

Becker & Hickl GmbH  
Nahmitzer Damm 30  
12277 Berlin  
Tel. +49 30 787 56 32  
FAX +49 30 787 57 34  
email: [info@becker-hickl.de](mailto:info@becker-hickl.de)  
<http://www.becker-hickl.de>

DEL\_DLL.DOC



# **DEL-150/350 Dynamic Link Libraries**

## **User Manual**

Version 2.0

March 2003

## Introduction

The DEL 32 bit Dynamic Link Library contains all functions to control the DEL-150(350) ps Delay Generator modules. The functions work under Windows 9x/ME/NT/2000/XP. Up to four DEL modules can be controlled using the DEL DLL. The program which calls the DLLs must be compiled with the compiler option 'Structure Alignment' set to '1 Byte'.

The distribution disks contain the following files:

DELDLL32.DLL	dynamic link library main file
DELDLL32.LIB	import library file for Microsoft Visual C/C++, Borland C/C++, Watcom C/C++ and Symantec C/C++ compilers
DEL_DEF.H	Include file containing type definitions, functions prototypes and pre-processor statements
DEL150.INI	DEL DLL initialisation file
DEL_DLL.DOC	This description file
USE_DEL.XXX	Set of files to compile and run a simple example of using the DEL DLL functions. The source file of the example is the file use_del.c. The example was originally prepared under Borland C/C++, V.4.52. For other compilers, please choose the correct import library file to link.

There is no special installation procedure required. Simply execute the setup program from the 1st distribution diskette and follow its instructions.

## **DEL-DLL Functions list**

The following functions are implemented in the DEL-DLL:

### **Initialisation functions:**

- DEL\_init
- DEL\_get\_id
- DEL\_test\_id
- DEL\_get\_version
- DEL\_set\_mode
- DEL\_get\_mode
- DEL\_get\_init\_status
- DEL\_get\_module\_info
- DEL\_test\_module
- DEL\_get\_error\_string

### **Setup functions:**

- DEL\_get\_parameters
- DEL\_set\_parameters
- DEL\_set\_slope
- DEL\_set\_threshold
- DEL\_set\_output\_pulse\_width
- DEL\_set\_output\_pulse\_level
- DEL\_invert\_output\_pulse
- DEL\_set\_delay\_range
- DEL\_set\_delay
- DEL\_set\_start\_delay
- DEL\_set\_end\_delay
- DEL\_set\_scan\_repeat
- DEL\_set\_time\_per\_step
- DEL\_get\_eeprom\_data
- DEL\_write\_eeprom\_data
- DEL\_get\_adjust\_parameters
- DEL\_set\_adjust\_parameters
- DEL\_load\_setup
- DEL\_save\_setup
- DEL\_activate\_12V

### **Scanning control functions:**

- DEL\_start\_scan
- DEL\_stop\_scan
- DEL\_reset\_scan

### **Status functions:**

- DEL\_test\_if\_scanning
- DEL\_get\_delay

DEL\_test\_if\_triggered

Functions listed above must be called with C calling convention which is default for C and C++ programs.

Identical set of functions is available for environments like Visual Basic which requires `_stdcall` calling convention. Names of these functions have 'std' letters after 'DEL', for example, `DELstd_get_delay` it is `_stdcall` version of `DEL_get_delay`.

Description and behaviour of these functions are identical to the functions from the first (default) set – the only difference is calling convention.

# Application Guide

## Initialisation of the DEL Parameters

Before the DEL module can be used the parameter values must be written into the internal structures of the DLL functions (not directly visible from the user program) and sent to the control registers of the DEL module. This is accomplished by the function **DEL\_init**.

The DEL DLL functions simultaneously control up to four DEL-150(350) modules.

The **DEL\_init** function

- reads the parameter values from a specified file
- checks and recalculates the parameters depending on the hardware restrictions and the adjust parameters from the EEPROM on DEL module
- sends the parameter values to the DEL control registers
- performs a hardware test of the DEL module
- reads the parameter values from the 'auto.cfg' file and sends the values to the DEL control registers. For this feature the 'use\_auto\_setup' parameter in the 'ini' file must be set to 1 and the 'auto.cfg' file must exist in DLL directory.

