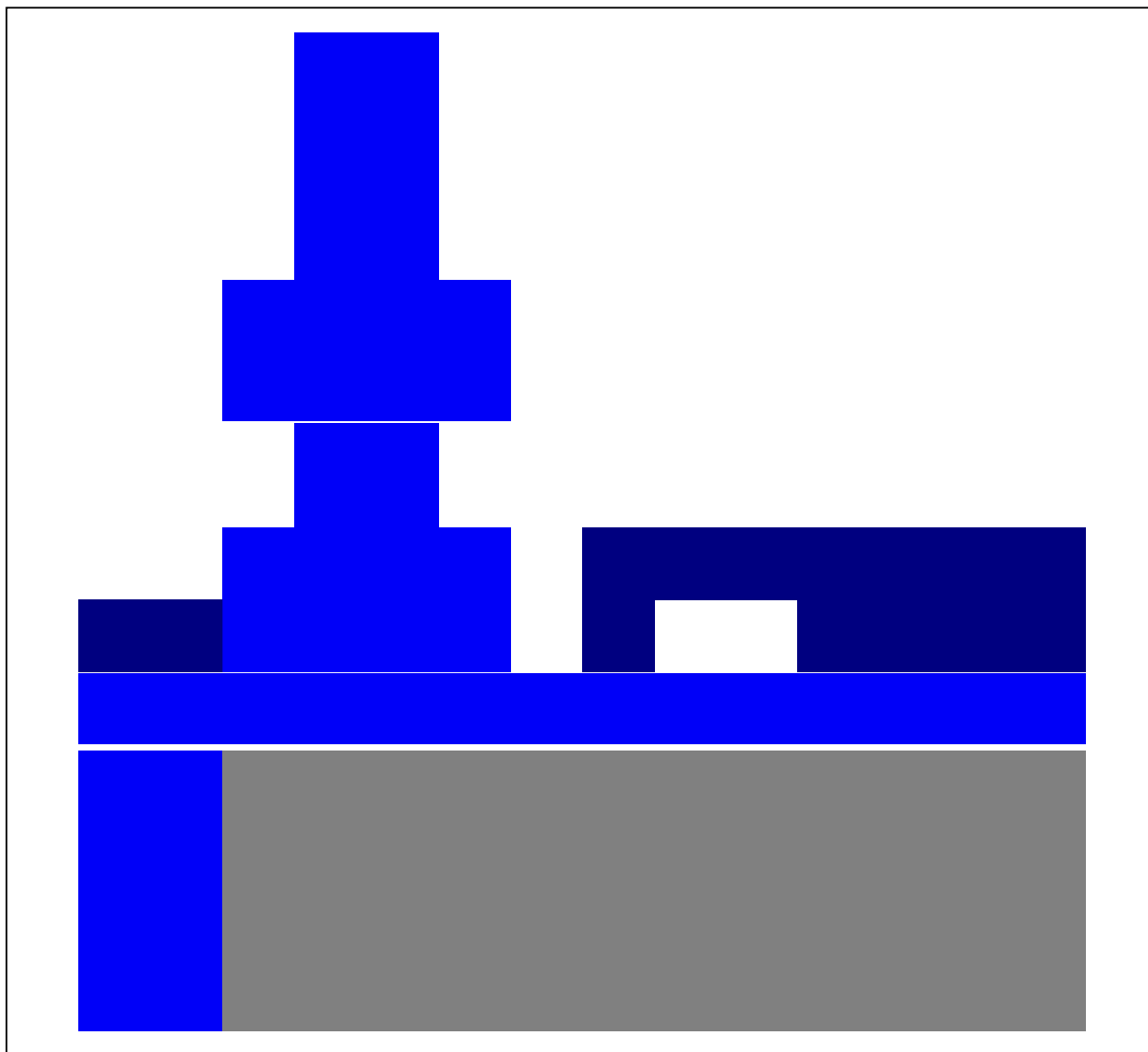


■ FP6470/60

Windows CM Remote Control (32-bits)



Operating Manual

Fourth edition
030601

FEI
Electron
Optics



Fourth edition

Date: 01-06-2003

© FEI Company - Eindhoven - The Netherlands - 2003

Preface

With the CM microscope having significantly increased the accuracy, speed and efficiency of your daily work program, you now also have the possibility of personalising and customising your instrument even further with the Windows[®] Remote Control package for the CM microscope.

The Windows Remote Control package consists of a user program, CM Monitor, and two Tutor programs (which in themselves can also be used to control the CM microscope).

CM Monitor enables you to save and retrieve alignments, stigmator settings and mode settings. You may find this helpful in daily operation and specifically in multi-user and training situations, where a number of microscope users may wish to save and retrieve their own specific operational settings.

With the Delphi[®] and Visual Basic[®] Tutor for Remote Control, it becomes easy to write your own Windows programs for controlling the CM microscope. Examples guide the way to all Remote Control functions, using many of the typical Windows user-interface elements such as dialogs, message boxes and help files.

If you intend to do your own programming and wish to distribute programs on a wider scale, or if you find errors, have questions or suggestions about the Remote Control, or want an update on freeware programs (programs which may be useful at the microscope and which may be distributed freely) available, please contact:

Dr. M.T. Otten
FEI Electron Optics Applications Laboratory
Building AAE
PO Box 80066
5800 KA Eindhoven
The Netherlands
tel. +31-40-2766106
fax +31-40-2766102
e-mail mto@nl.feico.com

Delphi is a trademark of Inprise Corporation.
Windows and Visual Basic are trademarks of Microsoft Corp.

Licence Agreement for

Windows CM Remote Control Software (32-bits) (FP6470/60)

1. Agreement

FEI grants to the buyer of the license - hereafter called "Licensee" - the right to use the product as specified in article 2, hereafter called "the Product", in consideration of the terms and conditions set forth in the subsequent articles. Licensee assents to this agreement by using the software.

2. The Product

Product name : Windows CM Remote Control Software (32-bits)
Type number : FP6470/60

3. Right of Use

The product may be used to interface with one single CM microscope. All rights, title and interest in and to the licensed product remain at all times the sole and exclusive property of FEI and/or in case FEI sub-license the product, of the third party ("Licensor").

4. Copyright and Non-disclosure

Licensee may make one (1) copy, in machine-readable form, of the product for backup or archiving purpose only. Licensee ensures that such a copy contains any and all proprietary, copyright and other notices and identification numbers which appear on the product from which the copy is made. Licensee will not make any more copies of the product. FEI grants Licensee the right to use the source code of the product as specified herein and to modify the source according to Licensee's specifications. Tutor Toolkit files, the REMOTECM.DLL and SECS2_32.EXE files may be copied freely and incorporated in or distributed with user-written programs.

5. Right to Audit

Licensee grants FEI the right to inspect Licensee's premises for the purpose of verification of the correct use of the Product, in accordance with this agreement.

6. Warranty

FEI guarantees that, at the time of shipment, the product will meet the published specifications. FEI does not guarantee that the licensed product is error-free. This is a limited guarantee and the only one given to Licensee and it replaces any other express, implied, statutory, or any other warranty. This guarantee may not be transferred to any subsequent user of the Product.

7. Termination

This agreement may be terminated by Licensee at any time, upon thirty (30) days written notice to FEI. Such termination shall not give Licensee any right to reimbursement of royalty fee. FEI may terminate this agreement at any time if Licensee does not comply with any of the terms and conditions of this agreement, upon thirty (30) days written notice to Licensee. In the event of termination, whether by Licensee or FEI, Licensee shall promptly stop using the product and shall return the product and any copies thereof, to FEI.

8. Limitation of Liability

FEI shall not be liable for any loss of profits, loss of use, interruption of business, or indirect or consequential damages resulting from the use of the product.

9. Exclusion of Reconfiguration

If Licensee wishes to resell modified parts of the source code, he shall first obtain a "License for Reconfiguration" from FEI.

10. General

In the event of termination because of Licensee's failure to comply with the obligations of this agreement, FEI reserves the right to invoke any and all remedies available in law or in the contract, including the right to claim damages resulting from non-agreed use of the product.

11. Applicable Law

This Agreement shall be considered to be construed and concluded in the Netherlands and governed by the laws of the Netherlands. In case of any dispute arising out of this Agreement the courts competent at 's-Hertogenbosch, the Netherlands, shall have sole jurisdiction.

1	INTRODUCTION	1
2	RELATION TO PREVIOUS VERSIONS.....	2
2.1	16-bits and 32-bits software	2
2.2	Difference with previous versions : CloseConnection.....	2
3	GETTING STARTED.....	3
3.1	HARDWARE AND SOFTWARE CONFIGURATION	3
3.2	CONNECTION BETWEEN MICROSCOPE AND REMOTE CONTROL PC.....	3
3.3	PREPARATION OF THE MICROSCOPE	3
3.4	TROUBLESHOOTING	4
4	INSTALLATION OF THE SOFTWARE	5
4.1	INSTALLING SOFTWARE	5
4.2	FIRST-TIME OPERATION WITH SECS2_32.EXE.....	6
5	STRUCTURE OF WINDOWS REMOTE CONTROL	7
6	WINDOWS CM MONITOR.....	8
7	WINDOWS TUTOR PROGRAMS.....	10
7.1	USING THE TUTOR PROGRAMS.....	10
7.1.1	Tutor windows	10
7.1.2	Tutor Menu	12
7.1.3	File menu	13
7.1.4	Info menu	13
7.1.5	Softkey menu	15
7.1.6	Pushbutton menu's.....	15
7.1.7	Turnknob menu	15
7.1.8	Mode menu	15
7.1.9	Direct operation menu	16
7.1.10	Goniometer menu	16
7.1.11	Window menu	18
7.1.12	Help menu.....	19
7.2	PROGRAMMING WITH TUTOR.....	20
7.2.1	Introduction	20
7.2.2	Getting started.....	20
7.3	DELPHI	20
7.3.1	Program structure.....	22
7.3.2	Program file RCMWIN32.PAS	22
7.3.3	Remote control functions	23
	Function AbortSecsAction.....	24
	Function ChangeFreeHT	24
	Procedure CloseConnection	24
	Function CSAskPos	24
	Function CSGetRegs	24
	Function CSGotoPos;	25
	Function CSSetRegs	25
	Function CurrentReadout.....	25
	Function DirectOperation	26
	Function EmissionCurrent.....	27
	Function EnableUserInput	27
	Function EquipmentAvailable.....	27
	Function GetAlignment	27
	Function GetCmCamValues	28

Function GetCmInfo	28
Function GetCmVar	29
Function GetFegData.....	30
Function GetHtAcknowledge.....	30
Function GetMode	31
Function GetRotationAlignment	31
Function GetStigmators	31
Function InstrumentMode	32
Function PressureReadout	32
Function PushButton.....	33
Function ResetDisplay	33
Function RetrEDXProt.....	34
Function RetrExtXY.....	34
Function RetrHTCond	34
Function ScreenCurrent.....	34
Function SetAlignment.....	34
Function SetMode.....	35
Function Set...Stigmator	35
Function SetRotationAlignment	35
Function SetTimeOutTime	36
Function SetTVdetPos	36
Function Softkey	36
Function SRSCalibrate	36
Function SRSGotoPos.....	36
Function SRSGotoReg	37
Function SRSReadPos	37
Function SRSReadReg.....	37
Function SRSSelectReg	37
Function SRSSetReg.....	38
Function SwitchFreeHT.....	38
Function TurnKnob	38
Function VideoL/VideoR	39
7.3.4 Tips	40
7.4 VISUAL BASIC.....	41
7.4.1 Program structure.....	41
7.4.2 Program file RCMWIN32.BAS	41
7.4.3 Visual Basic-specific comments	41
7.4.4 Remote control procedures	42
Function AbortSecsAction.....	43
Function ChangeFreeHT	43
Sub CloseConnection ().....	43
Function CSAskPos	43
Function CSGetRegs	43
Function CSGotoPos	44
Function CSSetRegs	44
Function CurrentReadout.....	45
Function DirectOperation	45
Function EmissionCurrent.....	46
Function EnableUserInput	46
Function EquipmentAvailable.....	46

Function GetAlignment	47
Function GetCmCamValues	47
Function GetCMInfo	48
Function GetCMvar	49
Function GetFegData	50
Function GetHtAcknowledge	50
Function GetMode	51
Function GetRotationAlignment	51
Function GetStigmators	51
Function InstrumentMode	52
Function PressureReadout	52
Function PushButton	53
Function ResetDisplay	53
Function RetrEDXProt	54
Function RetrExtXY	54
Function RetrHtCond	54
Function Screencurrent	54
Function SetAlignment	54
Function SetMode	55
Function SetRotationAlignment	55
Function Set...Stigmator	55
Function SetTimeOutTime	56
Function SetTVdetPos	56
Function Softkey	56
Function SRSCalibrate	56
Function SRSGotoPos	57
Function SRSGotoReg	57
Function SRSReadPos	57
Function SRSReadReg	57
Function SRSSelectReg	58
Function SRSSetReg	58
Function SwitchFreeHT	58
Function TurnKnob	59
Function VideoL/VideoR	59
7.4.5 Tips	60
8 PROGRAMMING IN OTHER LANGUAGES	61
8.1 INTRODUCTION	61
8.2 CONTROLLING THE DYNAMIC LINK LIBRARY	61
8.3 REMOTE CONTROL FUNCTIONS	62
8.3.1 Communication initiation, test and closing	62
EquipmentAvailable	62
CloseConnection	62
AbortSecsAction	62
SetTimeOutTime	63
8.3.2 Control of pushbuttons, softkeys and turnknobs	63
Pushbutton	63
Softkey	64
Turnknob	64
8.3.3 Control of specimen relocation system	65
Calibrate Specimen Relocation System	65

Move stage to location	65
Move stage to register location	66
Read current stage location	66
Read register contents	66
Select a register	66
Store coordinates in a register	67
8.3.4 Measuring currents, pressures and video signals	67
Read lens, deflection coils and stigmator currents	67
Read emission current	67
Read vacuum pressures	68
Read screen current	68
Read video signal left	68
Read video signal right	68
8.3.5 Retrieving and restoring microscope parameters	69
Retrieve alignment parameters	69
Retrieve camera settings information	69
Retrieve microscope information	70
Retrieve CM variables	71
Retrieve FEG parameters	72
Retrieve mode settings	72
Retrieve stigmator settings	73
Restore alignment parameters	73
Restore mode settings	73
Restore objective stigmator settings	74
Restore condenser stigmator settings	74
Restore diffraction stigmator settings	74
Get rotation alignment	74
Set rotation alignment	75
8.3.6 Retrieving status information	75
Check EDX protection	75
Check external XY deflection	75
Check high tension condition	76
Get high tension condition	76
8.3.7 Direct operations	76
Direct operations	76
EnableUserInput	77
ResetDisplay	78
Set instrument mode	78
Set TV detector position	79
Switch Free high tension control	79
Change Free high tension	79
8.3.8 Control of the CompuStage	80
CompuStage ask position	80
CompuStage go to position	80
CompuStage read registers	81
CompuStage write registers	82
9 EXPLANATION OF ERROR CODES	83
10 EXPLANATION OF SECS2 MESSAGES	85
10.1 INTRODUCTION	85
10.2 MESSAGE STRUCTURES	88

1 INTRODUCTION

The remote control accessory for FEI CM Transmission Electron Microscopes expands the instrument control capabilities in two specific directions:

- Firstly it can serve as a virtually unlimited storage system for instrument status and alignment information. CM Monitor is a program supporting this specific capability.
- Secondly, using the Delphi and Visual Basic Remote Control Tutor programs, pre-programming of operational procedures becomes available which may include instrument control, beam positioning or calculations based on on-line measurements. These types of programs may be developed by the user to his own specifications, using the Tutor programs delivered as source code as a programming example.

The hardware basis of the Remote Control System is a Personal Computer, running Microsoft Windows NT 3.5.1 or higher (NT 4.0, Windows 2000, Windows XP). By choosing this computer system a large number of other programs may also become relevant for the user.

This manual is a users guide to the Windows-based remote control system for the CM-series microscopes. It explains the start-up of the system, the use of the application programs included, and also gives an introduction to user programming.

The system is based on Delphi[®] version 4.0 or higher, or Visual Basic[®] version 6.0 or higher. Information for programming in other languages is also included (refer to chapter 7).

