development version, not distributed.
- Version 2.5.4 (292 KB, 20-06-05) was modified to improve the smoothness of the constant power level curves, and to correct a bug which had de-activated the control on the minimum number of parallel belts to be used when selecting belt drives with the functions GET_CALCULATION_BELT and GET_CALCULATION_MULTIBELT.

3. SYSTEM SPECIFICATIONS

3.1 Hardware/OS specifications:

- IBM-compatible PC, with Intel 486 DX4 100 MHz or higher processor
- Microsoft Windows 95, 98, Windows ME or Windows NT operating system.
- RAM min. 8 MB
- Free space on hard disk min. 18 MB to install the reduced set of fan ranges and versions (ADH+RDH+AT, from the redistributable, compressed package).
- Free space on hard disk min. 40 MB to install the complete set of fan ranges and versions (from Ventil 2.2.0 or later version).

3.2 Software specifications

The DLL was developed and tested as a Standard-32-bit DLL (Regular Static) according to Microsoft Convention for Windows 95/NT, which exports its functions with a C-compatible interface. All the functions of the DLL-file can be called with the command parameter "stdcall".

All internal calculations are made with the precision of the variable type double (8 bytes).

All the software functions don't produce any function value, and exchange return values through the calling list variables. They are C "Void functions" or Visual Basic and Fortran "Subroutines" or Pascal "Procedures". All through this document the word "function" will mean the same as "subroutine".

3.3 General conventions

A generic DLL function has the following interface structure:

```
DLL Export  Function name (
                int*                s1
                int*                s2
                double              IN      input array of double values
                char*               KEY     (max 30 Bytes)
                int*                z1
                int*                z2
                double              OUT     output array of double values
or
                user defined TYPE   xxx     variable as specified)
```

s1, s2, z1, z2 are integer*2 values, originally used for MS Excel 7.0 cells integration. Calling the DLL from programming languages (C, C++, Pascal, FORTRAN, Basic, etc.) these will usually be dummy variables. They have been kept only for back compatibility purposes. In the DLL there are no calculations of these values.

The last array is used as a buffer for the results. The number and content of the items in the input/output arrays depends on the individual function. In the field OUT, starting from item 1, the input data can be repeated, to show whether there were transmission mistakes, and then the output data listed. The variable OUT(0) usually contains a return error code, and is equal to NULL if the function call has been successful.

3.4 Product descriptors

The text variable KEY (pointer to char variable) is used to transfer the “type key”, which clearly identifies the particular fan. It consists of a string of max. 15 characters, composed by range, size and version descriptors, separated by spaces, like “ADN 225 L”.

Some functions use a RANGEKEY descriptor instead. This RANGEKEY descriptor is a character string, which identifies a range and version combination, like “AT-TIC”. The range and version descriptors are separated by a dash (-). The version descriptor itself may be composed, and contain one or more additional dashes.

A list of the available RANGEKEY values is located in the NIC_INI.ini configuration file, located together with the archive files and can be found in the output of the function GET_INI_CONFIG.

In the text fields, only the characters from ASCII(32) to ASCII(127) are allowed. The fields KEY and RANGEKEY are not case sensitive.

3.5 Character variables

All character variables must be C-styled, null terminated variables.

This is done automatically when the calling program is written with “C”. With other programming languages a NULL character (ASCII Zero) must be deliberately added after the last significant character, or the character variable must be padded with NULLs instead of zeros.

3.6 Software location

The DLL (i.e. the “Nicotra.DLL” file) should preferably be stored in the installation directory of the customer’s CAE program. The auxiliary dll files, Dformd.dll (418 KB, 17-11-98), Dforrt.dll (411 KB, 17-11-98) and Msvcrt.dll, when required, shall be located either in the system32 directory or, preferably, together with the nicotra.dll file. Care should be used to check that more recent versions of these files have not yet been installed with other applications. A first-order sub-directory, relative to the Nicotra.DLL location, shall be called “Nicotra_it” and shall contain every other file required, with a structure similar to the structure of the installation directories of the Ventil stand-alone program.

Any alternative location of the DLL files and of the archives requires the use of the SET_DLL_PATH function. This allows a position, of the Nicotra_it directory, different from the current working directory at the time of the first DLL function call, including remote location of the archive files on a networked drive, provided that the archives are still inside a directory called “Nicotra_it”.

3.7 Error codes

The DLL makes no direct dialogues with the end-user. The main program should use a special routine, to read the error description from the error code legend, corresponding to the error code returned, and to show it to the user in a dialogue box. An effort was made to trap major errors, to prevent a system crash.

An ASCII formatted file is supplied with this manual, and contains a legend of the error codes, or a list of numerical codes and their corresponding description in English language.

This file, called “FAILURE.TXT”, has a structure similar to the following example:

0; no error <CR><LF>	
1; file not found <CR><LF>	Error message for OUT(0) = 1
2; fan not found <CR><LF>	Error message for OUT(0) = 2
...	
etc.	(The messages are examples only).

4. FUNCTIONALITY OF THE SELECTION SUB-PROGRAMS

Functions to supply the following data are available from the Nicotra BDF DLL:

- Procedure for setting up the archive files search path.
- List of available fans included within the archives
- List of archives and of their version numbers
- List of available accessories for each selected fan range/size/version combination
- Nameplate/limit data for each selected fan
- Certification status of a specified fan
- Operating point calculation for each selected fan
- Multiple output, operating point calculation for selected fan ranges
- Extension of the noise data to include Lw values inside and outside a duct connected to the fan
- Belt drive design and life calculation for the selected fan and electric motor (electric motor data to be supplied at run time or reverting to default)
- As above, with multiple choice output
- Polynomial coefficients/coordinates used to plot the performance diagram for each selected fan
- Constant speed curves of the performance for each selected fan
- Bitmap-format picture of the selected fan
- EMF-format dimensional drawing of the selected fan

4.1 List of the available fan ranges and versions: **GET_INI_CONFIG**

Void Function (subroutine – procedure) name: GET_INI_CONFIG

DLLExport GET_INI_CONFIG	(int*	NSERIE	number of RANGEKEY combinations
	char*10	SERIE[0:99]	array of fixed length strings)

This function produces, on a command call, the number and list of available RANGEKEY values, read from the NIC_INI.ini configuration file.

SERIE is a 100 element array of fixed length (10 bytes) strings, each one containing a single RANGEKEY descriptor.

If the number of RANGEKEY values in the configuration file exceeds 100, only the first 100 lines are actually handled. This case doesn't happen with the available archive packages, but only during testing.

4.2 List of the available fans: **GET_PRODUCTS**

Void Function (subroutine – procedure) name: GET_PRODUCTS

DLLExport GET_PRODUCTS (char* LIST String with max. Nmax Bytes)

This function produces, on a command call, a list of products available to the DLL in the available archives.

The list will be returned as an ASCII string, with each individual product on a separate line, with lines separated by the <CR><LF> (ASCII(13) and ASCII(10)) characters.