The initialisation file is an ASCII file with a structure as shown below. We recommend either to use the file 'del150.ini' for initialisation or to start with 'del150.ini' and to introduce the desired changes.

```
; DEL150 initialisation file
; DEL parameters have to be included in .ini file only when parameter
; value is different from default.

[base]
simulation = 0                ; 0 - hardware mode(default) ,
                              ; >0 - simulation mode (see del_def.h for possible values)

baseadr = 0x380              ;base I/O address (0 ... 0x3FC,default 0x380)
use_auto_setup = 0          ;load (1) or not (0- default) parameters from setup file 'auto.cfg'
                              ; at the end of initialisation

pci_card_no = 0              ; number of the module on PCI bus if DEL-350 module
                              ; 0 - 3, default -1 ( ISA module - DEL-150)

pci_bus_no = -1              ; PCI bus on which DEL-350 modules will be looking for
                              ; 0 - 255, default -1 ( all PCI busses will be scanned)

[device]
slope = 1                    ;input signal slope -1 - negative, 1 - positive,
                              ; 0 - input disabled (default)
threshold = 0.08             ;input signal threshold [V] (-2V .. +2V , default 1.0 V )
out_width = 200.0            ;output signal width [ns] (2 ... 2000, default 10. )
out_level = 1                ;output signal level 0 - TTL ,1 - NIM (default)
delay_range= 10.             ;delay range [ns] (10 ... 100000 , default 100. )
delay = 5.0                  ;delay value [ns] ( 0 ... delay range , default 50. )
start_delay = 0.0            ;scan start delay value [ns]( 0 ... delay range , default 0.0 )
end_delay = 5.0              ;scan end delay value [ns] ( 0 ... delay range , default 50. )
time_per_step = 0.001        ; time per one scan step [sec]
                              ; ( 1e-6 ... 81.92 , default 0.001 )
scan_repeat= 0               ; automatic repeat of scan action (0 -off (default) ,1 -on)
out12V = 1                   ; 12V output active(1, default) / inactive ( 0)
```

; - only for DEL-350 modules

The **DEL\_init** function initialises one DEL module (normally the first one) depending on 'pci\_card\_no' value in .ini file. Module number to be used in subsequent calls to DLL functions is normally 0 and can be found by checking which module is in\_use ( see DEL\_get\_module\_info function) .

To initialise more (or other ) DEL modules ( if there are any ) DEL\_set\_mode function is used.

After calling the **DEL\_init** function the measurement parameters from the initialisation file are present in the module control registers and in the internal data structures of the DLLs. To give the user access to the parameters, the function **DEL\_get\_parameters** is provided. This function transfers the parameter values from the internal structures of the DLLs into a structure of the type 'DELdata' (see del\_def.h) which has to be defined by the user. The parameter values in this structure are described below.

unsigned short base_adr	base I/O address
short use_auto_setup;	auto.cfg file was loaded during initialisation
short pci_card_no;	no of PCI module(0-3) or -1 for ISA module
unsigned short test_eep;	test EEPROM or not
float threshold;	input signal threshold [V]
float out_width;	output signal width [ns]
short out_level;	output signal level 0 -TTL ,1 - NIM
short out_invert;	output signal inverted (1) or not (0)
float delay_range;	delay range [ns]
float delay;	delay [ns]
float start_delay;	start delay for scanning [ns]
float end_delay;	end delay for scanning [ns]
float time_per_step;	time per one step change of delay during scanning [s] (one step change = 4095 / actual delay range )
short steps;	number of steps in scanning action
short scan_repeat;	automatically repeat scanning ( or not)
short slope;	input signal slope <0 - negative,>0 - positive, 0 - input signal disabled - device not active
short out12V;	12V output active(1, default) / not active(0) - only for DEL-350 modules

To send the hardware parameters back to the DLLs and to the DEL module (e.g. after changing parameter values) the function **DEL\_set\_parameters** is used. This function checks and - if required - recalculates all parameter values due to cross dependencies and hardware restrictions. Therefore, it is recommended to read the parameter values by DEL\_get\_parameters after calling DEL\_set\_parameters.

The current state of the internal DELdata structure can be stored in a setup file (extension: .cfg) by the function **DEL\_save\_setup**. A previously saved state can be to a DEL module by calling the **DEL\_load\_setup** function.

Single parameter values can be transferred to the DLL and module level by parameter specific functions (e.g. for input signal slope use **DEL\_set\_slope** ).