2 RELATION TO PREVIOUS VERSIONS

2.1 16-bits and 32-bits software

In the past, Philips Electron Optics (now FEI) has provided two types of remote control software. The generally available type (previously PW6470/50; now FP6470/50) consisted of the same functionality as the package described here but was 16-bits software. This version still runs under any Windows operating system but it cannot be used for programming with modern (32-bits) programming languages. For interfacing to 32-bits programming languages a different version is required. A few years ago a 32-bits version for Windows NT platforms was made but it was never released, because it turned out to be unstable. As a consequence, the 32-bits version was never sold as a product. It was distributed on a limited scale, however, because it was needed for interfacing with the software of some camera manufacturers such as SIS and AMT. To ensure proper operation, the old 32-bits software should be replaced by this new version. The camera manufacturers and other third-parties such as TSL are allowed to distribute the 32-bit driver software (SESC2_32, CMREMOTE32.DLL and RAM address files) to all current and new installations requiring 32-bits Remote Control free of charge. However, the user programs such as CM Monitor, Delphi Tutor and Visual Basic Tutor are **NOT** included in the free update and should be purchased from FEI (this package).

2.2 Difference with previous versions : CloseConnection

To ensure proper operation of the new 32-bits driver and dynamic link library, any program interfacing through the dynamic link library, should call the CloseConnection procedure as the last remote control call made. CloseConnection forces the dynamic link library to close its COM connection to the driver (there is no automatic way of doing this; by the time the dynamic link library is unloading, it is already too late). If the CloseConnection call is not made, there are two consequences (neither of them fatal):

1. The program normally will remain hanging on a single thread (visible in Task Manager).
2. The remote control driver will give an access violation on closing (Error 216).

3 GETTING STARTED

3.1 HARDWARE AND SOFTWARE CONFIGURATION

The required remote control system components are:

- Serial connection kit
- Remote control function (PW6460/00) (automatically released in CM software 14.1)
- Windows Remote CM Setup CD
- Any Personal Computer (true compatible) with Windows NT 3.5.1 or higher
- A suitable compiler if user programming is intended. Recommended languages are Delphi version 4.0 or higher and Visual Basic version 6.0 or higher.

3.2 CONNECTION BETWEEN MICROSCOPE AND REMOTE CONTROL PC

The accessory **Remote control function** (PW6460/00) activates the remote control system that is located in the microscope software (without this function, the microscope is closed to the outside world). The Remote Control function is switched on on one of the PARAMETERS pages of the microscope.

To get started, the serial communication port of the microscope must be connected to one of the serial communication ports of the remote control computer (RS232-C connection). This connection is provided by the **Serial connection kit** and the RS232-C cable from the rear plinth of the microscope to the serial port of the remote control computer.

3.3 PREPARATION OF THE MICROSCOPE

Make sure the CM microscope is operational. Select the operational page, and key PARAMETERS several times until you get the COMMUNICATION PORT page. Make sure that the following parameter values (default settings) are selected. The parameters can only be changed if remote control is switched off.

PARAMETERS
COMMUNICATION PORT
REMOTE CONTROL

PARAMETERS

DATA RATE Bd
150... 4800 9600 9600 ...9600

RECEIVE TIMEOUT S
0.10... 0.40 0.50 0.60 ...10.0

PROTOCOL TIMEOUT S
0.20... 1.80 2.00 2.20 ...12.0

REPLY TIMEOUT S
1.00... 44.0 45.0 46.0 ... 120

RETRY COUNT
0 ... 2 3 4 ... 31

MASTER MASTER/SLAVE SLAVE

Fig. 1. COMMUNICATION PORT page, configured for Remote Control.

3.4 TROUBLESHOOTING

Remote Control frozen

When an inappropriate sequence of commands is sent to the microscope (e.g. during development of user-written programs), the Remote Control inside the microscope may freeze up. If no more communication is possible and everything else is normal (Remote Control on, all cables connected), then try switching the Remote Control off and on at the microscope. If there still is no communication, try restarting the microscope and personal computer. For the microscope:

1. Press STANDBY button (do not use OFF or the restart underneath panel dim; by using STANDBY the IGP remains on and the vacuum ready condition is restored more rapidly).
2. Press ON after 10 seconds.
3. Switch Remote Control on.

Communication is always restored after a microscope restart.

Programs do not start, with message "Cannot find CMREMOTE32.DLL"

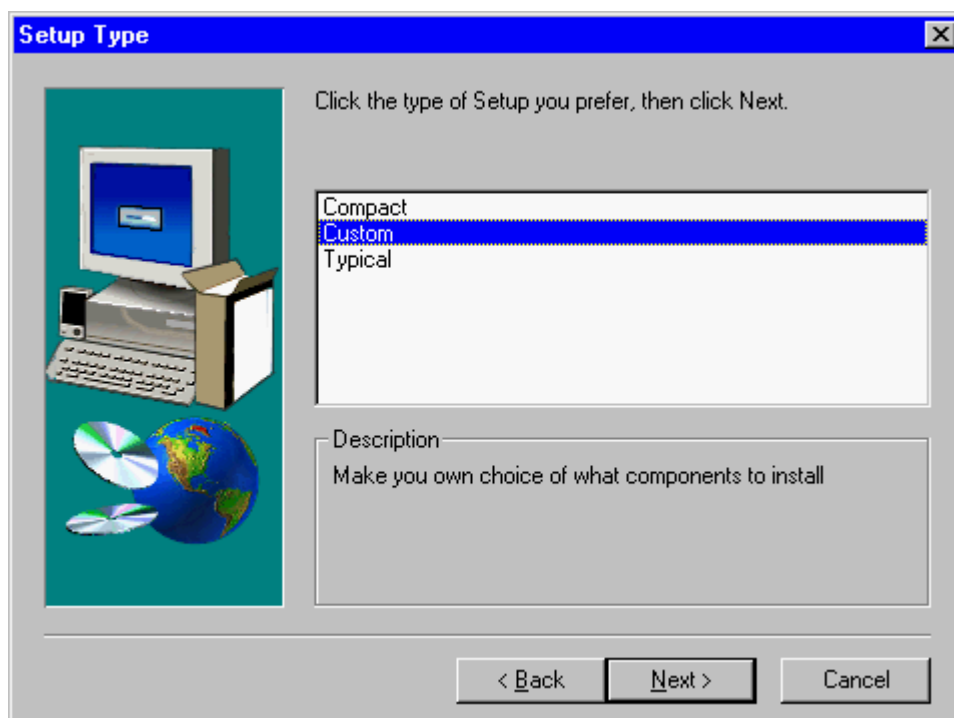
The Dynamic Link Library CMREMOTE32.DLL is not present in the Windows System32 directory. Re-install the 32-bits Windows Remote Control software.

4 INSTALLATION OF THE SOFTWARE

4.1 INSTALLING SOFTWARE

Installation of the Windows Remote Control programs should always be done via the Setup program. To start installation, insert the **32-bits Windows Remote CM** CD-ROM in the CD-ROM disk drive. Wait until the CD-ROM splash screen becomes visible (if it does not, start the CDRom_Master program in the root of the CD-ROM). In the splash screen select installation of the 32-bits Windows CM Remote Control software (the other choice is the 32-bits Freeware which is also present on the CD-ROM).

The setup program will allow selection of Compact, Custom or Typical installation. In the case of Custom installation, you select what you want. In the case of Compact installation, the essential program files are installed (SECS2_32 driver, CMREMOTE32.DLL, RAM Address files and CM Monitor). For Typical the Delphi Tutor program and source is added to that.



In general, you should never deselect the Program Files, Shared Dll's or Support Files in the Custom setup. If you do, and they have not been installed previously, 32-bits CM Remote Control will not work.

Windows CM Monitor is a program that allows saving alignments, stigmator settings and mode settings from the microscope onto disk and restoring them to the microscope.

The Tutor programs' functionality is identical to each other, their only difference being the fact that they were written in different programming languages. It may be useful to install one of these to allow use of the Remote Control functions on the microscope. Unless you

intend to do your own programming, it is not necessary to install the source codes (Delphi, Visual Basic, Help). If only the Tutor program itself is installed, then it is best to install only the Delphi Tutor since it is faster than Visual Basic (noticeable in the screen graphics).

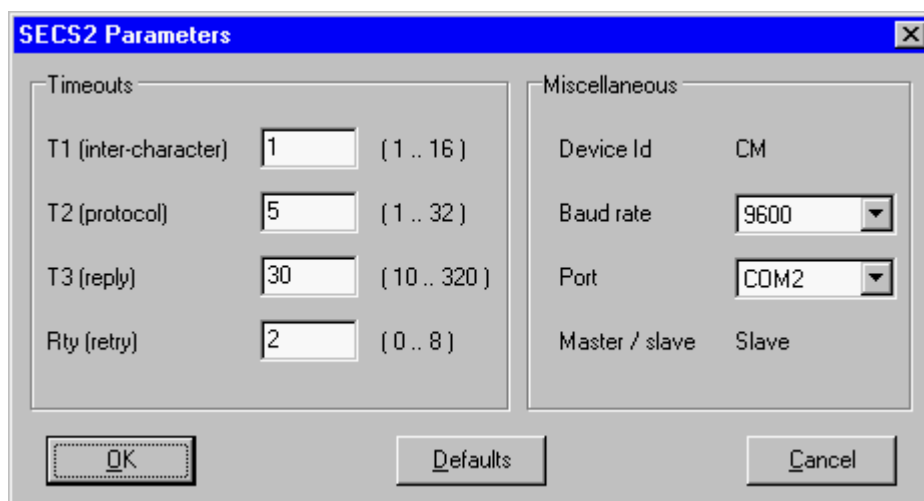
For a more detailed description of the Basic package (CM Monitor) and Tutor packages, see this manual.

4.2 FIRST-TIME OPERATION WITH SECS2_32.EXE

The first time Windows Remote Control is started on a PC, SECS2_32.EXE will open with a dialog in which the settings for communication (COM port, baud rate, etc.) can be set. Under normal circumstances only the COM port needs to be decided upon (if different from COM1:). Otherwise click on OK. The SECS2_32.EXE settings can be modified by double-clicking the SECS2_32.EXE icon. The SECS2_32.EXE program will open its window with menu. Under **Setup, Parameters** one can access the dialog with the communication parameters again.

Press **Defaults** to set the communication parameters to the default values.

Note: From microscope software version 12 onwards it is possible to use 19200 as baud rate as well as any lower value (default of microscope and SECS2_32.EXE is always baud rate 9600).



5 STRUCTURE OF WINDOWS REMOTE CONTROL

Remote Control of the CM microscope under Windows consists of three elements:

- the user program (of which any number can be active at any particular moment)
- the CMREMOTE32.DLL Dynamic Link Library, which controls SECS2_32.EXE
- SECS2_32.EXE, which takes care of communication with the CM microscope via the serial communication COM port on the PC.

SECS2_32.EXE will be loaded automatically with the first Windows Remote Control program started. It is usually running minimized but can be opened by double-clicking on its taskbar icon. The CMREMOTE32.DLL loaded automatically with each program (one difference between 16-bits and 32-bits platforms is that in the former only one instance of dynamic link libraries is running, while in the latter each program has its own instance in its own memory space).

All Remote Control programs should delegate communication via the COM port to SECS2_32.EXE by using the CMREMOTE32.DLL Dynamic Link Library. Although it is possible to write and use your own program for taking care of the COM port, doing so may interfere with the regular Remote Control programs for Windows and is strongly discouraged.

Note: It is not possible to mix MS-DOS Remote Control programs, 16-bits Windows Remote Control programs and 32-bits Windows CM Remote Control, because one COM port cannot be shared between several drivers. You can, however, run 32-bits programs, close the COM port connection in the SECS2_32 program (in the menu under Setup) and, provided none of the 32-bits programs regularly communicates automatically with the CM and re-opens the connection, and then run a 16-bits program without having to close the 32-bits software altogether. To allow the 32-bits software to reconnect you will, however, have to close the 16-bits software including SECS2.EXE (the 16-bits driver).

6 WINDOWS CM MONITOR

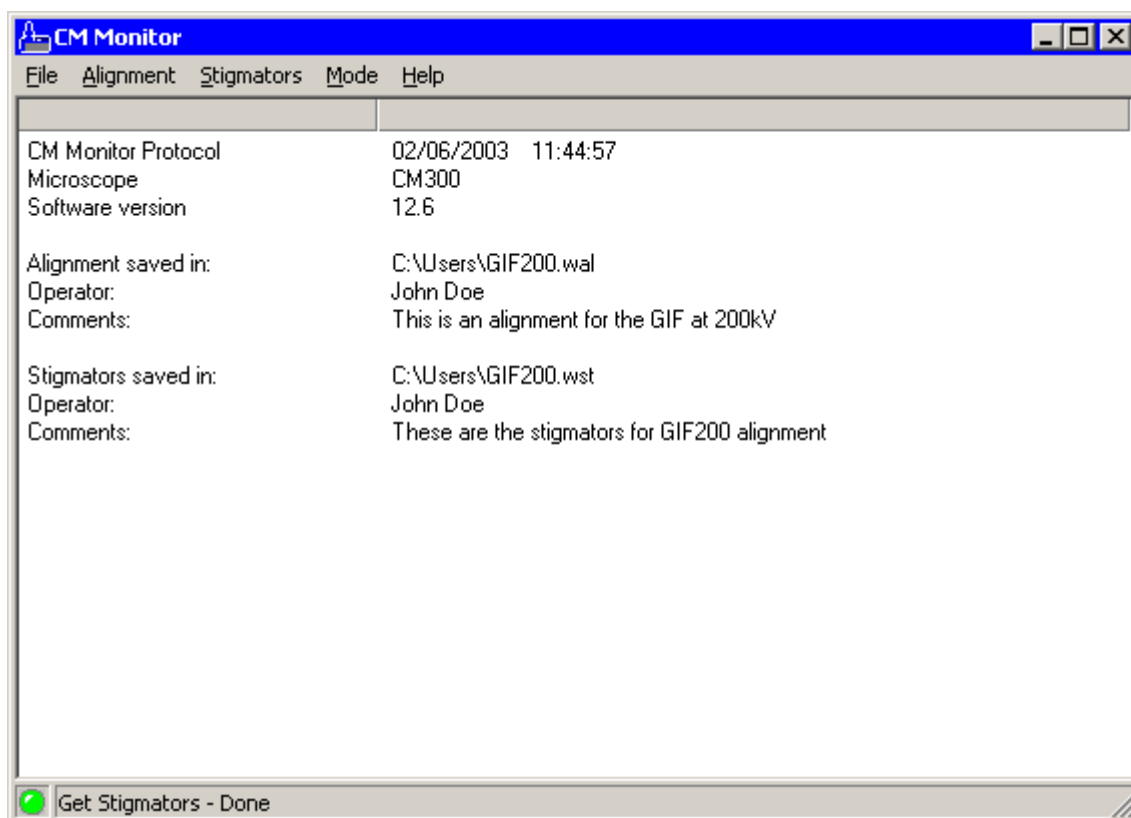
The CM Monitor program makes it possible to:

- retrieve alignments, stigmator settings and mode settings from the CM microscope and save them in a disk file
- restore these settings from file back to the CM microscope.

The alignment contains all current alignments (so for all modes together) and calibrations with the exception of the gun alignment. The gun alignment is excluded because it is expected that microscope alignments remain relevant over a much longer period than gun alignments (which change over time and especially when a filament exchange has occurred).

Note: The alignment, stigmator and mode settings files written by Windows CM Monitor as well as the Tutor program are read-only files to reduce the chance of anyone accidentally changing them and overwriting them in the wrong format.

While active, CM Monitor writes a protocol of the actions performed in its window. The protocol can be stored in a file (**File, **S**ave protocol**) or printed (**File, **P**rint Protocol**), for later reference.



The menu's **Alignment**, **Stigmators** and **Mode** lead to **Save on disk** or **Restore to CM**. In the case of the stigmators a choice can be made to restore only one type of stigmator or all together.

With each file are stored an operator name (20 characters) and comments (60 characters), making it easy to identify whose file it is and which conditions it refers to. In addition, an identifier is stored at the end of the file. If this identifier does not occur at its correct position (for example when someone accidentally has opened the file in a wordprocessing package and then saved it), the file is considered corrupt and the settings will not be restored to the microscope.

Note: The file format of Windows CM Monitor is not compatible with any of the MS-DOS formats used previously (MS-DOS Cmonitor or Tutor) or with the 16-bits CM Monitor or Tutor files. The Windows CM Monitor files have the extension ".wal" (windows alignment), ".wst" (windows stigmators) and ".wmd" (windows mode).

When Saving or Restoring settings on disk, the program will ask for a file name via the standard Windows File dialog. When Restoring files, the program will read the selected file and determine whether the identifier is present. If so, the program will then display the stored operator name and comments and the operator can choose to continue (restoring the settings to the microscope) or Cancel.