The maximum length of the List character variable is Nmax bytes.

Up to version 1.0.0 Nmax has been equal to 20000 bytes. With the latest additions to the fan range covered, the output of this function marginally exceeds the 20000 bytes length and is cut short. As a stopgap solution, for users of the dll in association with an AHU design program, the output can be reduced to a useful length by editing the Nic_Ini.ini file to remove unused fan ranges (e.g. single inlet fans). See chapter 5 for details.

Starting from version 2.0.0 of the Nicotra DLL have this variable extended to 30000 characters.

Special versions of the dll file, 2.1.1 and 2.2.1, have been available with this variable reduced back to 20000 characters, to be compatible with existing software. As this backwards compatibility requirement seems not existing any longer, development of these 20000 Byte string versions has now ceased.

Contact the Nicotra R&D dept. at r&d@nicotra.it if you still have a need for such a backwards compatibility feature.

Each line of the output has the following format, with commas as field separators:

type-key,	code number of supplier,	description
(max. 15 Bytes)	(max. 20 Bytes)	(max. 80 characters)

The first row will contain the “NICOTRA” name.

Example:

NICOTRA
ADN 160 L,610000W,Fan with forward-curved wheel
RDN 180 R,612021W,Fan with backward-curved wheel

4.3 List of data about the archive version: **GET_ARC_VERSION**

Void Function (subroutine – procedure) name: GET_ARC_VERSION

DLLExport GET_ARC_VERSION (int* LG Language tag
char* List String with max 20000 characters)

The LG integer variable has to be set to the following:

=1	German Language
=2	English Language
=3	Spanish Language
=4	French Language
=5	Italian Language
=6	Custom Language (when available).

This will identify which version of the language-specific .ACS (accessory list) files must be searched for. Other archive files are not language dependent.

The list will be returned as an ASCII string, with data related to each product on a separate line, and lines separated by the <CR> <LF> (ASCII(13) and ASCII (10)) characters. Each line has the following format, with commas as field separators:

Rangekey, PRD <Prd file version>, ACS <Acs file version>, DAT <Dat file version>

(max. 15 Bytes)

The first row will contain the “NICOTRA” name and the DLL version Number.

Example:

NICOTRA v. 2.0.0

AT-S,PRD 1.0,ACS 1.1,DAT V. 1.0

AT-SC,PRD 1.0,ACS 1.1,DAT V. 1.0

4.4 List of available accessories for each selected fan: **GET_ACCESSORY**

Void Function (subroutine – procedure) name: GET_ACCESSORY

DLLExport GET_ACCESSORY (	char*	KEY	product descriptor
	int*	LG	language tag
	char*	LIST	String with max 20000 characters)

This function produces, for each fan identified by the variable KEY in accordance with the values supplied by the preceding function, a list of the available accessories:

The LG short integer variable has to be set to one of the following values:

- =1 German Language
- =2 English Language
- =3 Spanish Language
- =4 French Language
- =5 Italian Language
- =6 Custom Language (when available)

to get the accessories description in the desired language.

Each accessory takes just one line in the string-field. Rows will be separated by <CR><LF> (ASCII(13) and ASCII(10)).

Each row contains the following data, separated by commas (,):

type-key,	Nicotra code-number,	description,	Quantity
(max. 15 Bytes)	(max. 20 Bytes)	(max. 80 Bytes)	(max. 5 Bytes)

The quantity field shows how many pieces are required for each fan.

The first row will contain the “NICOTRA” name to identify the supplier.

Example:

NICOTRA

RDN 560 K2,R45065,Inlet guard,2

4.5 Nameplate/limit data for each selected fan: **GET_STANDARDS_FANALONE**

Void Function (subroutine – procedure) name: GET_STANDARDS_FANALONE

```
DLLExport GET_STANDARDS_FANALONE (char*     KEY          fan type descriptor
                                  int*      z1
                                  int*      z2
                                  double    OUT[ 0:13],  output field of doubles)
```

This function gives, for the fan indicated by the variable KEY, the nominal and operational limits data, for which no additional input is required.

The OUT array has the following content:

Dimension	Unit	Input/Output	Remarks
Error parameter		OUT(0)	
Max. air volume	m³/h	OUT(2)	
Max. total pressure	Pa	OUT(3)	
Max. speed	1/min	OUT(4)	
Max. shaft power	kW	OUT(5)	
Moment of inertia	kg m²	OUT(6)	
No. of blades		OUT(7)	
Weight	kg	OUT(8)	
Width	mm	OUT(9)	
Height	mm	OUT(10)	
Depth	mm	OUT(11)	
Shaft diameter	mm	OUT(12)	
Max. radial force at shaft end	N	OUT(13)	

4.6 Certification flags for each selected fan: **GET_CERT_DATA**

Void Function (subroutine – procedure) name: GET_CERT_DATA

This function provides, for the fan indicated by the variable KEY, information on the type of third-party certification applicable to the specified product.

The use of this function is reserved to Nicotra.

4.7 Operating point calculation for single selected fan: **GET_CALCULATION_FANALONE**

Void Function (subroutine – procedure) name: GET_CALCULATION_FANALONE

DLL Export GET_CALCULATION_FANALONE

(int*	s1	
int*	s2	
double	IN[0:12]	input array of double values
char*	KEY	fan type descriptor
int*	z1	
int*	z2	
double	OUT[0:31]	output array of double values)

This function returns, for the fan identified by the variable KEY, the operating point data.

The command may have different operational modes, according to the selection " Option " flag located in IN(0).

Size	**Unit**	**Input/Output**	**Remarks**
Numerical input data:			
Option Flag		IN(0)	Mode flag, selects calculation starting from: 1. Air volume and static pressure 2. Air volume and total pressure 3 to 9: reserved for future development see note1
Installation type		IN(1)	Installation type (see note2) 1. "A" - free inlet free outlet 2. "B" - free inlet ducted outlet 3. "C" - ducted inlet free outlet (not used) 4. "D" - ducted inlet ducted outlet (not used)
Air density	kg/m³	IN(2)	see note 3
Temperature	°C	IN(3)	see note 3
Height	m	IN(4)	for optional air density calculation (note 3)
Flow rate	m³/h	IN(5)	
Static pressure	Pa	IN(6)	see Option Flag
Total pressure	Pa	IN(7)	see Option Flag
Speed	1/min	IN(8)	Reserved for future use, leave 0
Shaft power	kW	IN(9)	Reserved for future use, leave 0
Efficiency		IN(10)	Reserved for future use, leave 0
Sound power level	dB	IN(11)	Reserved for future use, leave 0