## Error Handling

Each DEL DLL function returns an error status. Return values  $\geq 0$  indicate error free execution. A value  $< 0$  indicates that an error occurred during the execution. The meaning of a

particular error code can be found in `del_def.h` file and can be read using **`DEL_get_error_string`**. We recommend to check the return value after each function call.

## Description of the DEL DLL Functions

---

### short DEL\_init (char \* ini\_file)

---

Input parameters: ini\_file: pointer to a string the containing initialisation file name including file name and extension)

Return value: returns error code, 0 - success, <0 - error

DEL\_init initialises the library-internal structures with parameters from the configuration file 'ini\_file' (recommended extension .ini) and initialises the internal registers of the DEL module.

The **DEL\_init** function

- reads the parameter values from a specified file 'ini\_file'
- checks the base I/O addresses for all active modules to avoid hardware conflicts
- checks and recalculates the parameters depending on hardware restrictions and the adjust parameters from the EEPROM on the DEL module
- sends the parameter values to the DEL control registers
- performs a hardware test of the DEL module
- reads the parameter values from the 'auto.cfg' file and sends the values to the DEL control registers. For this feature the 'use\_auto\_setup' parameter in the 'ini' file must be set to 1 and the 'auto.cfg' file must exist in DLL directory.

The DEL module, which will be initialised, is defined by 'pci\_bus\_no' and 'pci\_card\_no' parameter from ini\_file.

'pci\_card\_no' defines which DEL module on PCI or ISA bus to be initialised:

- value 0 – 3 defines one specific DEL-350 module ( if 'pci\_card\_no' is greater than number of DEL-350 modules on PCI bus, it is rounded to the number of modules –1 )
- value –1 means that that the function will try to initialise DEL-150 module on ISA bus (using 'baseadr' from .ini file as a base IO address)

'pci\_bus\_no' defines which PCI bus with DEL-350 modules will be initialised:

- value 0 – 255 defines specific bus number (from the range: 0 to number of PCI busses with DEL modules) ( if 'pci\_bus\_no' is greater than number of PCI busses with DEL modules, it is rounded to the number of busses –1 )
- value –1 means that that the function will try to find DEL modules on all PCI busses

The module will be initialised, but only when it is not in use (locked) by other application (valid only for DEL-350 modules).

After successful initialisation the module is locked to prevent that other application can access it (valid only for DEL-350 modules). The user should check initialisation status of all modules he wants to use by calling DEL\_get\_init\_status function. It is also recommended to call DEL\_get\_module\_info function to get additional module's information.

If, for some reasons, the module which was locked must be initialised, it can be done using the function DEL\_set\_mode with the parameter 'force\_use' = 1.

The initialisation file is an ASCII file with a structure as shown below. We recommend either to use the file del150.ini for initialisation or to start with del150.ini and to introduce the desired changes.

```

; DEL150 initialisation file
; DEL parameters have to be included in .ini file only when parameter
; value is different from default.

[base]
simulation = 0                ; 0 - hardware mode(default) ,
                               ; >0 - simulation mode (see del_def.h for possible values)

baseadr = 0x380               ;base I/O address (0 ... 0x3FC,default 0x380)
use_auto_setup = 0           ;load (1) or not (0- default) parameters from setup file 'auto.cfg'
                               ; at the end of initialisation

pci_card_no = 0               ; number of the module on PCI bus if DEL-350 module
                               ; 0 - 3, default -1 ( ISA module - DEL-150)

pci_bus_no = -1               ; PCI bus on which DEL-350 modules will be looking for
                               ; 0 - 255, default -1 ( all PCI busses will be scanned)

[device]
slope = 1                     ;input signal slope -1 - negative, 1 - positive,
                               ; 0 - input disabled (default)

threshold = 0.08              ;input signal threshold [V] (-2V .. +2V , default 1.0 V )
out_width = 200.0             ;output signal width [ns] (2 ... 2000, default 10. )
out_level = 1                 ;output signal level 0 - TTL ,1 - NIM (default)
delay_range = 10.             ;delay range [ns] (10 ... 100000 , default 100. )
delay = 5.0                   ;delay value [ns] ( 0 ... delay range , default 50. )
start_delay = 0.0             ;scan start delay value [ns]( 0 ... delay range , default 0.0 )
end_delay = 5.0               ;scan end delay value [ns] ( 0 ... delay range , default 50. )
time_per_step = 0.001         ; time per one scan step [sec]
                               ; ( 1e-6 ... 81.92 , default 0.001 )

scan_repeat = 0               ; automatic repeat of scan action (0 -off (default) ,1 -on)
out12V = 1                    ; 12V output active(1, default) / inactive ( 0)

```

After calling the **DEL\_init** function the measurement parameters from the initialisation file are present in the module control registers and in the internal data structures of the DLLs.