Note: The source code of CM Monitor is not part of the Remote Control package, because a large part of its functionality (Save and Restore alignments, stigmators and mode; the file dialog) is already incorporated in the Delphi and Visual Basic Tutor programs, for which the source code is included.

7 WINDOWS TUTOR PROGRAMS

The tutor programs DelphiTutor.exe (Delphi 4.0) and VBTutor32.exe (Visual Basic 6.0) have the same functionality, their main difference being the language in which they were written. Both can be used to control the microscope as well as be used as tools to aid in writing Windows Remote Control programs. Any part of the Tutor programs source code can be copied, modified and incorporated into your own programs.

The Tutor programs are also useful in outlining the type of remote control functions available as well in aiding in finding out what values should be used for certain controls. For example, you may wish to establish what a useful change in the Shift X control would be, which can be easily seen on the microscope by setting a certain value using the Tutor programs.

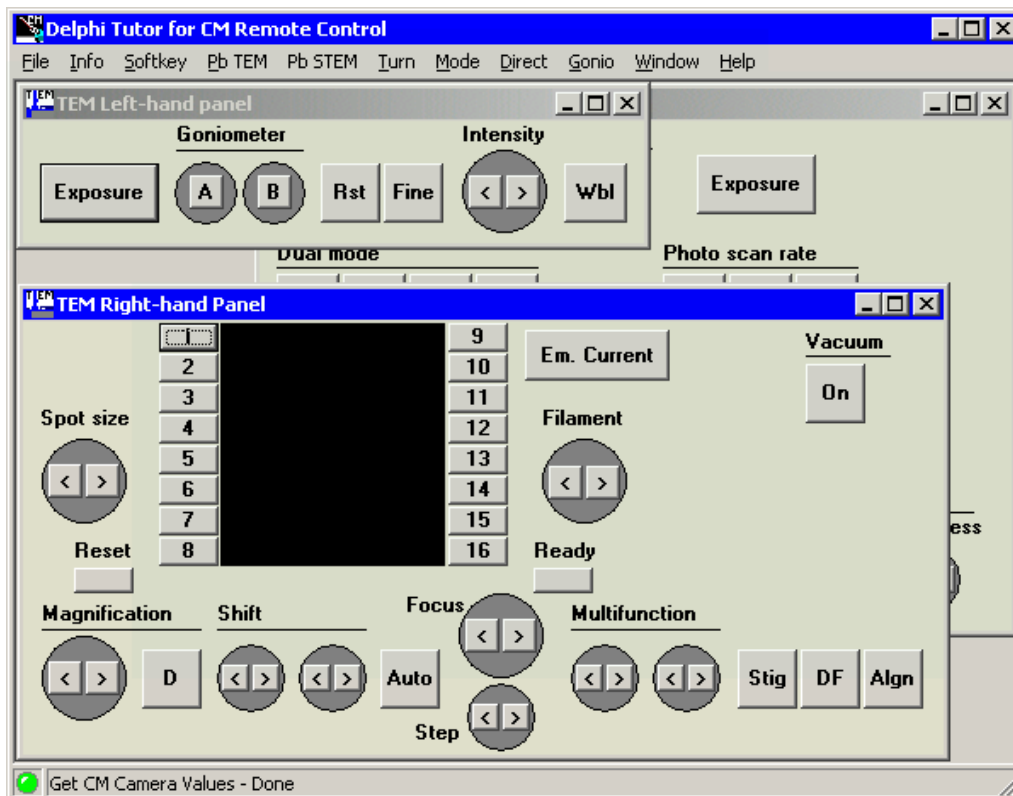
The Tutor programs contain all remote control functions. In a few cases, it is shown how non-existent functions can be simulated by a suitable combination of functions. As an example, the Free Lens Control mode cannot be set using the Instrument Mode remote control function (since it isn't a proper mode). Instead, it is simulated by Getting and the Setting the current Mode (this ensures we are on the mode's operational page), pressing the MODES softkey once, then pressing the FREE LENS CONTROL softkey once.

At startup the Tutor program will try to establish contact with the CM microscope. Once contact is found, the Tutor program is configured according to the type of microscope it is connected to. For example, functions that are usable only when a CompuStage or Field Emission Gun occurs on the microscope are not accessible when the instrument has neither. The program also tests for the presence of scanning. If this is not found, the STEM panel (see below) will not be shown.

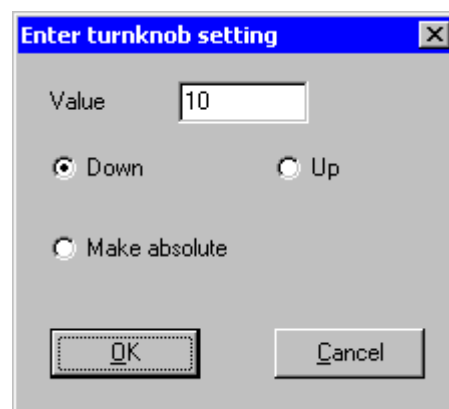
7.1 USING THE TUTOR PROGRAMS

7.1.1 Tutor windows

Once communication has been established, the program's main window will show which can have up to 3 child windows. The three child windows represent the control panels of the CM microscope. Pressing any of the buttons will do the same as pressing the same button on the microscope.

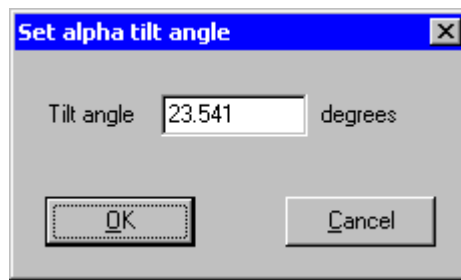


For the turn knobs two small buttons with < and > are displayed. Pressing one of these will turn the knob down (<) or up (>). If sensible, the turn knob will make single step if controlled this way (e.g. spot size or filament). For other turn knobs, a dialog is shown first where the size of the step can be entered:



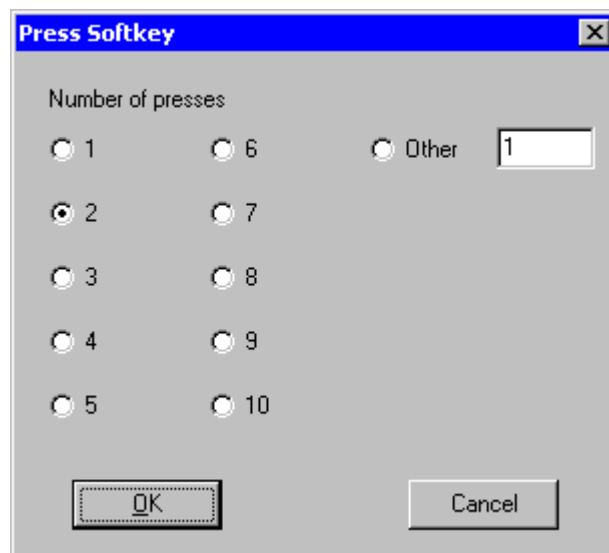
If the down or up buttons were used, the Down or Up radio button is already selected (but can still be changed). Where relevant, it is also possible to choose to make the setting absolute (the turn knob will then be turned till 0, then increased to the desired value).

For the alpha and beta tilt of the CompuStage, it is possible to press one of the buttons and the following dialog will appear:



Entering the tilt angle and pressing OK will result in the CompuStage going to the desired tilt angle.

For the softkeys a choice can be made in the Options (under the **Softkey** menu) to ask if softkeys must be pressed more than one time. Default is a single press. If the options multiple has been switched on, the following dialog will appear when a softkey is pressed:



In this case, 2 has been selected, resulting in the softkey being pressed twice.

7.1.2 Tutor Menu

The Tutor menu gives access to all remote control functions (thereby duplicating also the buttons on the simulated CM control panels). The menu is organised by functionality. The pushbuttons occupy two menu lists, one for the TEM, the other for STEM, partly to be able to separate easily between STEM and non-STEM systems, partly because there are so many that the pull-down menus would become overly large.

7.1.3 File menu

The File menu gives access to those remote control functions associated with using files:

- getting alignments from the microscope and restoring them
- getting mode settings from the microscope and restoring them
- getting stigmator settings from the microscope and restoring them

This functionality, as well as the file format used, is identical to that of CM Monitor. For further details, see chapter 5.

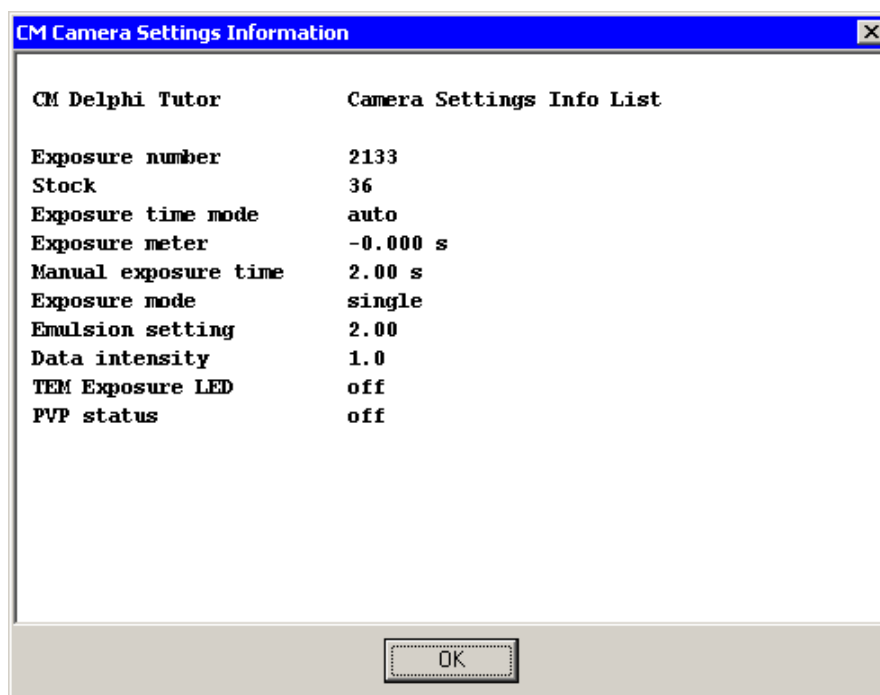
In addition the File menu contains a number of “genera” functions.

7.1.4 Info menu

The Info menu gives access to the various functions that provide information about the remote control and microscope status.

The function **Communication** checks whether remote control has access to the microscope and reports the microscope type and CM software version number. This function is also used at the program start-up to check whether there is communication with the microscope.

After Communication follow a series of items that list system information in the Info List:

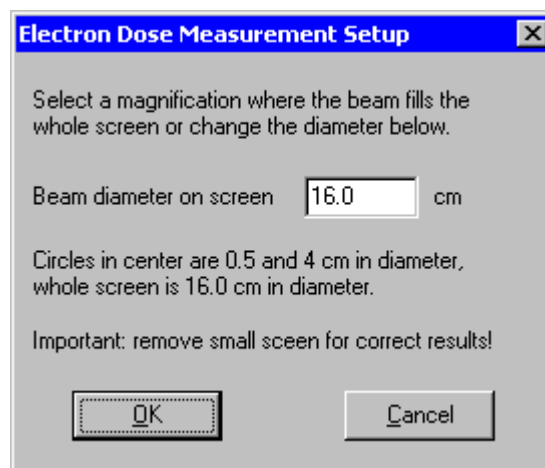


The type of information that can be retrieved falls into the following categories:

- cm info : CM microscope type, Objective lens type, Diffraction lens type, Gun type, magnification, for the CompuStage Holder inserted (yes/no), Goniometer type, whether the CompuStage is enabled, Joysticks are on/off, and the Axes available (X,Y,Z,A,B)
- cm variables : number of currently active CM microscope page, status of the pushbuttons (on/off; only those pushbuttons that have the on/off functionality, see below), magnification, high tension, measuring D1, D2 and angle if displayed on CM data monitor, spot size value (nm), intensity setting, value of beam shift X for low dose SEARCH state (relative to EXPOSURE state), beam shift Y value, image shift X value, image shift Y value, position of main viewing screen (down/up), setting of dark field mode, position of TV detector (central/off-axis/not selected), measuring values, beam shift settings, defocus display, focus step size, objective lens settings and dark-field tilts.
- FEG data : status, time, settings such as extraction voltage, limit, gun lens voltage,
- IGP3.
- camera settings such as exposure time, time mode, exposure mode, and exposure stock.
- lens, deflection coils, stigmator currents (same as displayed on the DISPLAY CURRENTS page of the CM microscope).

Other variables such as vacuum pressures, screen current, electron dose (screen converted to dose on the specimen), emission current, the HT tension status (= whether it can be switched on), whether EDX protection is on or off (a function that, if available, can be found on the microscope's VACUUM page), whether External XY deflection is on or off (the function that controls whether an external system, such as an EDX analyser, controls the beam deflection coils during scanning, found under the EXTERN softkey on the PARAMETERS 2 page), and the video signals on the left- and right-hand STEM monitors.

To determine the dose, the program will ask for the size of the electron beam as seen on the viewing screen:



After this value has been entered, the program reads the screen current and magnification and converts all to the electron dose on the specimen.

7.1.5 Softkey menu

The softkey menu presses the softkey selected one or more times (depending on the choice for single or multiple as explained above). It is identical to pressing one of the buttons on the TEM right-hand control panel window.

7.1.6 Pushbutton menu's

The pushbutton pull-down menu's, one for TEM, the other for STEM, allow pressing the pushbutton on the microscope. Some pushbuttons, such as Intensity RST and Autofocus, have only the "press" functionality. In that case, a selection is possible directly in the pull-down menu. Other pushbuttons also have the functionality of being pressed "on" or "off". For these pushbuttons a small arrow on the right-hand side points to a submenu where the selection between Press, On and Off can be made.

In special cases, other options are listed in a sub-menu. For example, the TEM exposure button is of the "press" only type, but on the microscope it has two functions: making the exposure when the main screen is up and normalising the imaging lenses when the screen is down. For the TEM exposure button three options are shown in the sub-menu: Press, TEM Exposure (this will work only if the main screen is up) and Normalise. The latter function will work normally even when the main is lifted (on the microscope this is not possible but via the remote control it is a separate function).

7.1.7 Turnknob menu

The Turnknob menu allows turning the knobs on the microscope. The main difference with pressing the < > buttons on the panels is that through menu selection the turnknob dialog (see above) always comes up, allowing setting of the steps, direction and, if applicable, whether the setting must become absolute or not. Some turnknobs have no absolute positions (e.g. Shift X). Their lower limit is not at zero but at an undefined value which can vary from situation to situation. For programming, it should be noted that the number of turns on some knobs such as the Shift X does not give an absolute shift but one that is dependent on the microscope status. The Shift controls are implemented in such a way that the shift appears roughly constant at all magnifications, which means that physically the step sizes become smaller as the magnification is increased.

7.1.8 Mode menu

The mode menu allows switching the microscope directly to one of the operating modes. Some of these functions are identical to those in the toolbar. If the program finds that a mode is not available, it will remove the listing from the pull-down menu.

7.1.9 Direct operation menu

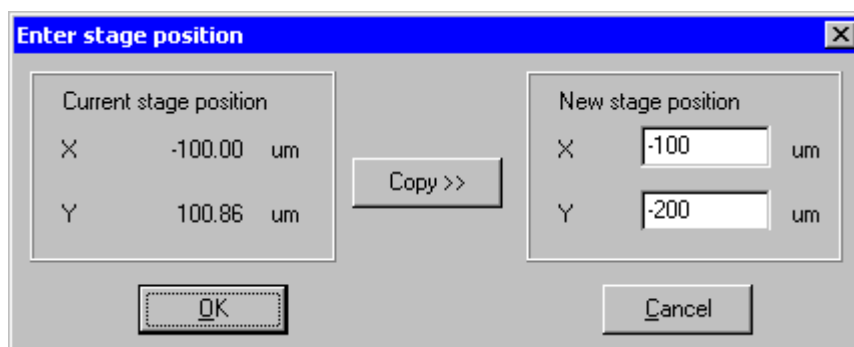
The direct operation menu contains a series of functions that switch the microscope to a certain absolute setting, e.g. the magnification, camera length, spot size, focus step size, etc. High tension can be switched to one of the preset values (as seen on the PARAMETERS page), the Free high tension control can be switched on and off (if available; on older software versions it is not standard; it is always there on CM software version 11 or higher) and the Free high tension can be changed by a specified value. Other functions switch on/off the beam blanker (the beam is deflected in the electron gun so it cannot reach the specimen), EDX protection and External XY deflection.