Power correction factor	-	IN(12)	Between 1.00 and 2.00 , see note 4 also

Size	**Unit**	**Input/Output**	**Remarks**
Numerical output data:			
Error code		OUT(0)	
Installation type		OUT(1)	Installation type (see note2) 1. “A” - free inlet free outlet 2. “B” - free inlet ducted outlet 3. “C” - ducted inlet free outlet (not used) 4. “D” - ducted inlet ducted outlet (not used)
Air density	kg/m³	OUT(2)	see note 3
Temperature	°C	OUT(3)	see note 3
Height	m	OUT(4)	for optional air density calculation (note 3)
Flow rate	m³/h	OUT(5)	
Static pressure	Pa	OUT(6)	
Total pressure	Pa	OUT(7)	
Speed	1/min	OUT(8)	
Shaft power	KW	OUT(9)	
Efficiency		OUT(10)	
Sound power level	DB	OUT(11)	outlet side dBW(A)
Sound power -outlet side	DB	OUT(12)	linear, not filtered
Sound power -inlet side	DB	OUT(13)	linear, not filtered
Octave spectrum - outlet side	DB	OUT(14..21)	not filtered for frequencies 63,125, 250,500,1000,2000,4000,8000 Hz
Octave spectrum - inlet side	DB	OUT(22..29)	not filtered for frequencies 63,125, 250,500,1000,2000,4000,8000 Hz
Smallest required motor power	KW	OUT(30)	Fan shaft power x safety coefficient (according to catalogue), see note 4 also
Sound power level	DB	OUT(31)	inlet side dBW(A)
Error code		OUT(0)	

note1: option numbers from 3 to 9 represent other input data combinations and are reserved for future development.

note2: installation types are defined in accordance to ISO 5801; only installation types A and B are supported at the moment.

note3: when a non zero value for fluid density is supplied in IN(2), this value is used for fan performance calculation and location height IN(4) is discarded; when the input value for air density is zero, this value is calculated from the values of air temperature IN(3) and location height IN(4), and then returned in OUT(2); in both cases air temperature IN(3) is checked against the maximum allowable temperature limit.

note4: when IN(12) ≤ 0, the value is automatically calculated according to the standard catalogue recommendations, e.g. equal to 1.20 if fan shaft power ≤ 10 kW, equal to 1.15 if > 10 kW (may be different for different product ranges).

4.8 Operating point calculation for a version range: **GET_CALCULATION_MULTIFANS**

Void Function (subroutine – procedure) name: GET_CALCULATION_MULTIFANS

DLL Export GET_CALCULATION_MULTIFANS

(int*		s1	
int*		s2	
double		IN[0:12]	input array of double values
char*		RANGEKEY	fan range-version descriptor
int*		z1	
int*		z2	
double*	OUT		error code
user defined " Fans " TYPE		VENT[0:30]	output array of user defined types)

This function gives, for the fan product range-version identified by variable RANGEKEY, the operating point data for each fan within that range-version, in decreasing order of total efficiency. The alphanumeric variable RANGEKEY must contain a range-version descriptor as described in the paragraph "General conventions" (e.g. "AT-TIC").

The command may have different operational modes, according to the selection flag "Option".

Size	Unit	Input/Output	Remarks
Option Flag		IN(0)	Mode flag, selects calculation starting from: 1. Air volume and static pressure 2. Air volume and total pressure 3 to 9: reserved for future development see note1
Installation type		IN(1)	Installation type (see note2) 1. "A" – free inlet free outlet 2. "B" – free inlet ducted outlet 3. "C" – ducted inlet free outlet (not used) 4. "D" – ducted inlet ducted outlet (not used)

Air density	kg/m³	IN(2)	see note 3
Temperature	°C	IN(3)	see note 3
Height	m	IN(4)	for optional air density calculation (note 3)
Flow rate	m³/h	IN(5)	
Static pressure	Pa	IN(6)	see Option Flag
Total pressure	Pa	IN(7)	see Option Flag
Speed	1/min	IN(8)	Reserved for future use, leave 0
Shaft power	kW	IN(9)	Reserved for future use, leave 0
Efficiency		IN(10)	Reserved for future use, leave 0
Sound power level	dB	IN(11)	Reserved for future use, leave 0
Power correction factor	-	IN(12)	Between 1 and 2
Error code		OUT	
Output		VENT[0:30]	Array of Fans TYPE variables

note1: option numbers from 3 to 9 represent other input data combinations and are reserved for future development.

note2: installation types are defined in accordance to ISO 5801; only installation types A and B are supported at the moment.

note3: when a non-zero value for fluid density is supplied in IN(2), this value is used for fan performance calculation and location height IN(4) is discarded; when the input value for air density is zero, this value is calculated from the values of air temperature IN(3) and location height IN(4), and then returned in VENT.RO; in both cases air temperature IN(3) is checked against the maximum allowable temperature limit.

note4: when IN(12) ≤ 0, the value is automatically calculated according to the standard catalogue recommendations, e.g. equal to 1.20 if fan shaft power ≤ 10 kW, equal to 1.15 if > 10 kW (may be different for different product ranges).

The general definition of an element of a “Fans” TYPE element is: VENT(n).element_name. In the following list the composing elements are listed and described.

Fans TYPE elements description

Element	Units	Description	Type
Tipo	-	Fan type descriptor	String*18
wrn	-	Warning label (if not empty operational limits have been exceeded), see note 1	String*12
nrec	-	Record number in the archive	Long
Eta	-	Efficiency	Single
Pvol	m³/h	Air flow rate	Single
Ptot	Pa	Total pressure	Single
Pstat	Pa	Static Pressure	Single
Pdin	Pa	Velocity Pressure	Single
Ngiri	1/min	Fan speed	Single

w	kW	Fan Power	Single
Wel	kW	Minimum required motor power	Single
vper	m/s	Peripheral Velocity	Single
Vel	m/s	Air average Velocity at fan outlet	Single
Lwam	dB(A)	Sound Power Level, A-weighted – outlet side	Single
Lwaa	dB(A)	Sound Power Level, A-weighted – inlet side	Single
Ro	kg/m³	Air density	Single
NB	1/min	Reserved	Single
Ns	1/min	Reserved	Single
Nmax	1/min	Max Fan Speed allowed in calculations	Single
Wmax	kW	Max Fan Power allowed in calculations	Single
diam	mm	Diameter of the impeller	Single
miner	kgm²	Moment of inertia	Single
fl1	kW	Max operating fan power	Single
fl2	1/min	Max operating fan speed	Single
fl3	°C	Min operating fan temperature	Single
fl4	°C	Max operating fan temperature	Single
fl5	1/min	Min operating fan speed	Single
NG	-	Number of impellers	Single
Cw	-	Reserved	Single
CN	-	Reserved	Single
cps	-	Reserved	Single
SPLm	dB	Sound power level – linearly weighted – outlet side	Single
fr1m	dB	SPL outlet side - Octave 63 Hz	Single
fr2m	dB	SPL outlet side - 125 Hz	Single
fr3m	dB	SPL outlet side - 250 Hz	Single
fr4m	dB	SPL outlet side - 500 Hz	Single
fr5m	dB	SPL outlet side - 1000 Hz	Single
fr6m	dB	SPL outlet side - 2000 Hz	Single
fr7m	dB	SPL outlet side - 4000 Hz	Single
fr8m	dB	SPL outlet side - 8000 Hz	Single
SPLa	dB	Sound power level – linearly weighted – inlet side	Single
fr1a	dB	SPL inlet side - Octave 63 Hz	Single
fr2a	dB	SPL inlet side - 125 Hz	Single
fr3a	dB	SPL inlet side - 250 Hz	Single
fr4a	dB	SPL inlet side - 500 Hz	Single
fr5a	dB	SPL inlet side - 1000 Hz	Single

fr6a	dB	SPL inlet side - 2000 Hz	Single
fr7a	dB	SPL inlet side - 4000 Hz	Single
fr8a	dB	SPL inlet side - 8000 Hz	Single
esp	-	Bearing life calculation exponent	Single
Q	N	Bearing life calculation Reference Radial Load	Single

N.B.