---

### short DEL\_get\_id (short mod\_no, short far\* id)

---

Input parameters:

mod\_no                    0 .. 3, DEL module number  
\*id                        pointer to the identification code

Return value:

0    no errors,    <0   error code

Description:

The procedure loads the 'id' variable with the identification code of the DEL module 'mod\_no'. 'DEL\_get\_id' is a low level procedure which is already called during the

initialisation by DEL\_init. Therefore, a call of 'DEL\_get\_id' is not normally required for standard applications.

---

### **short DEL\_test\_id (short mod\_no)**

---

Input parameters:

mod\_no                    0 .. 3, DEL module number

Return value:

0     correct id ,   <0   error code

Description:

The procedure checks whether the DEL module 'mod\_no' has the correct identification code. DEL\_test\_id is a low level procedure which is already called during the initialisation by DEL\_init. Therefore, a call of 'DEL\_get\_id' is not normally required for standard applications.

---

### **short DEL\_get\_version (short mod\_no, short far\* version)**

---

Input parameters:

mod\_no                    0 .. 3, DEL module number

\*version                  pointer to the version variable

Return value:

0     no errors,   <0   error code

Description:

The procedure loads the 'version' variable with the FPGA identification code of the DEL module 'mod\_no'. DEL\_get\_version is a low level procedure not needed normally.

---

### **short DEL\_set\_mode (short mode, short force\_use, short \*in\_use)**

---

Input parameters:

mode                      mode of DLL operation

force\_use                  force using the module if they are locked ( in use)

\*in\_use                    pointer to the table with information which module must be used

Return value:

on success - DLL mode ( >= 0 ) ,     <0     error code

Description:

The procedure is used to change the mode of the DLL operation between the hardware mode and the simulation mode. It is also used to switch the DLL to the simulation mode if hardware errors occur during the initialisation.

Table 'in\_use' should contain entries for all 4 modules:

0 – means that the module will be unlocked and not used longer

1 – means that the module will be initialised and locked

When the Hardware Mode is requested for each of 4 possible modules:

-if 'in\_use' entry = 1 : the proper module is locked and initialised (if it wasn't) with the initial parameters set (from ini\_file) but only when it was not locked by another application or when 'force\_use' = 1.

-if 'in\_use' entry = 0 : the proper module is unlocked and cannot be used further.

When one of the simulation modes is requested for each of 4 possible modules:

-if 'in\_use' entry = 1 : the proper module is initialised (if it wasn't) with the initial parameters set (from ini\_file).

-if 'in\_use' entry = 0 : the proper module is unlocked and cannot be used further.

Errors during the module initialisation can cause that the module is excluded from use.

Use the function DEL\_get\_init\_status and/or DEL\_get\_module\_info to check which modules are correctly initialised and can be use further.

Use the function DEL\_get\_mode to check which mode is actually set. Possible 'mode' values are defined in the del\_def.h file.

The procedure is used to change the DLL operating mode from 'Hardware Mode' to 'Simulation Mode' or vice versa. The simulation mode is used to run the software without the DEL module or in case of hardware errors found during the DEL\_init call. Use DEL\_get\_mode to check the actual mode after the call to DEL\_set\_mode.

---

### short DEL\_get\_mode (void)

---

Input parameters:

none

Return value: DLL operation mode

The procedure returns the current DLL operation mode. Possible 'mode' values are defined in the del\_def.h file:

```
#define DEL_HARD          0          /* hardware mode */
#define DEL_SIMUL150     150        /* simulation mode of DEL-150 module */
#define DEL_SIMUL350     350        /* simulation mode of DEL-350 module */
```

---

### short DEL\_get\_init\_status(short mod\_no)

---

Input parameters:

mod\_no                    0 .. 3, DEL module number

Return value: initialisation result code of the DEL module 'mod\_no'

Description:

The procedure returns the initialisation result code set by the function DEL\_init or DEL\_set\_mode. The possible values are shown below (see also del\_def.h):