7.1.10 Goniometer menu

The Goniometer menu provides control of the goniometer on the microscope, provided it is either the manual stage with the Specimen Relocation System (motorised X,Y called SRS) or the CompuStage. For a microscope equipped with the manual stage only, the Goniometer menu will not appear. The functions shown in the menu are dependent on the type of stage. The SRS functions Read position and Goto position work for the CompuStage (but display or set of course only X and Y). The SRS registers functions also work for the CompuStage but are not listed since control via the CompuStage registers is more convenient. SRS Calibrate should not be used for the CompuStage. None of the CompuStage functions work with the SRS.

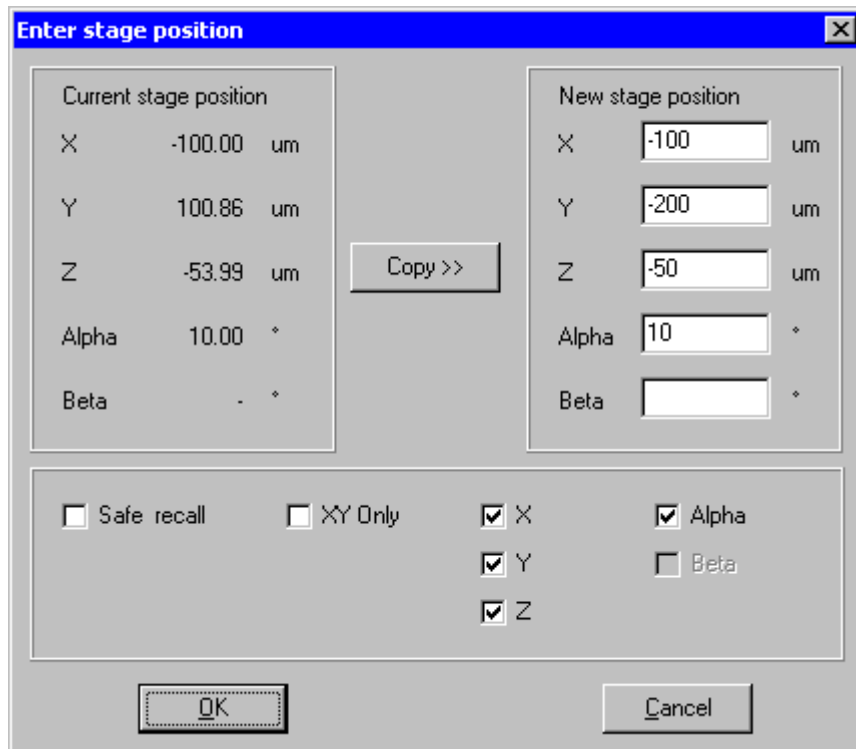
The functions for the goniometers allow reading the current position, telling the stage to go to a certain position and manipulating the registers in which the positions are stored.

The Go to position functions display a dialog that lists the current position which can then be copied into the new position (this is useful if only one of the values must be changed). Otherwise enter both X and Y values in case of the SRS.



For the CompuStage there exist a number of ways in which the Goto can be performed. The two ways normally used by the microscope are referred to as recall and XY only. In the former case all axes are set, in the latter only X and Y are changed. Both procedures follow the sequence β to zero, α to zero, move on X,Y,Z, α to new angle, β to new angle. These methods can be selected on the dialog. In addition it is possible to move on any single or combination of all five axes, either following the recall procedure or moving directly (recall off).

Warning: Go to's with recall off should only be done while tilting on α and/or β within safe angles as there is a risk of damage otherwise.



The 'Enter stage position' dialog box is used to set the stage coordinates. It features two main sections: 'Current stage position' and 'New stage position'. The 'Current stage position' section displays the following values: X: -100.00 um, Y: 100.86 um, Z: -53.99 um, Alpha: 10.00 °, and Beta: - °. The 'New stage position' section has input fields for X (-100 um), Y (-200 um), Z (-50 um), Alpha (10 °), and Beta (°). A 'Copy >>' button is located between the two sections. At the bottom, there are checkboxes for 'Safe recall' (unchecked), 'XY Only' (unchecked), and individual axis selection: X (checked), Y (checked), Z (checked), Alpha (checked), and Beta (unchecked). 'OK' and 'Cancel' buttons are at the bottom right.

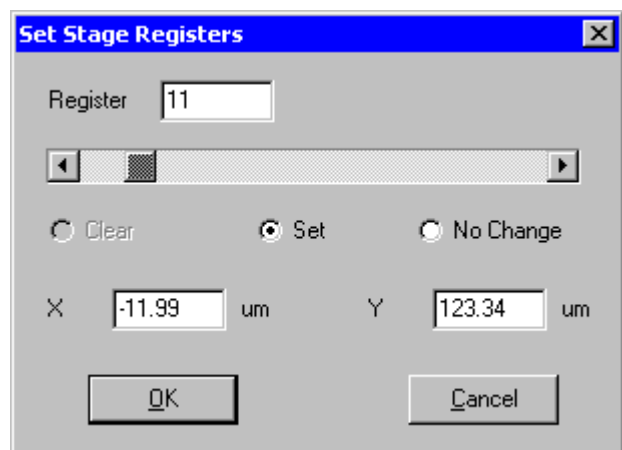
Select a method for the Go to using the check boxes and fill in or copy the appropriate axes:

- all for recall
- X,Y for XY only
- individual axes selected, with our without recall

For manipulating registers, there are some differences between the SRS and CompuStage functions. SRS registers (99 with 2 axes) are manipulated one by one. The Tutor programs start off by reading the contents of all registers, while displaying a window that shows the progress. The reason for reading all registers first is to make it easier to manipulate the registers themselves in one of the following dialogs:



The 'Select Stage Register' dialog box allows users to choose a register. It includes a 'Register' input field with the value 26, a horizontal scroll bar, and three radio buttons: 'Clear' (unchecked), 'Set' (unchecked), and 'No Change' (checked). Below these are labels for X and Y axes, each followed by 'um'. 'OK' and 'Cancel' buttons are at the bottom.



The 'Set Stage Registers' dialog box is used to set specific register values. It features a 'Register' input field with the value 11, a horizontal scroll bar, and three radio buttons: 'Clear' (unchecked), 'Set' (checked), and 'No Change' (unchecked). Below these are input fields for X (-11.99 um) and Y (123.34 um). 'OK' and 'Cancel' buttons are at the bottom.

By moving the scroll bar or changing the register number, the displayed register contents will change, becoming those of the register selected (showing nothing if the register is

empty). The four functions available are Read register, Select register (the register selected becomes highlighted on the CM microscope page), Goto register (the stage moves to the register location) and Set register. In the latter case (with the dialog shown on the right-hand side), the displayed values are in edit controls and can be changed. Once the value for the register (and values for X,Y entered), pressing OK will execute the function **for that particular register!**

There are no functions for clearing individual registers or all together. In order to do that, the proper sequence of softkeys must be pressed (go to XY CONTROL page, then press FORGET ALL or select register and press FORGET XY).

The CompuStage registers (25 with 5 axes) are all manipulated together. When reading, the registers, the program will display the Info list with the contents of the registers, stating either cleared (empty) or filled with the X,Y,Z,A,B location. When setting the contents of the registers, the program displays the following dialog:



By scrolling or changing the register number the contents of the registers are listed. Initially all registers have the No change selection (radio button) active. This means that even when exited via OK, no change will be made to the contents of the registers. In order to change a register, either select Clear or select Set. In the latter case, the X,Y,Z,A,B must also be entered. Changes can/must be made for registers individually (but while in the dialog; changes made are remembered). Once done for all registers desired, press OK and the registers of the CM will be updated with the new values.

7.1.11 Window menu

The Window menu controls the position of the three panel windows shown. Normal Arrangement brings them back to the situation at program start-up, that is, the windows' size optimised to the area covered by their controls and the positions with the TEM left-hand panel top left, right-hand panel bottom left and STEM panel top right. Note that these Windows can be sized, minimised and maximised but not closed.

Selecting one of the panel windows opens it (if it was minimised) and makes it the active, frontmost one.

7.1.12 Help menu

Two Help files come with the Tutors, one for help on operating the program, the other for help in programming (the Guide).

7.2 PROGRAMMING WITH TUTOR

7.2.1 Introduction

For programming your own Windows applications, the first step is to decide which language to use (if you haven't already decided). Both Visual Basic and Delphi are easy-to-use languages and have therefore been chosen to be used for the Tutor programs.

7.2.2 Getting started

As any experienced programmer will tell you, the first thing you have to do when making a program is to think well on what you want to achieve and how you are going to achieve it. Once you have a good idea about that, then the next step generally is to build the user interface of the program. The real execution code is usually the step after that, since it more or less forms the link between the interface elements, with some processing code thrown in for good measure. The final step should be making the Help file, especially if you are thinking about distributing the program.

In general, when you start off with a new programming language, it pays off to go through at least part of instruction manuals (the teaching parts, not the reference manuals) and to work your way through at least some of application examples provided with both languages. It may also be helpful to work through the code of the Tutor program(s) and see which functions are used and what purpose they serve (apart from the Remote Control functions).

As you will probably find, the Remote Control is the easy part of the programming, while it is the user interface that usually requires more time.

Because the Tutor programs for Remote CM are intended as examples of writing Windows programs for Remote Control, the following principles were adhered to:

- the programs should contain as many Remote Control functions as possible.
- the programs should have a wide variety of types of Windows controls, interface elements and functions.
- simplicity, clarity and workability of the code are of more importance than elegance, consistency and compactness.

As you go along, making your own programs, you may find easier, more logical or better ways of doing some things. If you do, do not hesitate to inform us and make your suggestions to the address listed in the Preface of this manual.

7.3 DELPHI

Probably the most important file coming with the Delphi Tutor is RCMWIN32.PAS. This unit forms the interface to the Remote Control Dynamic Link Library. All constants like pushbutton and turnknob identifications are listed here. So in calling e.g. the Pushbutton

function for the Diffraction button you simply use the PbDiffraction constant to identify the button. In addition, the calls to the Dynamic Link Library are listed. The only things you have to do for using the RCMWIN32 file is put it in the `uses` clause at the beginning of the program (also units if they use remote control functions).

In the CM Delphi Tutor program the Dynamic Link Library is loaded implicitly (loaded automatically when the program is started). This makes sense because the remote control functions will be used always and more or less continuously. However, for programs that will only under some circumstances make use of the remote control (and where the Dynamic Link Library may be absent - in which case the program will refuse to load with the message that Windows cannot locate the Dynamic Link Library), it may make more sense to load the Dynamic Link Library explicitly after checking for its presence. The following code fragment shows how explicit linking is done. Do not use the RCMWIN32.PAS file in this case (but take from it the required variable declarations and function structures).

Declaration section:

```
var
  RcmLibrary : THandle;
  winsysdir  : array[0..255] of char;
  i          : integer;

  EquipmentAvailable : function (hwindow: THandle; var szModel,szVersion:
    chartype): integer; far;
```

Code section for loading library:

```
i := 255;
GetSystemDirectory(winsysdir,i);
RcmLibrary:= LoadLibrary(pchar(strpas(winsysdir)+'\'+RCMDLLname));
if RcmLibrary <> 0 then
begin
  @EquipmentAvailable :=
    GetProcAddress(RcmLibrary,'EquipmentAvailable');

  ...
end;
if @EquipmentAvailable = nil then
  Application.MessageBox('Cannot access remote control library',
    '.....', mb_OK);
```

After loading, the functions can be used as in CM Delphi Tutor (so without the @ in the function name). Before exiting the program use `FreeLibrary(RcmLibrary)` to unload the library again.

7.3.1 Program structure

Delphi Tutor is structured around a parent MDI (Multiple Document Interface) window with three MDI Child windows, named TEM left-hand panel, TEM right-hand panel and STEM panel. Each of the three child windows contains a series of buttons whose functionality, position and appearance corresponds to the push buttons, softkeys and turn knobs of the CM microscope.

The choice for the MDI window was made because it provides an easy way of handling additional windows: they stay within the confines of the parent window, the parent window has scrollers when it is made too small to show all children, etc. Note that the MDI window is not suitable if only a single window is to be used with graphics or buttons displayed on it, since one cannot display any graphics on an MDI parent window.

The code for the MDI parent window is sequenced according to the menu, starting with some general code (FormCreate, FormCloseQuery, etc), followed by the File menu items, the Info menu items, the Softkeys, ... up to the Help menu items.

The code for the child windows follows the order of appearance of their buttons, from left to right. In many cases the code for the child windows simply refers to a procedure of the parent window, in some cases with an additional parameter passed. The child windows are made in such a way that they cannot be closed (since there is no code for the user to open them again).

The child windows are listed under the Window menu.

7.3.2 Program file RCMWIN32.PAS

All remote control functions are found in the RCMWIN32.PAS file which interfaces with the Remote CM Dynamic Link Library (CMREMOTE32.DLL). RCMWIN32.PAS can be incorporated simply into other programs.

7.3.3 Remote control functions

The following Remote Control functions are available, listed in alphabetical order. All return the rcmresult error message.

AbortSecsAction	ChangeFreeHt
CloseConnection	
CSAskPos	CSGetRegs
CSGotoPos	CSSetRegs
CurrentReadout	DirectOperation
EmissionCurrent	EnableUserInput
EquipmentAvailable	GetAlignment
GetCmCamValues	GetCmInfo
GetCmVar	GetFegData
GetHTAcknowledge	GetMode
GetRotationAlignment	GetStigmators
InstrumentMode	PressureReadout
Pushbutton	ResetDisplay
RetrEdxProt	RetrExtXY
RetrHTcond	ScreenCurrent
SetAlignment	SetMode
SetC2Stigmator	SetDifStigmator
SetObjStigmator	SetRotationAlignment
SetTimeOutTime	SetTVdetPos
Softkey	SRSCalibrate
SRSGotoPos	SRSGotoReg
SRSReadPos	SRSReadReg
SRSSelectReg	SRSSetReg
SwitchFreeHt	TurnKnob
Video Left	Video Right

Note: Relative to the DOS version of the Remote Control, many of the data types used are converted to simpler units, only the integer, single and char being used. Originally in the 16-bit version this was because Visual Basic didn't have data types such as bytes and words. Translation takes place in the Remote Control Dynamic Link Library CMREMOTE32.DLL.

In the DOS version of the remote control a few functions additional to the ones for Windows existed. These functions have been left out of the Windows remote control because they could cause severe problems. They are related to logging (getting data from the microscope as defined on the PRINTING PARAMETERS page). Logging is the only remote control function where the microscope takes the initiative for sending messages. With multi-program environments this has the risk that one program is asking information more or less continuously (e.g., stage position) while the microscope then answers with the logging data. These do not fit the answer expected and may cause program crashes or hang-up of the remote control function on the microscope.

Function AbortSecsAction

```
Function AbortSecsAction (hwindow: THandle): integer;
```

This function aborts waiting for a response from SECS2_32.EXE.

Function ChangeFreeHT

```
Function ChangeFreeHT(deltaht: single): integer;
```

This function changes the high tension by the deltaht value specified. Free HT must be on for this function to work.

This function has been implemented from microscope software version 11 onwards.

Procedure CloseConnection

This procedure (the only non-function among the remote control) closes the COM interface between the CMREMOTE32.DLL dynamic link library and the SECS2_32.EXE driver. It should be called as the last remote control call before the program closes.

Function CSAskPos

```
Function CSAskPos(hwindow: THandle; var x,y,z,a,b: single): integer;
```

CSAskPos reads the current x,y,z,a,b position of the CompuStage. Note that the x,y,z values are in micrometers, the tilt angles in degrees.

This function has been implemented from microscope software version 10 onwards, but beta tilt read-out only works correctly since version 11.

Function CSGetRegs

```
Function CSGetRegs(hwindow: THandle; var csregisters: TCSRegisters): integer;
```

CSGetRegs retrieves the x,y,z,a,b positions of all 25 registers of the CompuStage. csregisters consists of an array of 25x an integer filled and a CSPos (an array of 5 singles).

Filled is 0 when the register is empty, 1 when it is filled.

This function has been implemented from microscope software version 10 onwards.

Function CSGotoPos;

```
Function CSGotoPos(hwindow: THandle; x,y,z,a,b: single; m: integer): integer;
```

CSGotoPos moves the CompuStage to the x,y,z,a,b position specified. Note that the x,y,z values are in micrometers, the a and b values in degrees. m identifies the method of stage movement.