Single = Float = IEEE 4 bytes floating-point number

Long = Integer*4 = IEEE 4 byte Integer

String = Character

note 1: the warning character string wrn is set to 'OUT OF LIMIT' when a generic operating limit is exceeded in the required operating point; with some new products, an additional check of the operating point location has been added: the warning string can be set to 'TOO LEFT' when the required operating point is located to the left of the normal operating range, and to 'TOO RIGHT' when the right margin of the normal operating range is exceeded.

4.9 Extension of the basic fan data to a complete set: **GET_NOISE_DATA**

Void Function (subroutine – procedure) name: GET_NOISE_DATA

DLLExport GET_NOISE_DATA (

single*	INLET_DATA_OUT[0:9]	inlet noise data out of the duct
single*	INLET_DATA_IN[0:9]	inlet noise data inside the duct
single*	OUTLET_DATA_OUT[0:9]	outlet noise data out of the duct
single*	OUTLET_DATA_IN[0:9]	outlet noise data inside the duct
single*	BO_DATA[0:9]	break-out noise data
int*	FLAG_INSTALLAZIONE	installation type flag
int*	ERROR	error flag
user defined “Fans“ TYPE	VENT	single "Fans" structure)

This function gives, for the fan duty described in detail in the structure VENT, as calculated by the function GET_CALCULATION_MULTIFAN, a complete set of noise ratings inside and outside of a duct connected with the fan inlet and/or outlet. , and under The applicable installation category must be specified using the variable FLAG_INSTALLAZIONE.

INLET_DATA_OUT	noise data outside of the inlet bellmouth / inlet duct end
INLET_DATA_IN	noise data inside an inlet duct
OUTLET_DATA_OUT	noise data outside of an outlet duct
OUTLET_DATA_IN	noise data inside an outlet duct
BO_DATA	break-out noise data

So far, noise data inside an inlet duct (INLET_DATA_IN) and break-out noise data (BO_DATA), which are used for C and D type installation categories, are not supported and always left to 0. The array for noise data inside an outlet duct is filled out only for installation category "B".

The integer variable FLAG_INSTALLAZIONE must be either set to 0, for installation category "A", "free inlet, free outlet", or to 2, for installation category "B", "free inlet, ducted outlet".

Inside each of the five ten-element arrays, elements 0 to 7 are octave band sound power levels for the eight standard octave bands centred on 63, 125, 250, 500, 1000, 2000, 4000 and 8000 Hz. Element no. 8 is the (Linear) total sound power level Lw, and element no. 9 is the A-weighted total sound power level LwA.

4.10 Polynomial curves for performance diagram of the selected fan

These four functions allow plotting the performance diagram of the selected fan in a plane having Total Pressure in Pa as ordinate and Flow Rate in m³/h as abscissa, at the reference density of 1,20 kg/m³. Separate commands return the 5th degree polynomials representing the curves of each one of the following four families.

Set of curves	**Function name**	**Unit**
Curves of constant power P	GET_GRAPH_POWER	kW
Curves of constant speed n	GET_GRAPH_RPM	1/min
Curves of constant efficiency eta	GET_GRAPH_ETA	dimensionless
Curves of constant sound power Lw	GET_GRAPH_SOUND	(see note)

Each curve has the analytical form:

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5$$

where

y is the total pressure Δp_{total}[Pa] and

x is the flow rate Q[m^3/h], comprised between Vmin e Vmax.

Each command has the format

DLLExport GET_GRAPH_XXXXX

char*	KEY	
unsigned short int*	z1	
unsigned short int*	z2	
double	OUT[0:136],	output array of 137 double values

A max. of 15 curves, each one extending between the minimum and maximum values, will be calculated.

Each curve is defined by the value of the constant parameter (power, sound power level, speed, efficiency), the coefficient $a_0...a_5$, and the curve extension, defined by values of minimum and maximum air volume, (Vmin, Vmax). These data will be stored sequentially in successive fields as follows:

OUT(0) Error message

OUT(1) Number of constant values n

OUT[2], OUT[3].... OUT[1+nx9] =

{value1, a_0(value1), a_1(value1), a_2(value1), a_3(value1), a_4(value1), a_5(value1), Vmin(value1), Vmax(value1),

value2, a_0(value2), a_1(value2), a_2(value2), a_3(value2), a_4(value2), a_5(value2), Vmin(value2), Vmax(value2),

value3, a_0(value3), a_1(value3), a_2(value3), a_3(value3), a_4(value3), a_5(value3), Vmin(value3), Vmax(value3),

.....

value_n, a_0(value_n), a_1(value_n), a_2(value_n), a_3(value_n), a_4(value_n), a_5(value_n), Vmin(value_n), Vmax(value_n)}

NOTE FOR Constant Sound Power Lw: GET_GRAPH_SOUND

This function should employed together with the following PRESS_DB_CONST procedure. It's used to provide the number of constant sound power level curves, n, in OUT(1), the constant SWL step value of each curve, in value_n, and the minimum and maximum air flow [Vmin(value_n), Vmax(value_n)] for each SWL value, as described above , to be used as inputs in the PRESS_DB_CONST function.

The polynomial coefficients returned by GET_GRAPH_SOUND are useless and meaningless.

4.11 Constant SWL curves of the selected fan: **PRESS_DB_CONST**

This procedure allows plotting the constant sound-power-level curves on the performance diagram of the selected fan (in a plane having Total Pressure in Pa as ordinate and Flow Rate in m³/h as abscissa, at the reference density of 1,20 kg/m³). It returns, for the selected fan and sound power level LW, the pressure vs air volume curve, point by point.

The procedure has the format

DLLExport PRESS_DB_CONST

(char*	KEY	
float*	DB	Input: Sound power LW (dB)
float*	VMN	Input: Min air volume (m³/h)
float*	VMX	Input: Max air volume (m³/h)
float*	PY[0:100]	output: array of single values (Pa)
float*	VX[0:100]	output: array of single values (m³/h)
int*	Points	Input: required number of points (max 100)
int*	Erl	Output: Error flag)

It returns two single-index arrays containing the required curve point by point (index 1 is the lower air volume end of the curve, index = Points is the higher air volume end).