INIT_OK	0	no error
INIT_NOT_DONE	-1	init not done
INIT_WRONG_EEP_CHKSUM	-2	wrong EEPROM checksum
INIT_WRONG_MOD_ID	-3	wrong module identification code
INIT_WRONG_TRIGGER	-4	trigger test failed
INIT_WRONG_DACs	-5	DAC's test failed
INIT_CANT_OPEN_PCI_CARD	-6	cannot open PCI card
INIT_WINDRVR_VER	-7	incorrect WinDriver version
INIT_MOD_IN_USE	-8	module in use (locked) - cannot initialise
INIT_WRONG_LICENSE	-9	wrong or missing license key

---

### **short DEL\_get\_module\_info(short mod\_no, DELModInfo \* mod\_info);**

---

Input parameters:

mod\_no                    0 .. 3, DEL module number  
 \*mod\_info                pointer to result structure (type DELModInfo)

Return value: 0 no errors, <0 error code (see del\_def.h)

Description:

After calling the DEL\_init or DEL\_set\_mode function (see above) the DELModInfo internal structures for all 4 modules are filled. This function transfers the contents of the internal structure of the DLL into a structure of the type DELModInfo (see del\_def.h) which has to be defined by the user. The parameters included in this structure are described below.

short module_type	DEL module type (see del_def.h)
short bus_number	PCI bus number of the module
short slot_number	slot number on PCI bus 'bus_number' occupied by the module
short in_use	-1 used and locked by other application, 0 - not used, 1 - in use
short init	set to initialisation result code
unsigned short base_adr	base I/O address on PCI bus

---

### **short DEL\_test\_module (short mod\_no)**

---

Input parameters:

mod\_no                    0 .. 3, DEL module number

Return value:

0    no errors ,    <0    error code

Description:

The procedure performs a hardware test of DEL module. This is low level procedure called already during the initialisation by DEL\_init.

---

### **short DEL\_get\_error\_string(short error\_id, char \* dest\_string, short max\_length);**

---

Input parameters:

error\_id                DEL DLL error id (0 – number of DEL errors-1) (see del\_def.h file)

\*dest\_string pointer to destination string  
max\_length max number of characters which can be copied to 'dest\_string'

Return value: 0: no errors, <0: error code

The procedure copies to 'dest\_string' the string which contains the explanation of the DEL DLL error with id equal 'error\_id'. Up to 'max\_length' characters will be copied. Possible 'error\_id' values are defined in the del\_def.h file.

---

### **short DEL\_get\_parameters (short mod\_no, DELdata far \* data)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
data	pointer to result structure (type DELdata)

Return value: 0

Description:

After calling the **DEL\_init** function (see above) the measurement parameters from the initialisation file are present in the module and in the internal data structures of the DLLs. To give the user access to the parameters, the function **DEL\_get\_parameters** is provided. This function transfers the parameter values of the module 'mod\_no' from the internal structures of the DLLs into a structure of the type 'DELdata' (see del\_def.h) which has to be defined by the user. The parameter values in this structure are described below.

unsigned short base_adr	base I/O address
short use_auto_setup;	auto.cfg file was loaded during initialisation
short pci_card_no;	no of PCI module(0-3) or -1 for ISA module
unsigned short test_eep;	test EEPROM or not
float threshold;	input signal threshold [V]
float out_width;	output signal width [ns]
short out_level;	output signal level 0 -TTL ,1 - NIM
short out_invert;	output signal inverted (1) or not (0)
float delay_range;	delay range [ns]
float delay;	delay [ns]
float start_delay;	start delay for scanning [ns]
float end_delay;	end delay for scanning [ns]
float time_per_step;	time per one step change of delay during scanning [s] (one step change = 4095 / actual delay range )
short steps;	number of steps in scanning action
short scan_repeat;	automatically repeat scanning ( or not)
short slope;	input signal slope <0 - negative,>0 - positive, 0 - input signal disabled - device not active
short out12V;	12V output active(1, default) / not active(0) - only for DEL-350 modules

---

**short DEL\_set\_parameters (short mod\_no, DELdata far \*data)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
data	pointer to result structure (type DELdata)

Return value:

0 no errors , <0 error code

Description:

The procedure sends all parameters from the 'data' structure to the internal DLL structures and to the control registers of the DEL module.