Method identifiers:

Recall	=	\$40	(decimal 64) { full 5 axes safe recall }
XYonly	=	\$C0	(decimal 192) { xy recall }
Xgoto	=	\$1	
Ygoto	=	\$2	
Zgoto	=	\$4	
Agoto	=	\$8	
Bgoto	=	\$10	(decimal 16)

Examples:

safe goto all axes	:	use method RECALL;
direct goto axes Z and B	:	use method (Zgoto or Bgoto)

Note: although it is possible to combine RECALL and Xgoto, etc... the microscope still treats the method as RECALL only. In order to prevent resetting the axes that are not selected, first Ask the current position and leave the axes in the Goto call.

This function has been implemented from microscope software version 10 onwards.

Function CSSetRegs

```
Function CSSetRegs(hwindow: THandle; csregisters: TCSRegisters): integer;
```

CSSetRegs sets the x,y,z,a,b positions of all 25 registers of the CompuStage. csregisters consists of an array of 25x an integer filled and a CSPos (an array of 5 singles).

Filled is 0 when the register is to be cleared, 1 when it is to be filled with the values of CSPos, 2 if the contents of the register are to remain unchanged.

This function has been implemented from microscope software version 10 onwards.

Function CurrentReadout

```
Function CurrentReadout(hwindow: THandle; var currents: TCurrents; currentid: integer): integer;
```

This function reads the currents of the lenses, deflection coils and stigmators on the microscope (equivalent to DISPLAY CURRENTS on the PARAMETERS page). Currents

is an array of 26 singles. Currentid identifies which current should be read out. If currentid is -1 all 26 currents are read out. If currentid is in the range 0..25 then the value of the requested current is put into the currents[1].

Function DirectOperation

```
Function DirectOperation(hwindow: THandle; diroperation,param: integer):  
integer;
```

Direct operation (constants have been predefined in RCMWIN32.PAS) contains the following functionality with the diroperation:

- | | | |
|----|-------------------|---|
| 1 | DirOpDmode | set D mode, Param is the number of camera lengths from the smallest one |
| 2 | DirOpFocusStep | set Focus step size, Param is the step size |
| 3 | DirOpHMag | set HM/SA magnification, Param is the magnification step from the smallest (LM !) one |
| 4 | DirOpHTStep | set HT step, Param is the number of steps from the lowest HT setting |
| 5 | DirOpLAD | set LAD mode, Param is the number of steps from the smallest LAD camera length |
| 6 | DirOpLMag | set LM magnification, Param is the magnification step from the smallest one |
| 7 | DirOpPressReady | Press Ready button |
| 8 | DirOpSpotSize | set Spot size, Param is the spot size |
| 9 | DirOpHTmax | set High Tension to maximum, Param has no function |
| 10 | DirOpBBlon | switch beam blanker on, Param has no function |
| 11 | DirOpBBloff | switch beam blanker off, Param has no function |
| 12 | DirOpEDXproton | switch EDX protection on, Param has no function |
| 13 | DirOpEDXprotoff | switch EDX protection off, Param has no function |
| 14 | DirOpExtXYdeflon | switch External XY deflection on, Param has no function |
| 15 | DirOpExtXYdefloff | switch External XY deflection off, Param has no function |
| 16 | DirOpNormalise | normalise imaging lenses (= pressing TEM Exposure with main screen down but this operation can be performed with screen up) |

Direct Operation 16 has been from microscope software version 11 onwards.

Note: Neither setting HM/SA or LM magnifications switches the microscope to image mode and this function only operates correctly when the microscope is in image mode. The difference between the two direct operations is that the set LM magnification first goes down to the minimum LM magnification and then up to the requested value, while set HM/SA magnification goes up to the maximum and then down.

Function EmissionCurrent

```
Function EmissionCurrent(hwindow: THandle; var emssncurrent: single): integer;
```

Emission current reads the emission current of the microscope in Ampères.

Function EnableUserInput

```
Function EnableUserInput(hwindow: THandle; fEnable: integer): integer;
```

EnableUserInput enables and disables the user interface of the CM microscope (in other words, when disabled the user cannot change anything on the microscope using turnknobs, pushbuttons and softkeys).

Note: When using this function, always make sure to turn it off afterwards.

This function has been implemented from microscope software version 11 onwards.

Function EquipmentAvailable

```
Function EquipmentAvailable(hwindow: THandle; var szModel,szVersion: TChar): integer;
```

Function EquipmentAvailable establishes whether communication is possible and returns the type of microscope (CM10, CM12, CM20, CM30, CM100, CM120, CM200, CM300) and the CM software version number (93512 for version 1.2, 00021 for version 2.1, 00030 for 3.0, etc.). The '935' at the beginning at not been used from version 2.1 onwards.

In case there is doubt about the operation of a user-written program, the EquipmentAvailable test can also be done directly by SECS2_32.EXE. Open the icon by double-clicking on it and select Window, Are you there? The microscope should respond by giving its type and software version).

Function GetAlignment

```
Function GetAlignment(hwindow: THandle; var alignment: TAlignments; var siz: integer): integer;
```

This function will retrieve the alignments (excluding the gun alignment) from the microscope. It is advised to retrieve and set all alignments as a single block and not change individual values, since the actual structure of the alignments may vary with the CM software version and instrument type. The variable siz gives the actual length of the alignments. It should be stored with the alignment and passed back to the microscope when the alignment is reset.

Function GetCmCamValues

```
Function GetCmCamValues(hwindow: THandle; var camvalues: TCamValues):  
integer;
```

This procedure retrieves various parameters related to the camera.

Camvalues is a record with the following fields carrying information.

Exposure_nr	: array[1..4] of char	exposure number
Stock	: integer	exposure stock
Seriessize	: integer	number of exposures in series
TEMexposureLED	: integer	status of TEM Exposure button LED (odd = on, even = off)
Double	: integer	double exposure active (odd) or off (even)
Expmodeseries	: integer	exposure series active (odd) or off (even)
Seriesthrfocus	: integer	through-focus series active (odd) or fixed-focus (even)
Exptimeauto	: integer	automatic exposure selected if odd
Exptimmanual	: integer	manual exposure time selected if odd
Exptimer	: integer	exposure timer selected if odd
PVPrunning	: integer	Pre-Vacuum Pump is running if odd (cannot take an exposure then)
Measuredexptime	: single	measured exposure time value
Manualsetexptime	: single	manual exposure time value
Emulsionnumb	: single	emulsion setting
Dataintfactor	: single	data intensity setting

This function has been implemented from microscope software version 12 onwards.

Function GetCmInfo

```
Function GetCmInfo(hwindow: THandle; var cminfo: TCmInfo; var Leng:  
integer): integer;
```

This procedure retrieves various system parameters. Leng gives the length of cminfo (13 for CM software version 10 and 11). This value can be used to check for future extensions of the values in cminfo.

CmInfo is a record with the following fields carrying information. Some of the following fields relate only to the CompuStage.

CmType	: integer	0..7: CM10/12/20/300/100/120/200/300
ObjLens	: integer	0..3: High Contrast/TWIN/SuperTWIN/UltraTWIN
DiffLens	: integer	0..1: normal diffraction lens/special diffraction lens
Gun	: integer	0..2: Tungsten/LAB6/FEG
Magn	: single	the magnification as displayed on data monitor
HolderInserted	: integer	true =1
GonioMeter	: integer	0..2: manual/CompuStage/SRS
CompuEnabled	: integer	true when CompuStage enabled

Joysticks	: integer	bitwise: ...bazyx; 1 if enabled
AxesAvail	: integer	bitwise: ...bazyx; 1 if available

Note: For the CM10/CM100 TWIN (objective-lens 1) is the BioTWIN, for the CM12/CM120 High Contrast (objective-lens 0) is the BioTWIN.

This function has been implemented from microscope software version 10 onwards.

Function GetCmVar

```
Function GetCmVar(hwindow: THandle; var cmvar: TCmVar; var Leng: integer): integer;
```

This procedure retrieves various system variables. Leng gives the length of cmvar (54 for CM software version 11, 88 for software version 12). This value can be used to check for future extensions of the values in cmvar.

CmVar is a record with the following fields carrying information:

Page	: longint	the number of the current MICROCONTROLLER page
ButtonState	: longint	the states of pushbuttons, see below
Magn	: single	the current magnification (note: camera length in nm- μ !)
HT	: single	the current high tension
D1	: single	the measuring value D1
D2	: single	the measuring value D2
Angle	: single	the measuring angle
Spotsize	: single	the current spot size (in nm)
Intensity	: single	the current intensity setting (range 1000.. 100000)
BeamShiftX	: single	Low Dose Search state beam shift X
BeamShiftY	: single	Low Dose Search state beam shift Y
ImageShiftX	: single	Low Dose Search state image shift X (-35000 .. 35000)
ImageShiftY	: single	Low Dose Search state image shift Y (-35000 .. 35000)
MainScrn	: integer	main screen up (true) or down (false)
DFmode	: integer	dark field mode, bitwise bit 0: 0 X-Y, 1 Cone; bit 1: 0 manual, 1 dynamic
TVdetpos	: integer	position of the TV (0 central, 2 off-axis, 3 not selected)

From CM software version 12 onwards:

Measx,Measy	: single	two values giving the measuring data expressed as the x,y settings before conversion for the display
Beamx,Beamy	: single	two values giving the beam position for the currently active mode
Defocus	: single	defocus display value (as on MICROCONTROLLER screen) in nanometers
Focusstep	: integer	focus step value (1 to 9)
Objcurrent	: single	objective lens settings, a value between 0 and 100 000
Dftx,Dfty	: single	dark field tilt display values (as x-y or rot-tilt, depending on active dark-field mode)

Buttonstate contains the current status of the pushbutton that have the On,Off,Press functionality. Buttonstate defines bitwise if the following buttons are on (1) or off. Bit definition for 0..15 is

0	Diffraction	1	Stigmator
2	Dark field	3	Alignment
4	Scan Stop	5	Dual Signal
6	Dual Mag	7	Split screen
8	Exchange mode	9	Y mode
10	STEM Exposure	11	Set contr./br.
12	Auto contr./br.	13	Invert video
14	Intensity Fine	15	Wobbler

Note: In order to allow easy distinction between image magnification and camera length, the camera length is given in nanometers (no overlap in ranges).

This function has been implemented from microscope software version 11 onwards, extended in software version 12.

Function GetFegData

```
Function GetFegData(hwindow: THandle; var fegdata: TFegData): integer;
```

This function is retrieves information about the FEG. FegData contains the following:

state	: integer	the FEG state: 1 off, 2 shutdown, 3 IGP3 on, 4 standby, 5 operate, 6 service, 7 cold start, 8 warm start, 9 test, 10 error
time	: integer	the wait time as displayed on the MICROCONTROLLER
FEG pages		
disp	: single	6 singles containing the data displayed on the CONFIGURATION page and the FEG DISPLAY page: Fil, Extr, Gun, Emis, IGP3, Extr limit

Data measurement frequency on the microscope is every 2 seconds.

This function has been implemented from microscope software version 11 onwards.

Function GetHtAcknowledge

```
Function GetHtAcknowledge(hwindow: THandle; var HTcond: integer): integer;
```

This functions determines whether the HT on the microscope is on (odd) or off (even). It thus differs from RetrHTcond which only determines if the HT can be switched on or not.

This function has been implemented from microscope software version 12 onwards.

Function GetMode

```
Function GetMode(hwindow: THandle; var mode: TMode; var siz: integer): integer;
```

This function will retrieve the mode settings from the microscope. It is advised to retrieve and set all modes as a single block and not change individual values, since the actual structure of the modes settings may vary with the CM software version and instrument type. The variable `siz` gives the actual length of the mode settings. It should be stored with the mode settings and passed back to the microscope when the mode is reset.

Function GetRotationAlignment

```
Function GetRotationAlignment(hwindow: THandle; rotal: TRotationAlignment) : integer;
```

This function gets the rotation alignments of the microscope. Together with Set rotation alignment, this function is meant for automated microscope alignment.

This function has been implemented from microscope software version 11 onwards.

Function GetStigmators

```
Function GetStigmators(hwindow: THandle; var stigmators: TStig; var siz integer): integer;
```

This function will retrieve the stigmator settings from the microscope. It is advised to retrieve and set all stigmators settings as a single block and not change individual values, since the actual structure of the settings may vary with the CM software version and instrument type. The variable `siz` gives the actual length of the stigmator settings. It should be stored with the stigmator settings and passed back to the microscope when a stigmator is reset.

Function InstrumentMode

```
Function InstrumentMode(hwindow: THandle; instrmode: integer): integer;
```

This procedure sets the microscope into the requested mode. instrmode's are (constants with Md.. are predefined in RCMWIN32.PAS):

Value	Mode	Constant
1	Microprobe scan	MdMicroprobeScan
2	Nanoprobe scan	MdNanoprobeScan
3	Nanoprobe	MdNanoprobe
4	Post-specimen scan	MdPostSpecimenScan
5	Rocking Beam	MdRockingBeam
6	Scanning	MdScanning
7	TEM Low Dose	MdLowDose
8	(HR-)TEM	MdTem

Function PressureReadout

```
Function PressureReadout(hwindow: THandle; var Pressure: TPressure);
```

This procedure returns the vacuum pressures of P1, P2, P3 and IGP. The values are the same as those on the VACUUM page of the microscope.

Function PushButton

```
Function PushButton(hwindow: THandle; button,presstype: integer):  
integer;
```

Pushbutton presses one of the push buttons on the CM microscope. Some push buttons can have presstype On, Off or Press, other ones just Press. On (0), Off (1) and Press (2) are defined in RCMWIN32.PAS. Buttons that cannot be pushed are HT, Vacuum Off and the main panel On, Standby, Off.

The push button id's (constants with Pb.. predefined in RCMWIN32.PAS) are (bold identifies as On,Off,Press; otherwise Press only):

PbDiffraction	3	PbSplitScreen	43
PbAutofocus	4	PbExchSignal	44
PbReset	6	PbExp1	45
PbReady	7	PbExp2	46
PbStigmator	27	PbExp3	47
PbDarkField	28	PbZmode	48
PbAlignment	29	PbYZmode	49
PbVacuumOn	31	PbYmode	50
PbFullFrame	32	PbStemExposure	51
PbSelectedArea	33	PbSetButton	52
PbCrosshairs	34	PbAutoButton	53
PbLine	35	PbInvertSignal	54
PbScanstop	36	PbExchControl	55
PbSlow	37	PbExposure	60
PbFast	38	PbIntRST	61
PbTV	39	PbIntFine	62
PbDualSignal	41	PbWobbler	63
PbDualMag	42		

Function ResetDisplay

```
Function ResetDisplay(hwindow: THandle): integer;
```

This function forces the CM microscope to update the display on the MICROCONTROLLER. Updates of the display have low priority on the microscope so the remote control may appear to be functioning incorrectly. Use the ResetDisplay function after switching Free HT off (the microscope software does not update the display of the HT value correctly).

This function has been implemented from microscope software version 11 onwards.

Function RetrEDXProt

```
Function RetrEDXProt(hwindow: THandle; var EDXprot: integer): integer;
```

This function retrieves whether the EDX protection function (on the VACUUM page if present) is on (1) or off (0).

Function RetrExtXY

```
Function RetrExtXY(hwindow: THandle; var ExtXY: integer): integer;
```

This function retrieves whether the External XY deflection function (if an External Analog Interface Board is mounted, displayed as EDX detector on the DETECTORS page) is on (1) or off (0).

Function RetrHTCond

```
Function RetrHTCond(hwindow: THandle; var HTcond: integer): integer;
```

This function determines whether the High Tension may be switched on (1) or not (0). The actual switching is not possible through the remote control for safety reasons.

Function ScreenCurrent

```
Function ScreenCurrent(hwindow: THandle; var scrncurrent: single):  
integer;
```

This function measures the electron current on the viewing screen in Ampères. Note that the current read-out is amplified by about 15x when the small screen is inserted. Whether this screen is in or out is not traceable by the CM software (the switching is done via hardware only) and therefore cannot be determined through remote control.

In order to get the real electron current falling on the screen, the value returned by the remote control should be multiplied by 2.5x to compensate for loss of backscattered electrons (measured with a Faraday cage for 200 kV).

Function SetAlignment

```
Function SetAlignment(hwindow: THandle; alignment: TAlignments; siz:  
integer): integer;
```

This function will reset the alignments (excluding the gun alignment) back into the microscope. It is advised to retrieve and set all alignments as a single block and not change individual values, since the actual structure of the alignments will vary with the CM software version and instrument type. The variable siz is the actual length of the

alignments. It should be stored with the alignment and passed back to the microscope when the alignment is reset.