Vector elements with index = 0 are not point coordinates: VX(0) is the required SWL value, PY(0) is the actual number of points calculated along the constant SWL curve, equal to Points when the curve is not cut through the maximum or minimum speed lines.

The actual number of calculated points can be lower than the required one when the constant SWL curve is cut across the maximum or minimum constant speed curves. A minimum number of 15 points is calculated.

4.12 Constant SWL curves of the selected fan: **POINTS_DB_CONST**

This is an extended version of the above function providing the constant sound-power-level curves on the performance diagram of the selected fan (in a plane having Total Pressure in Pa as ordinate and Flow Rate in m³/h as abscissa, at the reference density of 1.20 kg/m³). In the extended version, fan installation type, side and air density can be specified, and it returns, for the selected fan and sound power level LW, the pressure vs. air volume curve, point by point. Its use is recommended over the simpler and older version, which might be discontinued in the future.

The procedure has the format

DLLExport POINTS_DB_CONST

(char*	KEY	
float*	DB	Input: Sound power LW (dB)
float*	VMN	Input: Min air volume (m³/h)
float*	VMX	Input: Max air volume (m³/h)
float*	PY[0:100]	Output: array of single values (Pa)
float*	VX[0:100]	Output: array of single values (m³/h)
int*	POINTS	Input: required number of points (max 100)
int*	INSTALLATION_FLAG	input: installation type flag
int*	SIDE_FLAG	input: fan side flag
float*	RO	input: Standard density value
int*	ERL	Output: Error flag)

INSTALLATION_FLAG is an integer variable used to select the correct installation setup. A value of 0 (zero) means installation type A (free inlet – free outlet) while a value of 2 means installation type B (free inlet – ducted outlet).

SIDE_FLAG is an integer variable used to choose the target side for the sound power data, returned by the POINTS_DB_CONST dll routine. If SIDE_FLAG is set to 1, the sound power levels returned by the

POINTS_DB_CONST dll routines apply to the inlet side, if SIDE_FLAG is set to 2, the sound power levels returned by the dll routine apply to the outlet side.

VMN and VMX are minimum and maximum volume flow values (in m^3/h) for the constant SWL line to be calculated. If left to zero, the subroutine will calculate the line going from the left to the right extreme edges of the catalogue performance range for the specified fan.

RO is the input density value in kg/m^3, normally 1.20. The subroutine does not provide default values: this variable must either be set to the standard value of 1.20 or to the actual density value.

The POINTS_DB_CONST routines returns two single-index arrays containing the required curve point by point (index 1 is the lower air volume end of the curve, index = Points is the higher air volume end).

Array elements with index = 0 are not point coordinates: VX(0) is the required SWL value, PY(0) is the actual number of points calculated along the constant SWL curve, equal to Points when the curve is not cut through the maximum or minimum speed lines. The actual number of calculated points can be lower than the required one when the constant SWL curve is cut across the maximum or minimum constant speed curves. A minimum number of 15 points is calculated.

4.13 Parabolic constants defining the normal operation area: **GET_GRAPH_K**

This procedure provides the information to plot the left (high) and right (low) parabolas limiting the recommended operation area, in the total pressure vs. volume flow rate diagram, for a specific fan, identified by the descriptor KEY, working at an air density ROCALC.

DLLExport GET_GRAPH_K

(char*	KEY	Input: Product descriptor
float*	ROCALC	Input: Air density [kg/m^3]
float*	KPAR1	Output: Parabolic constant of left-side limit [$Pa/(m^3/s)^2$]
float*	KPAR2	Output: Parabolic constant of right-side limit [$Pa/(m^3/s)^2$]
int*	ERROR	Output: Error Flag)

The fan recommended operation area is between

$$P_{t-Max} = KPAR1 \cdot Q^2$$

and

$$P_{t-\min} = KPAR2 \cdot Q^2$$

where $P_{t\text{-}Max}$ and $P_{t\text{-}min}$ are total pressure values in Pa, and Q is volume flow rate in m^3/s.

Operation outside of this range may still be feasible, according to the fan speed and the level of structural loads on the fan, but the use of a different fan size, or the change between an FC fan and a BC one, may be preferable and provide more stable operation.

4.14 Constant speed performance curves through design operating point: **GET_PCURVES**

Void Function (subroutine – procedure) name: GET_PCURVES

DLL Export GET_PCURVES

(user defined " Fans " TYPE	VENT	input structure
float*	TABLE(1:7,1:20)	output array
short int*	ERC	error code/o.p. index)

This function accepts as input a single structure of the "Fans" type, like the elements of the output array given by the GET_CALCULATION_MULTIFANS. This particular user-defined type is described in detail in chapter 4.7, and identifies a particular fan and the parameters of its operating point. Any output structure given by GET_CALCULATION_MULTIFANS is an appropriate input for the GET_PCURVES function.

The output array of single precision, floating point variables contains values of Static Pressure in Pa, Dynamic Pressure in Pa, Total Pressure in Pa, Shaft Power in kW, Total Efficiency as percentage and Sound Power Level in dBW(A), all as functions of Volume Flow rate given in m³/h. This allows plotting of the constant speed performance curves of the fan, at the speed required to achieve the selected duy.

TABLE(1, 1:20) = Volume flow [m³/h]
TABLE(2, 1:20) = Static Pressure [Pa]
TABLE(3, 1:20) = Dynamic Pressure in Pa
TABLE(4, 1:20) = Total Pressure in Pa
TABLE(5, 1:20) = Shaft Power in kW
TABLE(6, 1:20) = Total Efficiency [%]
TABLE(7, 1:20) = Sound Power Level [dBW(A)]

The short integer output variable ERC is either an error code (≥ 300) or the index (between 1 and 20) of the original operating point in the sequence of volume flow values. This identifies the co-ordinates of an operating point marker in the diagram.

When both inlet and outlet sound power values are available, outlet side values are listed in the table, otherwise, the only available Lw values are used.

The error code variable ERC may be set to either -1 or -2 in input, to specify that Inlet side sound power levels or outlet side sound power levels must be used, respectively, and provided that both are available. Other input values, including 0, are ignored.

The value of ERC is always overwritten with the index showing the location of the selection point in the output table.

4.15 Bitmap-format picture of the selected fan: **GET_PICTURE**

Void Function (subroutine – procedure) name: GET_PICTURE

DLL Export GET_PICTURE

(char* KEY		input	
HBITMAP	handle to Bitmap	output)	

This function returns a picture of the selected fan. This is sized to be shown inside a picture box with size 200(v) x 300(h) pixel.

The picture is supplied as a Bitmap by the function GET_PICTURE. The function will load the bitmap in memory and supply directly the "handle" to the required Bitmap. When the resultant "handle" is null, the picture is not available.