The new parameter values are recalculated according to the parameter limits or hardware restrictions (e.g. DAC resolution). Furthermore, cross dependencies between different parameters are taken into account to ensure the correct hardware operation. It is recommended to read back the parameters after setting to get their real values after recalculation.

The values of 'base\_adr', 'pci\_card\_no' and 'test\_eep' are not changed. They can be changed only by a new config\_file and a new DEL\_init call.

---

**short DEL\_set\_slope (short mod\_no, short slope)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
slope	new slope value- -1 negative, 1 positive, 0 – input disabled

Return value:

0 no errors , <0 error code

Description:

The procedure sets the trigger slope for the DEL module defined by 'mod\_no'.

---

**short DEL\_set\_threshold (short mod\_no, float threshold)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
threshold	new threshold value (-2V +2V)

Return value:

0 no errors , <0 error code

Description:

The procedure sets the trigger threshold for the DEL module defined by 'mod\_no'.

---

**short DEL\_set\_output\_pulse\_width (short mod\_no, float width)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
width	new width value (2 to 2000 [ns])

Return value:

0 no errors , <0 error code

Description:

The procedure sets the output pulse width for the DEL module defined by 'mod\_no'.

---

**short DEL\_set\_output\_pulse\_level (short mod\_no, short level)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
level	new level value- 0 – TTL, 1 - NIM

Return value:

0 no errors , <0 error code

Description:

The procedure sets the level of the output signal for the DEL module defined by 'mod\_no'.

---

**short DEL\_invert\_output\_pulse (short mod\_no, short invert)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
invert	new invert value- 0 – no inversion, 1 - inversion

Return value:

0 no errors , <0 error code

Description:

The procedure is used to set the logical polarity of the output signal (non inverted / inverted). The module to which the setting applies is defined by 'mod\_no'.

---

**short DEL\_set\_delay\_range (short mod\_no, float range)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
range	new range value (10 to 100000 [ns])

Return value:

0 no errors , <0 error code

Description:

The procedure sets the delay range for the DEL module defined by 'mod\_no'.

In order to avoid discontinuities in the delay scale we recommend to set 'range' to the maximum required delay value and to control the delay by DEL\_set\_delay.

---

**short DEL\_set\_delay (short mod\_no, float delay)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
delay	new delay value (0 to 'delay range' [ns])

Return value:

0 no errors , <0 error code

Description:

The procedure sets the delay value for the DEL module 'mod\_no'. The delay can be set to any value within the 'Delay Range' set by DEL\_set\_delay\_range. The module to which the parameter applies is defined by 'mod\_no'.

On order to avoid discontinuities in the delay scale we recommend to set 'Range' to the maximum required delay value and to essentially control the delay by DEL\_set\_delay.

---

**short DEL\_set\_start\_delay (short mod\_no, float delay)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
delay	new start delay value (0 to delay range [ns])

Return value:

0 no errors , <0 error code

Description:

The procedure sets the start delay value used in the scanning mode. The module to which the parameter applies is defined by 'mod\_no'.

---

**short DEL\_set\_end\_delay (short mod\_no, float delay)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
delay	new end delay value (0 delay range [ns])

Return value:

0 no errors , <0 error code

Description:

The procedure sets the start delay value used in the scanning mode. The module to which the parameter applies is defined by 'mod\_no'.

---

**short DEL\_set\_scan\_repeat (short mod\_no, short repeat)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
repeat	automatic repeat of scan action: 0 – off, 1 - on

Return value:

0 no errors , <0 error code

Description:

The procedure switches on/off the automatic repeating of scanning action for the DEL module defined by 'mod\_no'.

---

**short DEL\_set\_time\_per\_step (short mod\_no, float time)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
time	new time value (1e-6 ... 81.92 [s])

Return value:

0 no errors , <0 error code

Description:

The procedure sets the time per delay step in the scanning mode. The module to which the parameter applies is defined by 'mod\_no'.

---

**short DEL\_get\_eeprom\_data (short mod\_no, DEL\_EEP\_Data far \*eep\_data)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
eep_data	pointer to result structure

Return value:

0 no errors , <0 error code

Description:

The structure "eep\_data" is filled with the adjust and manufacturing parameters in the EEPROM of the DEL module specified by 'mod\_no'. The structure "DEL\_EEP\_Data" is defined in the file del\_def.h.

Normally, the EEPROM data need not be read explicitly because the EEPROM is read during DEL\_init and the module type information and the adjust values are taken into account when the DEL module registers are loaded.