In order to ensure that the alignments are reset properly, the SetAlignment function should be preceded by pressing the Alignment button off and followed by pressing the Wobbler on, Diffraction on, Wobbler off, Diffraction off.

Function SetMode

```
Function SetMode(hwindow: THandle; mode: TMode; siz: integer): integer;
```

This function will set the mode settings back into the microscope. It is advised to retrieve and set all modes as a single block and not change individual values, since the actual structure of the modes settings may vary with the CM software version and instrument type. The variable siz gives the actual length of the mode settings. It should be stored with the mode settings and passed back to the microscope when the mode is reset.

Function Set...Stigmator

```
Function Set...Stigmator(hwindow: THandle; stigmators: TStig;  
siz:integer): integer;
```

This is a set of three functions, where ... can stand for C2, Obj or Dif. These functions will set the stigmator settings back into the microscope. Setting stigmator values is done for each stigmator type separately. It is advised to retrieve and set all stigmators settings as a single block and not change individual values, since the actual structure of the settings may vary with the CM software version and instrument type. The variable siz gives the actual length of the stigmator settings. It should be stored with the stigmator settings and passed back to the microscope when a stigmator is reset.

Function SetRotationAlignment

```
Function SetRotationAlignment(hwindow: THandle; rotal:  
TRotationAlignment) : integer;
```

This function sets the rotation alignments of the microscope, where rotal is an array of 7 x and y values. When using this function, make sure to use Get rotation alignment first to get the data from the microscope and then change the desired rotation centre (otherwise all other rotation centres get reset to zero or an undefined value). Together with Get rotation alignment, this function is meant for automated microscope alignment.

This function has been implemented from microscope software version 11 onwards.

Function SetTimeOutTime

```
Function SetTimeOutTime(hwindow: THandle; TimeOut: integer): integer;
```

This function sets the time-out for the DLL to wait for messages from the SECS application. TimeOutTime is in milliseconds. If the time out has elapsed, the DLL also aborts any further action by SECS2_32.EXE.

Function SetTVdetPos

```
Function SetTVdetPos(hwindow: THandle; pos: integer): integer;
```

This procedure switches the setting for the TV positions. Pos has the values 0 for central, 2 for off-axis and 3 for not selected. To be used if the microscope has more than one TV system, mounted in different positions.

This function has been implemented from microscope software version 11 onwards.

Function Softkey

```
Function Softkey(hwindow: THandle; sftkey,presses: integer): integer;
```

Softkey presses one of the softkeys alongside the MICROCONTROLLER screen by Presses. Counting convention of the softkeys is 0 to 15, starting at top left, going down, then to the right-hand side (so bottom left = 7; top right = 8, etc.).

Function SRSCalibrate

```
Function SRSCalibrate(hwindow: THandle): integer;
```

SRSCalibrate starts calibration of the Specimen Relocation System. Because the calibration is a lengthy procedure, any program that uses this function should wait for ca. 30 seconds and then poll the microscope every second or so about the stage position. As long as the microscope does not report 'RCMOK', the calibration hasn't finished. Build a termination in the loop in case the calibration fails, otherwise the program will run forever.

This function has been implemented from microscope software version 2.1 onwards.

Function SRSGotoPos

```
Function SRSGotoPos(hwindow: THandle; x,y: single): integer;
```

SRSGotoPos moves the Specimen Relocation System or the CompuStage to the x,y position specified. Note that the x,y values are in micrometers. For the CompuStage SRSGotoPos is equivalent to CSGotoPos with method identified as 'XYonly'.

This function has been implemented from microscope software version 2.1 onwards.

Function **SRSGotoReg**

```
Function SRSGotoReg(hwindow: THandle; register: integer): integer;
```

SRSGotoReg moves the Specimen Relocation System to the position in the register selected. Register is a number in the range 1 up to and including 99.

This function has been implemented from microscope software version 2.1 onwards.

Function **SRSReadPos**

```
Function SRSReadPos(hwindow: THandle; var x,y: single): integer;
```

SRSReadPos reads the current position of the Specimen Relocation System or the CompuStage. Note that the x,y values are in micrometers. For the CompuStage SRSReadPos is equivalent to CSAskPos with only x and y reported.

This function has been implemented from microscope software version 2.1 onwards.

Function **SRSReadReg**

```
Function SRSReadReg(hwindow: THandle; register: integer; var x,y: single): integer;
```

SRSReadReg reads the contents of the specified register of the Specimen Relocation System. Register is a number in the range 1 up to and including 99.

This function has been implemented from microscope software version 2.1 onwards.

Function **SRSSelectReg**

```
Function SRSSelectReg(hwindow: THandle; register: integer): integer;
```

SRSSelectReg selects the specified register of the Specimen Relocation System and moves the highlight to it. Register is a number in the range 1 up to and including 99.

This function has been implemented from microscope software version 2.1 onwards.

Function SRSSetReg

```
Function SRSSetReg(hwindow: THandle; register: integer; x,y : single): integer;
```

SRSSetReg sets the specified x,y position in the register of the Specimen Relocation System selected. Note that the x,y values are in micrometers. Register is a number in the range 1 up to and including 99.

This function has been implemented from microscope software version 2.1 onwards.

Function SwitchFreeHT

```
Function SwitchFreeHT(hwindow: THandle; onoff: integer): integer;
```

SwitchFreeHT switches Free HT Control (on the PARAMETERS page) on or off, depending on the value of onoff (1 = on; 0 = off);

This function has been implemented from microscope software version 11 onwards.

Function TurnKnob

```
Function TurnKnob(hwindow: THandle; Knob,Count: integer): integer;
```

Turnknob turns one of the turn knobs on the CM microscope. The numbers of turns is given by Count. Note that the latter is an integer while turnknobs cannot be turned more than ± 65534 (word or unsigned small integer) turns. Trap input causing overflow and, for larger numbers, repeat Turnknob until the required number of turns has been given.

If the number of steps is negative, the knob will be turned counter-clockwise.

The turnknob id's (constants with Tk.. predefined in RCMWIN32.PAS) are:

TkRatio	0	TkIntensity	9
TkContrast	1	TkShiftX	12
TkBrightness	2	TkShiftY	13
TkZoom	3	TkMultifunctionX	14
TkMagnification	6	TkMultifunctionY	15
TkFocus	7	TkSpotSize	16
TkFocusStep	8	TkFilament	17

Function VideoL/VideoR

```
Function VideoL(hwindow: THandle; var video: integer): integer;  
Function VideoR(hwindow: THandle; var video: integer): integer;
```

VideoL and VideoR are functions that read the intensity of the signal on the left- and right-hand STEM monitors. The video level reported is in the range 0 (white) to 255 (black). Delphi Tutor inverts this to more conventional settings so 0 is black and 255 is white.

7.3.4 Tips

1. Although it is good practice to start each remote control program by testing whether the microscope responds (the Equipment Available function), do not put such calls in procedures where form handle is not yet defined such as the `FormCreate` of the main window. Instead put such calls in another procedure like the way Delphi Tutor starts up.
2. Keep your old code by numbering versions. You can then always retrace where a certain bug started occurring and by comparing files often easily establish what the bug is. Also you may change things but later on decide that something used in an earlier version was useful. When you still have that version, you can easily find the item again.
3. Keep backups elsewhere. A lot of effort can be wasted by accidentally deleting files or failing hardware.
4. Stick as much as possible to established Windows conventions. It may be more logical to deviate in some circumstances, but users expect certain functions in certain places. Deviations are often difficult to handle. All but simple programs should have a Help file.
5. Put comments in the code wherever something unusual is done and to explain what variables are for. A lot of people (including the author) have difficulty even with their own code when seeing it back after even as little as half a year, because things that were obvious at first are no longer so. Using object-oriented programming also helps to keep things transparent.

7.4 VISUAL BASIC

Probably the most important file coming with the Visual Basic Tutor is RCMWIN32.BAS. This file forms the interface to the Remote Control Dynamic Link Library. All constants like pushbutton and turnknob identifications are listed here. So in calling e.g. the Pushbutton function for the Diffraction button you simply use the PbDiffraction constant to identify the button. In addition, the calls to the Dynamic Link Library are declared.

7.4.1 Program structure

Visual Basic Tutor is structured around a parent MDI (Multiple Document Interface) window with three MDI Child windows, named TEM left-hand panel, TEM right-hand panel and STEM panel. Each of the three child windows contains a series of buttons whose functionality, position and appearance corresponds to the push buttons, softkeys and turn knobs of the CM microscope.

The choice for the MDI window was made as it provides an easy way of handling additional windows: they stay within the confines of the parent window, the parent window has scrollers when it is made too small to show all children, etc.

The child windows are listed under the Window menu. The TEM left-hand panel is topmost, the STEM panel, if present, at the bottom.

7.4.2 Program file RCMWIN32.BAS

All remote control procedures and data types are declared in the RCMWIN32.BAS file. The purpose of this code is to declare the appropriate calls for the Remote CM Dynamic Link Library. The code in RCMWIN32.BAS is kept separate from the other code of Visual Basic Tutor so that it can be incorporated simply into other programs.

7.4.3 Visual Basic-specific comments

1. For some (unknown) reason the dialogs in Visual Basic Tutor have icons (this is not normal Windows behaviour). It is unknown how this must be changed to normal (icon-less) behaviour.
2. When a Visual Basic program is stopped during debugging, the CloseConnection procedure is not called. Not only will this lead to later problems with the SECS2_32 driver, but it has been noticed that Visual Basic seems to hang up SECS2_32 for a while and may crash when SECS2_32 is closed. Once again it is not known what to do about this. The problem only happens during debugging.

7.4.4 Remote control procedures

The following Remote Control procedures are available, listed in alphabetical order. All return the rcmresult error message.

AbortSecsAction	ChangeFreeHt
CloseConnection	
CSAskPos	CSGetRegs
CSGotoPos	CSSetRegs
CurrentReadout	DirectOperation
EmissionCurrent	EnableUserInput
EquipmentAvailable	GetAlignment
GetCmCamValues	GetCmInfo
GetCMVar	GetFegData
GetHTAcknowledge	GetMode
GetRotationAlignment	GetStigmators
InstrumentMode	PressureReadout
Pushbutton	ResetDisplay
RetrEdxProt	RetrExtXY
RetrHTcond	ScreenCurrent
SetAlignment	SetMode
SetC2Stigmator	SetDifStigmator
SetObjStigmator	SetRotationAlignment
SetTimeOutTime	SetTVdetPos
Softkey	SRSCalibrate
SRSgotoPos	SRSgotoReg
SRSReadPos	SRSReadReg
SRSselectReg	SRSsetReg
SwitchFreeHt	TurnKnob
Video Left	Video Right

Note: Relative to the DOS version of the Remote Control, many of the data types used are converted to simpler units, only the integer, longint, single and char being used. This is because Visual Basic doesn't have data types such as bytes and words. Translation takes place in the Remote Control dynamic Link Library CMREMOTE32.DLL.

In the DOS version of the remote control a few functions additional to the ones for Windows existed. These functions have been left out of the Windows remote control because they could cause severe problems. They are related to logging (getting data from the microscope as defined on the PRINTING PARAMETERS page). Logging is the only remote control function where the microscope takes the initiative for sending messages. With multi-program environments this has the risk that one program is asking information more or less continuously (e.g., stage position) while the microscope then answers with the logging data. These do not fit the answer expected and may cause program crashes or hang-up of the remote control function on the microscope.

Function AbortSecsAction

```
Function AbortSecsAction (ByVal hwindow As Integer) As Integer
```

This function aborts waiting for a response from SECS2_32.EXE.

Function ChangeFreeHT

```
Function ChangeFreeHT (ByVal hwindow As Integer, deltaht As Single) As Integer
```

This procedure changes the high tension by the deltaht value specified. Free HT must be on for this function to work.

This function has been implemented from microscope software version 11 onwards.

Sub CloseConnection ()

This sub (the only non-function among the remote control) closes the COM interface between the CMREMOTE32.DLL dynamic link library and the SECS2_32.EXE driver. It should be called as the last remote control call before the program closes.

Function CSAskPos

```
Function CSAskPos (ByVal hwindow As Integer, x As Single, y As Single, z As Single, a As Single, B As Single) As Integer
```

CSAskPos reads the current x,y,z,a,b position of the CompuStage. Note that the x,y,z values are in micrometers, the tilt angles in degrees.

This function has been implemented from microscope software version 10 onwards, but beta tilt read-out only works correctly since version 11.

Function CSGetRegs

```
Function CSGetRegs (ByVal hwindow As Integer, CSRegisters As CSREGTYPE) As Integer
```

CSGetRegs retrieves the x,y,z,a,b positions of all 25 registers of the CompuStage. CSRegisters consists of an array of 25x GonRegType which in turn consists of an integer filled and a CSPos (an array of 5 singles).

Filled is 0 when the register is empty, 1 when it is filled.

This function has been implemented from microscope software version 10 onwards.

Function CSGotoPos

```
Function CSGotoPos (ByVal hwindow As Integer, ByVal x As Single, ByVal y  
As Single, ByVal z As Single, ByVal a As Single, ByVal b As Single, ByVal  
m As Integer) As Integer
```

CSGotoPos moves the CompuStage to the x,y,z,a,b position specified. Note that the x,y,z values are in micrometers, the a and b values in degrees. m identifies the method of stage movement.

m identifiers:

Recall	=	64	{ full 5 axes safe recall }
XYonly	=	192	{ xy recall }
Xgoto	=	1	
Ygoto	=	2	
Zgoto	=	4	
Agoto	=	8	
Bgoto	=	16	

Examples:

safe goto all axes	:	use method RECALL;
direct goto axes Z and B	:	use method (Zgoto or Bgoto)

Note: although it is possible to combine RECALL and Xgoto, etc... the microscope still treats the method as RECALL only. In order to prevent resetting the axes that are not selected, first Ask the current position and leave the axes in the Goto call.

This function has been implemented from microscope software version 10 onwards.

Function CSSetRegs

```
Function CSSetRegs (ByVal hwindow As Integer, CSRegisters As CSREGTYPE)  
As Integer
```

CSSetRegs sets the x,y,z,a,b positions of all 25 registers of the CompuStage. CSRegisters consists of an array of 25x GonRegType which in turn consists of an integer filled and a CSPos (an array of 5 singles).

Filled is 0 when the register is to be cleared, 1 when it is to be filled with the values of CSPos, 2 if the contents of the register are to remain unchanged.

This function has been implemented from microscope software version 10 onwards.

Function CurrentReadout

Function CurrentReadout (ByVal hwnd As Integer, currents As CURRENTSTYPE, currentid As Integer) As Integer

This function reads the current of the lenses, deflection coils and stigmators on the microscope (equivalent to DISPLAY CURRENTS on the PARAMETERS page).

The currents are returned in Currents, an array of 26 Singles.

currentid identifies which current should be read out. If currentid is -1 all 26 currents are read out. If it is in the range 0..25 the requested current is put into the currents(1).

Function DirectOperation

Function DirectOperation (ByVal hwnd As Integer, ByVal diroperation As Integer, ByVal param As Integer) As Integer

Direct operation (constants have been predefined in RCMWIN32.BAS) contains the following functionality with the diroperation:

- | | | |
|----|-------------------|---|
| 1 | DirOpD | set D mode, Param is the number of camera lengths from the smallest one |
| 2 | DirOpFocusStep | set Focus step size, Param is the step size |
| 3 | DirOpHM | set HM/SA magnification, Param is the magnification step from the smallest (LM !) one |
| 4 | DirOpHTStep | set HT step, Param is the number of steps from the lowest HT setting |
| 5 | DirOpLAD | set LAD mode, Param is the number of steps from the smallest LAD camera length |
| 6 | DirOpLM | set LM magnification, Param is the magnification step from the smallest one |
| 7 | DiropPressReady | Press Ready button |
| 8 | DirOpSpotSize | set Spot size, Param is the spot size |
| 9 | DirOpHTmax | set High Tension to maximum, Param has no function |
| 10 | DiropBeamBlankOn | switch beam blanker on, Param has no function |
| 11 | DiropBeamBlankOff | switch beam blanker off, Param has no function |
| 12 | DirOpEDXproton | switch EDX protection on, Param has no function |
| 13 | DirOpEDXprotoff | switch EDX protection off, Param has no function |
| 14 | DiropExtXYOn | switch External XY deflection on, Param has no function |
| 15 | DiropExtXYOff | switch External XY deflection off, Param has no function |
| 16 | DiropNormImLenses | normalise imaging lenses (= pressing TEM Exposure with main screen down but this operation can be performed with screen up) |

Direct Operation 16 has been implemented from microscope software version 11 onwards.