4.16 Bitmap-format picture filename of the selected fan: **GET_PICTURE_NAME**

Void Function (subroutine – procedure) name: GET_PICTURE_NAME

DLL Export GET_PICTURE_NAME

(KEY	char*	fan type descriptor	input
File_Name	char*)	Bitmap File Name (max 255 char)	output

This function returns a picture of the selected fan. This is sized to be shown inside a picture box with size 200(v) x 300(h) pixel.

The picture is supplied as a Bitmap-type (.bmp) file, whose complete path\filename.bmp is returned by the function GET_PICTURE_NAME.

This function with a filename output was added to simplify work with programming languages like VB, which do not allow easy representation of a bitmap through its handle.

4.17 EMF-format dimensional drawing of the selected fan: **GET_WMF**

Void Function (subroutine – procedure) name: GET_WMF

DLLExport GET_WMF

(KEY	char*	fan type descriptor	input
hMF)	HMETAFILE	Handle to metafile	output

This function supplies a drawing of the selected fan in scaleable vector graphics format (black/white) in the form of a Windows Enhanced-Metafile (.emf format). This function allows inserting a reasonably detailed view of the fan, including dimensions, in a printout of the technical data. The drawings show all the essential dimensions with reference to an included table, showing relevant numerical values in mm.

The return variable is the “handle” to the drawing Enhanced-Metafile. When the values of the "handle" are zero, the drawing is not available. The drawings show all necessary indications for:

dimensions
fixing points
holes
connection dimensions.

4.18 EMF-format dimensional drawing of the selected fan: **GET_WMF_DIM_LIST**

Void Function (subroutine – procedure) name: GET_WMF_DIM_LIST

DLLExport GET_WMF_DIM_LIST

(char*	KEY	fan type descriptor
HMETAFILE	hMF	Handle to metafile
char*	FileName	Enh-metafile complete path (max 255 char)
int*	sort	dimensions sort flag
char*	Letter[0:27]	output array of 2 bytes chars
double*	Dimen[0:27])	output array of doubles

This function supplies a drawing of the selected fan in scaleable vector graphics format (black/white) in the form of a Windows Enhanced-Metafile.

This additional function differs from the preceding one because a complete filename is offered in output as an alternative to the Metafile Handle for those programming languages which cannot easily use the handle.

Fan dimensions are supplied separately from the drawing as an array of two character strings containing the reference letters used in the drawing, and an array of doubles with the corresponding dimensions in mm.

There is the option to re-order these two arrays according to an auxiliary text file containing the list of the vector element numbers in the new required order, one on each line. The text file must have the same name of the corresponding range-version archive and extension .SRT, e.g. “ADN-R.SRT”, and be located together with the archive files in the Nicotra_it directory. This option allows locating the same dimensions in fixed positions within the array, independently from the choices of reference letters of each particular drawing. This is useful to automatically get dimensional details like fixing holes distance along and across the shaft direction, shaft height and so on.

Dimensions re-ordering must be activated setting the variable “sort” to a value higher than zero.

If the metafile cannot be found, or if the sort option is activated and the appropriate .SRT file cannot be found, the handle is set to zero.

4.19 Belt drive design and life calculation: single optimum design - **GET_CALCULATION_BELT**

Void Function (subroutine – procedure) name: GET_CALCULATION_BELT

This command allows an automated calculation of the optimal belt drive, giving the minimum bearing life required and satisfying the following constraints:

1. Smallest possible number of belts, compatible with the required power
2. Smallest pulley diameters.

Calculations are based on belt data archives from the Optibelt™ catalog, while sizes of pulleys and bushes are taken from the Fenner™ and Brook-Hansen™ catalogues.

DLL Export GET_CALCULATION_BELT

(int*	s1	
int*	s2	
double	IN[0:23]	input array of doubles
char*	KEY	fan type descriptor
char*	BELT	(max. 30 Bytes)
char*	ERRORMESSAGE	(max 80 Bytes)
int*	z1	
int*	z2	
double	OUT[0:19]	output array of doubles)

This command may have different operational modes, according to specific values in the inputs.

Variable	Unit	Input/Output	Remarks
Fan type		KEY	Fan type descriptor
Belt drive type		BELT	Required belt section and groove number range with format: belt_section min_gr._nr,Max_gr. nr Example: SPA 2,3 (note 1)
Numerical input data:			
Operation mode		IN(0)	Allowable values are: 1, 2 (default), 3. See note 3.
Required distance between shaft centres	mm	IN(1)	If IN(1)=0 the lowest admissible distance is used (default)
Minimum belt drive life	h	IN(2)	If =0 default value of 40 000 h is used
Fan shaft required speed	1/min	IN(3)	Always > 0
Fan shaft required power	kW	IN(4)	See note 2
Motor supply voltage	V	IN(5)	Not used
Motor supply frequency	Hz	IN(6)	If IN(6)=0 50 Hz is used (default)
Motor pole number		IN(7)	Always > 0
Motor nominal speed	1/min	IN(8)	See note 2
Motor nominal power	kW	IN(9)	See note 2
Motor nominal speed (lower speed of two speed motors)	1/min	IN(10)	Not used
Motor nominal power (lower speed of two speed motors)	kW	IN(11)	Not used
Motor shaft diameter	mm	IN(12)	See note 2
Belt drive design service factor		IN(13)	when left empty reverts to default 1.3
Motor shaft permissible radial load	N	IN(14)	See note 2
Motor bearing life under permissible radial load	h	IN(15)	See note 2
Exponent for motor bearing life calculation		IN(16)	3 for ball bearings, 3.33 for roller bearings; see note 2
Motor nominal torque	Nm	IN(17)	unused
Motor starting torque	Nm	IN(18)	unused
Operating temperature	°C	IN(19)	
Max allowed rpm change	1/min	IN(20)	See note 2
Motor pulley diameter	mm	IN(21)	See note 3
Fan pulley diameter	mm	IN(22)	See note 3
Number of grooves	-	IN(23)	Not used

Note 1: min_gr._nr is the minimum allowable number of belts
Max_gr._nr is the maximum allowable number of belts

The example shown requires the selection of an SPA section belt drive with a minimum of 2 and a maximum of 3 belts.

Note 2: if IN(9) and IN(8) = 0 and IN(4), IN(7) and IN(20) >0 then the speed and power of the motor will be automatically selected from a standard motor catalogue; also IN(12), IN(13), IN(15), IN(16) will be taken from such catalogue.

if IN(9) and IN(8) > 0 then all motor data in input are required; IN(20) is fixed by default in 100 1/min regardless the input value.

Note 3: if IN(0) = 1 this command will select pulley sizes only among pulleys available with conical bushes. If IN(0) = 2 the code will evaluate also pulleys without bushes.

If IN(0) = 3 the code will calculate the bearing life starting from the pulley dimensions supplied as input data in IN(21), IN(22) and in IN(23).