---

**short DEL\_write\_eeprom\_data (short mod\_no, unsigned short write\_enable, DEL\_EEP\_Data far \*eep\_data)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
write_enable	write enable password
eep_data	pointer to result structure

Return value:

0 no errors , <0 error code

Description:

The function is used to write data to the EEPROM of an DEL module specified by 'mod\_no'. To prevent corruption of the adjust data the function writes data to the EEPROM only if the 'write\_enable' password is correct.

---

**short DEL\_get\_adjust\_parameters (short mod\_no, DEL\_Adjust\_Para far \*adjpara)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
adjpara	pointer to result structure

Return value:

0 no errors , <0 error code

Description:

The structure 'adjpara' is filled with the adjust parameters that are currently in use. The parameters can either be previously loaded from the EEPROM by DEL\_init or DEL\_get\_eeprom\_data or - not recommended - set by DEL\_set\_adust\_parameters.

The structure "DEL\_Adjust\_Para" is defined in the file del\_def.h.

Normally, the adjust parameters need not be read explicitly because the EEPROM is read during DEL\_init and the adjust values are taken into account when the DEL module registers are loaded.

---

**short DEL\_set\_adjust\_parameters (short mod\_no, DEL\_Adjust\_Para far \* adjpara)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
adjpara	pointer to result structure

Return value:

0 no errors , <0 error code

Description:

The adjust parameters in the internal DLL structures (not in the EEPROM) are set to values from the structure "adjpara". The function is used to set the module adjust parameters to values other than the values from the EEPROM. The new adjust values will be used until the next call of DEL\_init. The next call to DEL\_init replaces the adjust parameters with the values from the EEPROM. We strongly discourage to use modified adjust parameters, because the module function can be seriously impaired by wrong adjust values.

The structure "DEL\_Adjust\_Para" is defined in the file del\_def.h.

---

**short DEL\_save\_setup (short mod\_no, char far \*setup\_file)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
setup_file	pointer to string containing setup file name and path

Return value:

0 no errors , <0 error code

Description:

The function is used to store the DEL module parameters to the file 'setup\_file'. The file can be loaded back by the DEL\_load\_setup function. The file contains a copy of internal DLL DEL data structure.

---

**short DEL\_load\_setup (short mod\_no, char far \*setup\_file)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
setup_file	pointer to string containing setup file name and path



---

**short DEL\_stop\_scan (short mod\_no)**

---

Input parameters:

mod\_no                    0 .. 3, DEL module number

Return value:

0    no errors ,    <0    error code

Description:

The function is used to stop a current scanning action. After being stopped, the scanning can be continued from the current state by the function DEL\_start\_scan. To reset the scan state the DEL\_reset\_scan function is provided. The current delay can be read by the function DEL\_get\_delay.

If DEL\_stop\_scan is called with no scanning action started an error is returned.

---

**short DEL\_reset\_scan (short mod\_no)**

---

Input parameters:

mod\_no                    0 .. 3, DEL module number

Return value:

0    no errors ,    <0    error code

Description:

The function is used to reset the current scanning action. The scanning action is stopped and the current delay is set to 'start delay'.

---

**short DEL\_test\_if\_scanning (short mod\_no)**

---

Input parameters:

mod\_no                    0 .. 3, DEL module number

Return value:

>0    no errors, scanning status ,    <0    error code

Description:

The function is used to test the state of the scanning action. The function returns the status of the scanning action. Possible status values are listed below ( see also del\_def.h file).

STATUS_RESET	100
STATUS_STOP	200
STATUS_SCAN	300
STATUS_READY	400

---

**short DEL\_get\_delay (short mod\_no, float \*delay)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
delay	pointer to variable which will be filled with delay value

Return value:

0 no errors , <0 error code

Description:

The function sets the 'delay' variable to the actual delay value (in [ns] ) of the DEL module 'mod\_no'.

---

**short DEL\_test\_if\_triggered (short mod\_no, short far \*triggered)**

---

Input parameters:

mod_no	0 .. 3, DEL module number
triggered	pointer to variable to be set

Return value:

0 no errors , <0 error code

Description:

The function sets the 'triggered' variable to 0 if the DEL module 'mod\_no' was not triggered or to 1 if the module was triggered since the initialisation or since the last call of 'DEL\_test\_if\_triggered'.