Note: Neither setting HM/SA or LM magnifications switches the microscope to image mode and this function only operates correctly when the microscope is in image mode. The difference between these two direct operations is that the set LM magnification first goes down to the minimum LM magnification and then up to the requested value, while set HM/SA magnification goes up to the maximum and then down.

Function EmissionCurrent

```
Function EmissionCurrent (ByVal hwindow As Integer, emssncurrent As Single) As Integer
```

Emission current reads the emission current of the microscope in Ampères.

Function EnableUserInput

```
Function EnableUserInput (ByVal hwindow As Integer, ByVal fEnable As Integer) As Integer
```

EnableUserInput enables and disables the user interface of the CM microscope (in other words, when disabled the user cannot change anything on the microscope using turnknobs, pushbuttons and softkeys).

This function has been implemented from microscope software version 11 onwards.

Note: When using this function, always make sure to turn it off afterwards.

Function EquipmentAvailable

```
Function EquipmentAvailable (ByVal hwindow As Integer, ByVal szModel As String, ByVal szVersion As String) As Integer
```

Function EquipmentAvailable establishes whether communication is possible and returns the type of microscope in szModel (CM10, CM12, CM20, CM30, CM100, CM120, CM200, CM300) and the CM software version number in szVersion (for example, 93512 for version 1.2, 00021 for version 2.1, 00030 for 3.0, etc.). The '935' at the beginning at not been used from version 2.1 onwards.

In case there is doubt about the operation of a user-written program, the EquipmentAvailable test can also be done directly by SECS2_32.EXE. Open the icon by double-clicking on it and select Window, Are you there? The microscope should respond by giving its type and software version).

Function GetAlignment

```
Function GetAlignment (ByVal hwindow As Integer, alignment As  
ALIGNMENTSTYPE, leng As Integer) As Integer
```

This function will retrieve the alignments (excluding the gun alignment) from the microscope. It is advised to retrieve and set all alignments as a single block and not change individual values, since the actual structure of the alignments may vary with the CM software version and instrument type. The variable leng gives the actual length of the alignments. It should be stored with the alignment and passed back to the microscope when the alignment is reset.

Function GetCmCamValues

```
Function GetCmCamValues (ByVal hwindow As Integer, camvalues As  
CAMVALUETYPE) As Integer
```

This procedure retrieves various parameters related to the camera.

Camvalues is a record with the following fields carrying information.

Exposurenr	: array[1..4] of char	exposure number
Stock	: integer	exposure stock
Seriessize	: integer	number of exposures in series
TEMexposureLED	: integer	status of TEM Exposure button LED (odd = on, even = off)
Double	: integer	double exposure active (odd) or off (even)
Expmodeseries	: integer	exposure series active (odd) or off (even)
Seriesthrfocus	: integer	through-focus series active (odd) or fixed-focus (even)
Exptimeauto	: integer	automatic exposure selected if odd
Expt>manual	: integer	manual exposure time selected if odd
Exptimer	: integer	exposure timer selected if odd
PVPrunning	: integer	Pre-Vacuum Pump is running if odd (cannot take an exposure then)
Measuredexptime	: single	measured exposure time value
Manualsetexptime	: single	manual exposure time value
Emulsionnumb	: single	emulsion setting
Dataintfactor	: single	data intensity setting

This function has been implemented from microscope software version 12 onwards.

Function GetCMInfo

Function GetCMInfo (ByVal hwnd As Integer, cminfo As CMINFOTYPE, Leng As Integer) As Integer

This procedure retrieves various system parameters into cminfo. Some of the following fields relate only to the CompuStage. Leng gives the length of cminfo (13 for CM software version 10 and 11). This value can be used to check for future extensions of the values in cminfo.

Cminfo is a user-defined type with the following fields carrying information:

CmType	: integer	0..7: CM10/12/20/300/100/120/200/300
ObjLens	: integer	0..3: High Contrast/TWIN/SuperTWIN/UltraTWIN
DiffLens	: integer	0..1: normal diffraction lens/special diffraction lens
Gun	: integer	0..2: Tungsten/LAB6/FEG}
Magn	: single	the magnification as displayed on data monitor
HolderInserted	: integer	true = 1
GonioMeter	: integer	0..2: manual/CompuStage/SRS
CompuEnabled	: integer	true when CompuStage enabled
Joysticks	: integer	bitwise: ...bazyx; 1 if enabled
AxesAvail	: integer	bitwise: ...bazyx; 1 if available

Note: For the CM10/CM100 TWIN (objective-lens 1) is the BioTWIN, for the CM12/CM120 High Contrast (objective-lens 0) is the BioTWIN.

This function has been implemented from microscope software version 10 onwards.

Function GetCMvar

Function GetCMvar (ByVal hwnd As Integer, cmvar As CMVARTYPE, Leng As Integer) As Integer

This procedure retrieves various system variables into cmvar. Leng gives the length of cmvar (54 for CM software version 11, 88 for software version 12). This value can be used to check for future extensions of the values in cmvar.

CmVar is a user-defined type with the following fields carrying information:

Page	: long	the number of the current MICROCONTROLLER page
ButtonState	: long	the states of pushbuttons, see below
Magn	: single	the current magnification (note: camera length in nm- s !)
HT	: single	the current high tension
D1	: single	the measuring value D1
D2	: single	the measuring value D2
Angle	: single	the measuring angle
Spotsize	: single	the current spot size (in nm)
Intensity	: single	the current intensity setting (range 1000.. 100000)
BeamShiftX	: single	Low Dose Search state beam shift X
BeamShiftY	: single	Low Dose Search state beam shift Y
ImageShiftX	: single	Low Dose Search state image shift X (-35000 .. 35000)
ImageShiftY	: single	Low Dose Search state image shift Y (-35000 .. 35000)
MainScrn	: integer	main screen up (true) or down (false)
DFmode	: integer	dark field mode, bitwise bit 0: 0 X-Y, 1 Cone; bit 1: 0 manual, 1 dynamic
TVdetPos	: integer	position of the TV (0 central, 2 off-axis, 3 not selected)

From CM software version 12 onwards:

Measx,Measy	: single	two values giving the measuring data expressed as the x,y settings before conversion for the display
Beamx,Beamy	: single	two values giving the beam position for the currently active mode
Defocus	: single	defocus display value (as on MICROCONTROLLER screen) in nanometers
Focusstep	: integer	focus step value (1 to 9)
Objcurrent	: single	objective lens settings, a value between 0 and 100 000
Dftx,Dfty	: single	dark field tilt display values (as x-y or rot-tilt, depending on active dark-field mode)

Buttonstate contains the current status of the pushbutton that have the On,Off,Press functionality. Buttonstate defines bitwise if the following buttons are on (1) or off. Bit definition for 0..15 is

0	Diffraction	1	Stigmator
2	Dark field	3	Alignment
4	Scan Stop	5	Dual Signal
6	Dual Mag	7	Split screen
8	Exchange mode	9	Y mode
10	STEM Exposure	11	Set contr./br.
12	Auto contr./br.	13	Invert video
14	Intensity Fine	15	Wobbler

Note: In order to allow easy distinction between image magnification and camera length, the camera length is given in nanometers (no overlap in ranges).

This function has been implemented from microscope software version 11 onwards, extended in software version 12.

Function GetFegData

```
Function GetFegData (ByVal hwindow As Integer, fegdata As FEGDATATYPE) As Integer
```

This function is used to retrieve information about the Field Emission Gun into fegdata. Fegdata is a user-defined type that contains the following:

state : integer the FEG state: 1 off, 2 shutdown, 3 IGP3 on, 4 standby, 5 operate, 6 service, 7 cold start, 8 warm start, 9 test, 10 error

time : integer the wait time as displayed on the MICROCONTROLLER FEG pages

Fil, Extr, Gun, Emis, IGP3, ExtrLimit : 6 singles containing the data displayed on the CONFIGURATION page and the FEG DISPLAY page.

Data measurement frequency on the microscope is every 2 seconds.

This function has been implemented from microscope software version 11 onwards.

Function GetHtAcknowledge

```
Function GetHtAcknowledge (ByVal hwindow As Integer; HTcond As Integer) As Integer
```

This functions determines whether the HT on the microscope is on (odd) or off (even). It thus differs from RetrHTcond which only determines if the HT can be switched on or not.

This function has been implemented from microscope software version 12 onwards.

Function GetMode

```
Function GetMode (ByVal hwindow As Integer, mode As MODETYPE, leng As Integer) As Integer
```

This function will retrieve the mode settings from the microscope. It is advised to retrieve and set all modes as a single block and not change individual values, since the actual structure of the modes settings may vary with the CM software version and instrument type. The variable leng gives the actual length of the mode settings. It should be stored with the mode settings and passed back to the microscope when the mode is reset.

Function GetRotationAlignment

```
Function GetRotationAlignment (ByVal hwindow As Integer, rotal As ROTATIONALIGNMENTTYPE) As Integer
```

This function gets the rotation alignment of the microscope. Together with Set rotation alignment, this function is meant for automated microscope alignment.

This function has been implemented from microscope software version 11 onwards.

Function GetStigmators

```
Function GetStigmators (ByVal hwindow As Integer, stigmators As Stigtype, leng As Integer) As Integer
```

This function will retrieve the stigmator settings from the microscope. It is advised to retrieve and set all stigmators settings as a single block and not change individual values, since the actual structure of the settings may vary with the CM software version and instrument type. The variable leng gives the actual length of the stigmator settings. It should be stored with the stigmator settings and passed back to the microscope when a stigmator is reset.

Function InstrumentMode

```
Function InstrumentMode (ByVal hwindow As Integer, ByVal InstrMode As Integer) As Integer
```

This procedure sets the microscope into the requested mode. instrmode's are (constants with Md.. are predefined in RCMWIN32.BAS):

Value	Mode	Constant
1	Microprobe scan	MdMicroprobeScan
2	Nanoprobe scan	MdNanoprobeScan
3	Nanoprobe	MdNanoprobe
4	Post-specimen scan	MdPostSpecimenScan
5	Rocking Beam	MdRockingBeam
6	Scanning	MdScanning
7	TEM Low Dose	MdLowDose
8	(HR-)TEM	MdTem

Function PressureReadout

```
Function PressureReadout (ByVal hwindow As Integer, pressures As PRESSURETYPE) As Integer
```

This procedure returns the vacuum pressures of P1, P2, P3 and IGP in pressures, an array of four singles. The values are the same as those on the VACUUM page of the microscope.

Function PushButton

```
Function PushButton (ByVal hwindow As Integer, ByVal button As Integer,  
ByVal presstype As Integer) As Integer
```

Pushbutton presses one of the push buttons on the CM microscope. Some push buttons can have presstype On (0), Off (1) or Press (2), other ones just Press. On (0), Off (1) and Press (2) are defined in RCMWIN32.BAS. The push button id's (constants with Pb.. predefined in RCMWIN32.BAS) are (bold identifies as On,Off,Press; otherwise Press only):

PbDiffraction	3	PbSplitScreen	43
PbAutofocus	4	PbExchSignal	44
PbReset	6	PbExp1	45
PbReady	7	PbExp2	46
PbStigmator	27	PbExp3	47
PbDarkField	28	PbZmode	48
PbAlignment	29	PbYZmode	49
PbVacuumOn	31	PbYmode	50
PbFullFrame	32	PbStemExposure	51
PbSelectedArea	33	PbSetButton	52
PbCrosshairs	34	PbAutoButton	53
PbLine	35	PbInvertSignal	54
PbScanstop	36	PbExchControl	55
PbSlow	37	PbExposure	60
PbFast	38	PbIntReset	61
PbTV	39	PbIntFine	62
PbDualSignal	41	PbWobbler	63
PbDualMag	42		

Buttons that cannot be pushed are HT, Vacuum Off and the main panel On, Standby, Off.

Function ResetDisplay

```
Function ResetDisplay (ByVal hwindow As Integer) As Integer
```

This function forces the CM microscope to update the display on the MICROCONTROLLER. Updates of the display have low priority on the microscope so the remote control may appear to be functioning incorrectly. Use the ResetDisplay function after switching Free HT off (the microscope software does not update the HT value correctly).

This function has been implemented from microscope software version 11 onwards.

Function RetrEDXProt

```
Function RetrEDXProt (ByVal hwindow As Integer, EDXProt As Integer) As Integer
```

This function retrieves whether the EDX protection function (on the VACUUM page if present) is on (1) or off (0).

Function RetrExtXY

```
Function RetrExtXY (ByVal hwindow As Integer, ExtXY As Integer) As Integer
```

This function retrieves whether the External XY deflection function (if an External Analog Interface Board is mounted, displayed as EDX detector on the DETECTORS page) is on (1) or off (0).

Function RetrHtCond

```
Function RetrHtCond (ByVal hwindow As Integer, HTcond As Integer) As Integer
```

This function determines whether the High Tension may be switched on (1) or not (0). The actual switching of the High Tension is not possible through the remote control for safety reasons.

Function Screencurrent

```
Function Screencurrent (ByVal hwindow As Integer, scrncurrent As Single) As Integer
```

This function measures the electron current on the viewing screen in Ampères. Note that the current read-out is amplified by about 15x when the small screen is inserted. Whether this screen is in or out is not traceable by the CM software.

In order to get the real electron current falling on the screen, the value returned by the remote control should be multiplied by 2.5x to compensate for loss of backscattered electrons (measured with a Faraday cage for 200 kV).

Function SetAlignment

```
Function SetAlignment (ByVal hwindow As Integer, alignment As ALIGNMENTSTYPE, ByVal leng As Integer) As Integer
```


This function will reset the alignments (excluding the gun alignment) back into the microscope. It is advised to retrieve and set all alignments as a single block and not change individual values, since the actual structure of the alignments will vary with the CM software version and instrument type. The variable leng is the actual length of the alignments. It should be stored with the alignment and passed back to the microscope when the alignment is reset.

In order to ensure that the alignments are reset properly, the SetAlignments function should be preceded by pressing the Alignment button off and followed by pressing the Wobbler on, Diffraction on, Wobbler off, Diffraction off.

Function SetMode

```
Function SetMode (ByVal hwnd As Integer, mode As MODETYPE, ByVal leng As Integer) As Integer
```

This function will set the mode settings back into the microscope. It is advised to retrieve and set all modes as a single block and not change individual values, since the actual structure of the modes settings may vary with the CM software version and instrument type. The variable leng gives the actual length of the mode settings. It should be stored with the mode settings and passed back to the microscope when the mode is reset.

Function SetRotationAlignment

```
Function SetRotationAlignment (ByVal hwnd As Integer, rotal As ROTATIONALIGNMENTTYPE) As Integer
```

This function sets the rotation alignments of the microscope, where rotal is an array of 7 x and y values. When using this function, make sure to use Get rotation alignment first to get the data from the microscope and then change the desired rotation centre (otherwise all other rotation centres get reset to zero or an undefined value). Together with Get rotation alignment, this function is meant for automated microscope alignment.

This function has been implemented from microscope software version 11 onwards.

Function Set...Stigmator

```
Function Set...Stigmator (ByVal hwnd As Integer, stigmators As Stigtype, ByVal leng As Integer) As Integer
```

This is a set of three functions, where ... can stand for C2, Obj or Dif. These functions will set the stigmator settings back into the microscope. Setting stigmator values is done for each stigmator type separately. It is advised to retrieve and set all stigmators settings as a single block and not change individual values, since the actual structure of the settings may vary with the CM software version and instrument type. The variable leng gives the actual length of the stigmator settings. It should be stored with the stigmator settings and passed back to the microscope when a stigmator is reset.

Function SetTimeOutTime

```
Function SetTimeOutTime (ByVal hwindow As Integer, ByVal TimeOut As Integer) As Integer
```

This function sets the time-out for the DLL to wait for messages from the SECS application. TimeOutTime is in milliseconds. If the time out has elapsed, the DLL also aborts any further action by SECS2_32.EXE.

Function SetTVdetPos

```
Function SetTVdetPos (ByVal hwindow As Integer, ByVal Pos As Integer) As Integer
```

This procedure switches the setting for the TV positions. Pos has the values 0 for central, 2 for off-axis and 3 for not selected. To be used if the microscope has more than one TV system mounted in different positions.

This function has been implemented from microscope software version 11 onwards.

Function Softkey

```
Function Softkey (ByVal hwindow As Integer, ByVal sftky As Integer, ByVal presses As Integer) As Integer
```

Softkey presses one of the softkeys alongside the MICROCONTROLLER screen by presses. Counting convention of the softkeys is 0 to 15, starting at top left, going down, then to the right-hand side (so bottom left = 7; top right = 8, etc.).