Numerical output data:			
Error code		OUT(0)	
Actual achieved fan speed	1/min	OUT(1)	Calculated fan speed
Number of belts		OUT(2)	
Diameter of fan pulley	mm	OUT(3)	
Conical bush code (fan)		OUT(4)	Taperlock™ size code
Fan shaft diameter	mm	OUT(5)	
Diameter of motor pulley	mm	OUT(6)	
Conical bush code (motor)		OUT(7)	Taperlock™ size code
Motor shaft diameter	mm	OUT(8)	
Motor bearing life in operating conditions	h	OUT(9)	
Belt length	mm	OUT(10)	
Calculated distance between shaft centres	mm	OUT(11)	
Maximum transmissible power	kW	OUT(12)	
Belt speed	m/s	OUT(13)	
Radial load on rotating shaft	N	OUT(14)	
Static belt tension	N	OUT(15)	
Radial load on shaft when stationary	N	OUT(16)	
Test force	N	OUT(17)	
Belt deflection under test load	mm	OUT(18)	
Fan bearing life in Operating conditions	h	OUT(19)	

4.20 Belt drive design and life calculation: multiple choice - **GET_CALCULATION_MULTI_BELT**

Void Function (subroutine – procedure) name: GET_CALCULATION_MULTI_BELT

This command allows an automated calculation of belt drives to achieve a minimum bearing life, giving a multiple output, which allows relative evaluation of the possible alternative designs. Output is ordered to have:

1. increasing number of belts (within the allowed range) and
2. increasing diameters.

Calculations are based on belt data archives from the Optibelt™ catalog, while sizes of pulleys and bushes are taken from the Fenner™ and Brook-Hansen™ catalogues.

DLL Export GET_CALCULATION_MULTI_BELT

(int*	s1	
int*	s2	
double	IN[0:23]	input array of doubles
char*	KEY	fan type descriptor
char*	BELT	(max. 30 Bytes)
char*	ERRORMESSAGE	(max 80 Bytes)
int*	z1	
int*	z2	
double*	OUT	error code
user defined “Trasmiss “ TYPE	TRAS[0:30]	output array of user defined types)

This command may have different operational modes, according to specific values in the inputs.

Variable	**Unit**	**Input/Output**	**Remarks**
Fan type		KEY	Fan type descriptor
Belt drive type		BELT	Required belt section and groove number range with format: belt_section min_gr._nr,Max_gr. nr Example: SPA 2,3 (note 1)
Numerical input data:			
Operation mode		IN(0)	Allowable values are: 1, 2 (default), 3. See note 3.
Required distance between shaft centres	mm	IN(1)	If IN(1)=0 the smallest possible distance is used
Minimum belt drive life	h	IN(2)	If =0 default value of 40 000 h is used
Fan shaft required speed	1/min	IN(3)	Always > 0

Fan shaft required power	kW	IN(4)	See note 2
Motor supply voltage	V	IN(5)	Not used
Motor supply frequency	Hz	IN(6)	If IN(6)=0 50 Hz is used (default)
Motor pole number		IN(7)	Always > 0
Motor nominal speed	1/min	IN(8)	See note 2
Motor nominal power	kW	IN(9)	See note 2
Motor nominal speed (lower speed of two speed motors)	1/min	IN(10)	Not used
Motor nominal power (lower speed of two speed motors)	kW	IN(11)	Not used
Motor shaft diameter	mm	IN(12)	See note 2
Belt drive design service factor		IN(13)	When left empty reverts to default 1.3
Motor shaft permissible radial load	N	IN(14)	See note 2
Motor bearing life under permissible radial load	h	IN(15)	See note 2
Exponent for motor bearing life calculation		IN(16)	3 for ball bearings, 3.33 for roller bearings; see note 2
Motor nominal torque	Nm	IN(17)	Unused
Motor starting torque	Nm	IN(18)	Unused
Operating temperature	°C	IN(19)	
Max allowed rpm change	1/min	IN(20)	See note 2
Motor pulley diameter	mm	IN(21)	See note 3
Fan pulley diameter	mm	IN(22)	See note 3
Number of grooves	-	IN(23)	Not used
Error code		OUT	
OUTPUT:	-	TRAS[0:30]	Array of Trasmiss TYPE variables

Note 1: min_gr._nr is the minimum allowable number of belts
Max_gr._nr is the maximum allowable number of belts

The example shown requires the selection of an SPA section belt drive with a minimum of 2 and a maximum of 3 belts.

Note 2: if IN(8) and IN(9) = 0 and IN(4), IN(7) and IN(20) >0 then the speed and power of the motor will be automatically selected from a standard motor catalogue; also IN(12), IN(13), IN(15), IN(16) will be taken from such catalogue.

if IN(8) and IN(9) > 0 then all motor data in input are required; IN(20) is fixed by default in 100 1/min regardless the input value.

Note 3 if IN(0) = 1 this command will select pulley sizes only among pulleys available with conical bushes. If IN(0) = 2 the code will evaluate also pulleys without bushes.

If IN(0) = 3 the code will calculate the bearing life starting from the pulley dimensions supplied as input data.

The general definition of a TYPE element is: TRAS.element_name. In the following list the composing elements are listed and described.

Trasmiss TYPE elements description

Element	Units	Description	Type
Pm	-	Motor poles number	Long
Ngmot	1/min	Motor speed	Long
Ngole	-	Number of grooves	Long
Dmot	mm	Motor pulley diameter	Long
Dven	mm	Fan pulley diameter	Long
NgventN	1/min	Achieved fan speed	Long
Ngvent	1/min	Required fan speed	Single
Wa	kW	Maximum transmissible power	Single
Wr	kW	Minimum required motor power	Single
Wass	kW	Fan Power	Single
Wmot	kW	Motor Power	Single
Heff	h	Fan bearing life in operating conditions	Single
Qn	kN	Radial load on rotating shaft	Single
Ieff	mm	Actual distance between shaft centres	Long
L	mm	Belt length	Long
Gamma		Smaller pulley belt contact angle	Long
Cl		Belt length correction coefficient	Single
Cg		Belt contact angle correction coefficient	Single
U	m/s	Belt speed	Single
Cbven	-	Conical bush code (fan): Taperlock™ size code	Long
Cbmot	-	Conical bush code (motor): Taperlock™ size code	Long
Damot	mm	Motor shaft diameter	Long
Daven	mm	Fan shaft diameter	Long
Heffm	h	Motor bearing life in operating conditions	Single
Ts	kN	Static belt tension	Single
Depth	mm	Belt yield under test load	Single
Fn	kN	Test force	Single
Qst	kN	Radial load on shaft when stationary	Single
CradMot	N	Motor shaft reference radial load	Single
Hmot	h	Motor bearing life under reference radial load	Single
EspMot	-	Exponent for motor bearing life calculation	Single

N.B. :

Single = Float = IEEE 4 bytes real number

Long = Integer*4 = IEEE 4 byte Integer

String = Character

4.21 Setting up the archive search path - **SETDLLPATH**

Void Function (subroutine – procedure) name: **SETDLLPATH**

This command allows setting the search path for the archive files, when required, allowing simplified use in a multi-user LAN environment.

DllExport SETDLLPATH (char* DLLPath)

The DLLPath argument is the address (pointer) of a NULL terminated string, 260 characters long, used to store the path to the “Nicotra_it” folder, which will contain all the archive files required by the DLL to operate.

Every call to the SETDLLPATH routine, with a valid path in the DLLPath string, sets the search path for the archive files to the child subdirectory DLLPath + “\Nicotra_it\”.

When the SETDLLPATH is called with a valid string argument, the value of the DLLPath string will be left unchanged and will contain the path to the *parent* directory of the Nicotra_it folder.

If any other function of the Nicotra.DLL file is called *before* calling the SETDLLPATH routine, the search path to the archive files will be built, by default, starting from the Current Working Directory (Current Working Directory + “\Nicotra_it\”), at the time of the first function call. This path will be stored and kept unchanged until the DLL session is closed, or the SETDLLPATH is deliberately used to change it.

The SETDLLPATH routine can also be called with a NULL string as input, in two different cases:

- if neither the SETDLLPATH routine, nor any other routine of the dll, has ever been called since the DLL upload time, a call with a Null argument string will build and store the path to the archive files, starting, by default, from the Current Working Directory and adding the child directory name (Current Working Directory + “\Nicotra_it\”),
- if the SETDLLPATH routine has already been used to explicitly set the path, or if this has already been built, by default, at the first call of a different routine from Nicotra.DLL, the path already stored is kept unchanged.

When SETDLLPATH is called with a NULL string argument, the routine will change the DLLpath string value into the new searching path, *including* the “\Nicotra_it\” child directory name. This is essentially an enquiry for the currently set path and can be used for troubleshooting.

Example:

If the Current Working Directory is: “C:\Programs\Executable file\” and the Nicotra_it folder is located inside the “C:\Archive data\” directory, there can be three cases:

- If the SETDLLPATH is called with the string argument “C:\Archive data\”, the search path to the archive files will be set to: “C:\Archive data\Nicotra_it\” and the input value “C:\Archive data\” will be left unchanged.
- If the SETDLLPATH routine is called, for the first time, with a NULL string argument, the search path to the archive files will be set by default to: “C:\Programs\Executable file\Nicotra_it” and the same string will be written inside the DLLPath variable.
- If we call the SETDLLPATH routine with a NULL string argument, but the routine has been previously called with the string argument “C:\Archive data”, the existing search path to the archive files will be returned inside the DLLPath variable as: “C:\Archive data\Nicotra_it”.

5. ARCHIVE SCOPE AND COVERAGE

The archives annexed to Ventil version 3.1.2 cover performance of the ADH, RDH, RDA, ASH, RSH, AS, AT and PFN1 product ranges, in their different single, twin and triple production versions, in accordance with catalogues “I-2 01/04”, “D-3 03/05”, “Z-1 02/05”, “W-1 10/04”, “B-2 10/00”, “A-7 11/03” , “P-3 05/03”.

Ventil 3.1.2 includes also performance ratings for three Inch-sized product ranges, ATU, ADHU and RDHU, according to catalogues ATU 07-2004, ADHU 07-2004 and RDHU 08-2004.

ADH, RDH, RDA and AT archives can be distributed also as a special, compressed package, intended to assist software developers who have to integrate DLL files and archives with an AHU calculation package.

A second, separate, compressed package, contains the DLL files and the archive files for ATU-ADHU-RDHU and RDAU inch-sized ranges.

ADN and RDN archives according to catalogue “C-6 03/00” have been included with Ventil up to version 2.0.0. They have been discontinued, as these products are now entirely superseded by ADH/RDH.

ADZ and RDZ archives, according to catalogue “L-3 10/94”, were distributed, on some particular markets, with Ventil version 1.0.2 and 1.0.3. They are not distributed any more, as these products are entirely superseded by ADH/RDH.

ASZ and RSZ archives, according to catalogue “M-3 05/99”, have been included with Ventil up to version 2.1.0. Again, they have been discontinued, as these products are now entirely superseded by ASH/RSH.

The traditional catalogues remain, at the moment, the official source for product performance figures.

Please look at any “readme.txt” file included with SW for any last minute archive update.

Important Notice:

Those customers who wish to restrict the scope of the selection functions, to select fans just from a part of the available ranges, can amend the file NIC_INI.INI, deleting those lines referencing to undesired fan range/version combinations.

Archive files should never be deleted *before* amending the NIC_INI.INI file, to avoid software malfunctions and error messages due to the unavailability of expected files.

6. UPDATE HISTORY

Rev. 0: 14/02/2001

The unsigned short integer type dummy variables have been rectified to integer*2 type, to allow a correct exchange. As these variables are not actually used by the dll, this doesn't produce compatibility problems with main programs written for the earlier versions.

Rev. 1: 12/04/2001

Added GET_PICTURE_NAME function.

Removed optional variable from function GET_PICTURE.

Rev. 2: 03/07/2001

Changed List length to 30000 bytes in GET_PRODUCTS.

Added DLL version number in the output of GET_ARCHIVE_VERSION.

Rev. 3: 09/07/2001

Reverted to standard List length of 20000 bytes in GET_PRODUCTS.

Totally revised text and structure.

Added function GET_INI_CONFIG.

Rev. 4: 16/07/2001

Changed again List length to 30000 bytes in GET_PRODUCTS.

Defined output from GET_INI_CONFIG function.

Distributed ver. 2.0.0, not compatible with Win95.

Rev. 5: 27/11/2001

Distributed ver. 2.1.0 with Ventil 2.0.0 on CD-Rom.

Introduced 20000 bytes GET_PRODUCTS string on backward-compatible version 2.1.1.

Listed different versions of the DLL v. 2 and their differences.

Clarified DLL file location.

Added description of the GET_PCURVES function.

Rev. 6: 28/01/2002
Updated introducing dll versions 2.2.0 and 2.2.1.

Rev. 7: 20/06/2002
Updated introducing dll version 2.3.0 and function SETDLLPATH.

Rev. 8: 10/10/2002
Updated introducing dll version 2.3.1 and various minor corrections.

Corrected note 4 in paragraphs 4.6 and 4.7.

Rev. 9: 03/02/2003
Updated introducing dll version 2.3.2.

Corrected bug in function GET_PICTURE_NAME.

Rev. 10: 02/08/2004
Updated introducing dll version 2.4.0.

Added description of the POINTS_DB_CONST function.

Rev. 11: 03/05/2005
Updated introducing dll version 2.5.2.

Added description of the function GET_NOISE_DATA.

Added description of the function GET_CERT_DATA.

Added description of the function GET_GRAPH_K.

Rev. 12: 13/10/2005
Revised description of the function GET_GRAPH_K.

Rev. 13: 08/03/2006
Described revised DLL versions 2.5.3 and 2.5.4 .