Function SRSCalibrate

```
Function SRSCalibrate (ByVal hwindow As Integer) As Integer
```

SRSCalibrate starts calibration of the Specimen Relocation System. Because the calibration is a lengthy procedure, any program that uses this function should wait for ca. 30 seconds and then poll the microscope every second or so about the stage position. As long as the microscope does not report 'RCMOK', the calibration hasn't finished. Build a termination in the loop in case the calibration fails, otherwise the program will run forever.

This function has been implemented from microscope software version 2.1 onwards.

Function SRSGotoPos

```
Function SRSGotoPos (ByVal hwindow As Integer, x As Single, y As Single)
As Integer
```

SRSGotoPos moves the Specimen Relocation System or the CompuStage to the x,y position specified. Note that the x,y values are in micrometers. For the CompuStage SRSGotoPos is equivalent to CSGotoPos with method identified as 'Xyonly'.

This function has been implemented from microscope software version 2.1 onwards.

Function SRSGotoReg

```
Function SRSGotoReg (ByVal hwindow As Integer, ByVal register As Integer)
As Integer
```

SRSGotoReg moves the Specimen Relocation System to the position in the register selected. Register must be in the range 1 up to and including 99.

This function has been implemented from microscope software version 2.1 onwards.

Function SRSReadPos

```
Function SRSReadPos (ByVal hwindow As Integer, x As Single, y As Single)
As Integer
```

SRSReadPos reads the current position of the Specimen Relocation System or the CompuStage. Note that the x,y values are in micrometers. For the CompuStage SRSReadPos is equivalent to CSAskPos with only x and y reported.

It is implemented from microscope software version 2.1 onwards.

Function SRSReadReg

```
Function SRSReadReg (ByVal hwindow As Integer, ByVal register As Integer,
x As Single, y As Single) As Integer
```

SRSReadReg reads the contents of the specified register of the Specimen Relocation System. Register must be in the range 1 up to and including 99.

This function has been implemented from microscope software version 2.1 onwards.

Function SRSSelectReg

```
Function SRSSelectReg (ByVal hwindow As Integer, ByVal register As Integer) As Integer
```

SRSSelectReg selects the specified register of the Specimen Relocation System and moves the highlight to it. Register must be in the range 1 up to and including 99.

This function has been implemented from microscope software version 2.1 onwards.

Function SRSSetReg

```
Function SRSSetReg (ByVal hwindow As Integer, ByVal register As Integer, Byval x As Single, Byval y As Single) As Integer
```

SRSSetReg sets the specified x,y position in the register of the Specimen Relocation System selected. Note that the x,y values are in micrometers. Register must be in the range 1 up to and including 99.

This function has been implemented from microscope software version 2.1 onwards.

Function SwitchFreeHT

```
Function SwitchFreeHT (ByVal hwindow As Integer, ByVal onoff As Integer) As Integer
```

SwitchFreeHT switches Free HT Control (on the PARAMETERS page) on or off, depending on the value of onoff (1 = on; 0 = off);

This function has been implemented from microscope software version 11 onwards.

Function TurnKnob

```
Function TurnKnob (ByVal hwindow As Integer, ByVal Knob As Integer, ByVal Count As Integer) As Integer
```

Turnknob turns one of the turn knobs on the CM microscope. The numbers of turns is given by Turns. Note that the latter is an integer while some turnknobs can be turned more than " 32767 turns. Trap input causing integer overflow and, for larger numbers, repeat Turnknob until the required number of turns has been given. If the number of steps is negative, the knob will be turned counter-clockwise.

The turnknob id's (constants with Tk.. predefined in RCMWIN32.BAS) are:

TkRatio	0	TkIntensity	9
TkContrast	1	TkShiftX	12
TkBrightness	2	TkShiftY	13
TkZoom	3	TkMultifunctionX	14
TkMagnification	6	TkMultifunctionY	15
TkFocusKnob	7	TkSpotSize	16
TkFocusStep	8	TkFilament	17

Function VideoL/VideoR

```
Function VideoL (ByVal hwindow As Integer, video As Integer) As Integer  
Function VideoR (ByVal hwindow As Integer, video As Integer) As Integer
```

VideoL and VideoR are functions that read the intensity of the signal on the left- and right-hand STEM monitors. The video level reported is in the range 0 (white) to 255 (black). Visual Basic Tutor inverts this to more conventional settings so 0 is black and 255 is white.

7.4.5 Tips

1. Keep your old code by numbering versions. You can then always retrace where a certain bug started occurring and by comparing files often easily establish what the bug is. Also you may change things in later versions but later on decide that something used in an earlier version was useful. When you still have that version, you can easily find the item again.
2. Keep backups elsewhere. A lot of effort can be wasted by accidentally deleting files or failing hardware.
3. Stick as much as possible to established Windows conventions. It may be more logical to deviate in some circumstances, but users expect certain functions in certain places. Deviations are often difficult to handle. All but the simplest of programs should have a Help file.
4. Put comments in the code wherever something unusual is done and to explain what variables are for. A lot of people (including the author) have difficulty even with their own code when seeing it back after even as little as half a year, because things that were obvious at first are no longer so.

8 PROGRAMMING IN OTHER LANGUAGES

This chapter provides detailed information about the organisation of data transport to and from the Remote Control Dynamic Link Library, and then onto SECS2_32.EXE and the CM microscope. This information is needed if you want to do your own programming in a language other than Delphi or Visual Basic.

The first section of this chapter gives a short introduction on how the data are transported, and how the Dynamic Link Library should be used. The final section gives, for every possible function, a detailed description of how data are transported to and from the Dynamic Link Library.

8.1 INTRODUCTION

For data transport to and from the CMREMOTE32.DLL Dynamic Link Library a number of data structures need to be defined and the appropriate calls must be made to the DLL. For Delphi and Visual Basic the relevant declarations are contained in the RCMWIN32.PAS and RCMWIN32.BAS program files. You can either copy one of these files (the .PAS file is in ASCII format) and modify it for the programming language you intend to use or write the required code from the information provided below.

8.2 CONTROLLING THE DYNAMIC LINK LIBRARY

Unless done automatically by the programming language in use, the connection to the CMREMOTE32.DLL must be made at startup to ensure it is or becomes loaded. When the program is exited, CMREMOTE32.DLL should be unloaded.

Because of the limitations originally present in data types for Visual Basic, the only data types used by the Dynamic Link Library are integers, single-precision reals, long integers and char's. The Dynamic Link Library takes care of translating all other data types.

The following types are used:

- char : a byte value, representing a character decoded according to the ASCII table.
- integer : a signed 32-bit number ranging from -2147483648 to 2147483647 (also longint)
- single : a signed 32-bit number, ranging from 1.5×10^{-45} to 3.4×10^{38} with seven to eight significant digits

All DLL functions return the rcmresult, an error relating to the acceptance of the remote operation by the CM microscope or SECS2_32.EXE.

8.3 REMOTE CONTROL FUNCTIONS

8.3.1 Communication initiation, test and closing

EquipmentAvailable

The function EquipmentAvailable is used for initiating communication and testing the communication link. When successful, it returns the instrument model and CM software version.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT EquipmentAvailable(HWND hWnd, char FAR *szModel, char FAR *szVersion)
```

where **szModel** and **szVersion** are arrays of 6 char's.

In case there is doubt about the operation of a user-written program, the EquipmentAvailable test can also be done directly by SECS2_32.EXE. Open the icon by double-clicking on it and select **Window, Are you there?** The microscope should respond by giving its type and software version).

CloseConnection

This procedure closes the COM connection between the CMREMOTE32.DLL and the SECS2_32 driver. It should be called as the last remote control call of the program and before the dynamic link library is unloaded.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT void CloseConnection()
```

AbortSecsAction

This function aborts waiting for a response from SECS2.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT AbortSecsAction(HWND hWnd)
```


SetTimeOutTime

This function sets the time-out for the DLL to wait for messages from the SECS application. TimeOutTime is in milliseconds. If the time out has elapsed, the DLL also aborts any further action by SECS2_32.EXE.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SetTimeOutTime(HWND hWnd, int TimeOutTime)
```

8.3.2 Control of pushbuttons, softkeys and turnknobs

This category of functions enables you to control (nearly) all pushbuttons, softkeys and turnknobs on the microscope. Functions that cannot be controlled remotely (for obvious safety reasons) are HT, Vacuum Off and the hardware On, Off and Standby buttons.

Pushbutton

This function presses a pushbutton resulting in the function belonging to it being switched on, off or just pressed, depending on the value of presstype (0 = on, 1 = off, 2 = press). The following table gives an overview of all pushbuttons and related Id values. An asterisk in the column 'Press only' indicates that this button can only be pressed, so not switched to on or off. The function returns the rcmresult.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT PushButton(HWND hWnd, int button, int presstype)
```

where **button** is the identification of the pushbutton and **presstype** is 0..2

Pushbutton	button	Press only	Pushbutton	button	Press only
Alignments	29		Line scan	35	*
Auto	53		Ready	07	*
Autofocus	04	*	Reset	06	*
Cross	34	*	Reset intensity	61	*
Dark field	28		Scan stop	36	
Diffraction	03		Sel. area scan	33	*
Dual exch.	44		Set	52	
Dual magn.	42		Slow scan	37	*
Dual signal	41	*	Split screen	43	
Exch. signal	55	*	STEM exp.	51	
Exposure 1	45	*	Stigmators	27	
Exposure 2	46	*	TEM exp.	60	*
Exposure 3	47	*	TV	39	*
Fast scan	38	*	Vacuum On	31	*
Fine	62		Wobbler	63	
Full frame	32	*	Ymode	50	
Inv. video	54		YZmode	49	*
			Zmode	48	*

Softkey

This function presses a softkey on the microscope one or more times. The softkey Id's are numbered (top to bottom) from 0 to 7 on the left-hand side, and 8 to 15 on the right-hand side of the data monitor.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT Softkey(HWND hWnd, int softkey, int presses)
```

where **softkey** is the softkey number and **presses** the number of time softkey must be pressed.

Turnknob

This function turns a control knob (panel coder) the required number of steps in the desired direction.

One complete revolution of a turnknob is 512 steps. If the number of steps is negative, the knob will be turned counter-clockwise.

The following table gives an overview of all turnknobs on the microscope and related Id values:

Turnknob	Id	Turnknob	Id
Brightness	02	Multifunction X	14
Contrast	01	Multifunction Y	15
Filament	17	Ratio	00
Focus	07	Shift X	12
Focus step	08	Shift Y	13
Intensity	09	Spotsize	16
Magnification	06	Zoom	03

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT TurnKnob(HWND hWnd, int Knob, int Count)
```

where **Knob** is the turnknob identifier and **Count** the number of turns. Note that Count is an integer even though the range of some turnknobs is more than " 32767. If more turns are to be given, divide the total turns in steps that are compatible with integers and repeat the function for the number of steps required.

8.3.3 Control of specimen relocation system

This category of functions, referring to the manual goniometer with motor drives (called the Specimen Relocation System) has been implemented from version 2.1 microscope software onwards. Some of these functions, such as **Move to position** and **Read position** work equally well with the CompuStage (though for X,Y axes only).

Calibrate Specimen Relocation System

This function calibrates the specimen relocation system.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SRSCalibrate(HWND hWnd)
```

Move stage to location

This function moves the stage to a given location.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SRSGotoPos(HWND hWnd, float x, float y)
```

where **x** and **y** are singles.

Move stage to register location

This function moves the stage to a location stored in a register.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SRSGotoReg(HWND hWnd, int registers)
```

where **registers** is the register number 1..99.

Read current stage location

This function reads the coordinates of the current stage location.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SRSReadPos(HWND hWnd, float FAR * x, float FAR * y)
```

where **x** and **y** are singles.

Read register contents

This function reads the coordinates of a stage location stored in a register.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SRSReadReg(HWND hWnd, int registers, float FAR * x,  
float FAR * y)
```

where **registers** is the register number 1..99 and **x** and **y** are singles.

Select a register

Selects a specimen relocation system register as the current one.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SRSSelectReg(HWND hWnd, int registers)
```

where **registers** is the register number 1..99.

Store coordinates in a register

This function stores the coordinates of a stage location in a register.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SRSSetReg(HWND hWnd, int registers, float x, float y)
```

where **registers** is the register number 1..99 and **x** and **y** are singles.

8.3.4 Measuring currents, pressures and video signals

This category of functions performs internal measurements of microscope parameters, such as lens currents, vacuum pressures, video levels, etc.

Read lens, deflection coils and stigmator currents

This function reads the lens currents from the microscope. The structure of the currents read-out is the same as that on DISPLAY CURRENTS of the PARAMETERS page on the CM microscope.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT CurrentReadout(HWND hWnd, float FAR * currents, int CurrentID)
```

where **currents** is an array of 26 singles.

If **CurrentID** is -1 then the array represents:

1..8	C1, C2, TWN, OBJ, DIF, INT, P1, P2
9..20	GUN U-X, U-Y, L-X, L-Y; BEAM U-X, U-Y, L-X, L-Y; IMAGE U-X, U-Y, L-X, L-Y
21..26	C2 STIG X, Y; OBJ STIG X, Y; DIF STIG X, Y

Otherwise if **CurrentID** is in the range 0..25 then the current of the selected item (0 is C1, 1 is C2, etc.) entered in the first element of currents.

Read emission current

This function reads the emission current (in Ampères) from the microscope (the value as indicated on the microscope's emission meter).

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT EmissionCurrent(HWND hWnd, float FAR * emssncurrent)
```

where **emssncurrent** is a single.

Read vacuum pressures

This function reads the vacuum pressures from the microscope.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT PressureReadout(HWND hWnd, PRESSURETYPE FAR * pressures)
```

where **pressures** is an array of 4 singles representing P1, P2, P3 and IGP, respectively.

Read screen current

This function reads the screen current (in Ampères) from the active screen. The currently active screen is the main viewing screen, unless the small screen is inserted. In the latter case, electrons hitting the main viewing screen do not affect the screen current reading of the small screen.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT ScreenCurrent(HWND hWnd, float FAR * scrncurrent)
```

where **scrncurrent** is a single.

In order to measure the beam current falling on the specimen, the screen current should be multiplied by 2.5x to correct for losses due to backscattered electrons (as measured with a Faraday cage at 200 kV).

Read video signal left

Returns the video level of the signal on the left-hand STEM monitor or left-hand half of the left-hand monitor (with split screen active)

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT VideoL(HWND hWnd, int FAR * video)
```

where **video** is the digitised video signal, scaled between 0 and 255 (representing white and black, respectively).

Read video signal right

Returns the video level of the signal on the right-hand STEM monitor or right-hand half of

the left-hand monitor (with split screen active)

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT VideoR(HWND hWnd, int FAR * video)
```

where **video** is the digitised video signal, scaled between 0 and 255 (representing white and black, respectively).

8.3.5 Retrieving and restoring microscope parameters

This category of functions either reads microscope column alignments, mode description data and stigmator settings from the microscope, or restores them.

There is also the possibility of starting and stopping logging of data such as general microscope parameters, configuration settings, currents, pressures, etc.

Retrieve alignment parameters

This function retrieves the microscope column alignment parameters as set during the steps on the "ALIGNMENTS" pages.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT GetAlignment(HWND hWnd, ALIGNMENTSTYPE FAR * alignment, int FAR * Length)
```

where **alignment** is an array of integers of dimension **Length**. Reserve at least 5000 integers for the alignment to accomodate possible future increases in size.

Retrieve camera settings information

This function returns the contents of the settings of the microscope camera (plate or 35mm). It has been implemented from microscope software version 12 onwards.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT GetCMCamValues(HWND hWnd, GETCAMVALUETYPE FAR * camvalues)
```

where **camvalues** is a user-defined data type with the following structure and meaning:

char	exposure_nr1	first character of exposure number
char	exposure_nr2	second
char	exposure_nr3	third
char	exposure_nr4	fourth
integer	displaystock	exposure stock size
integer	seriessize	the size of the exposure series
integer	temexposureled	whether the TEM Exposure button LED is on (odd) or off (even)
integer	double	if double exposure is active (odd)
integer	expmodeseries	if exposure series is active (odd)
integer	seriesthrfocus	if exposure series is through focus (odd) or fixed focus (even)
integer	exptimeauto	if odd, exposure time mode is auto
integer	exptimemanual	if odd, exposure time mode is manual
integer	exptimetimer	if odd, exposure time mode is timer
integer	pvprunning	if odd, the Pre-Vacuum Pump is running (cannot make an exposure then)
integer	measuredexposuretime	the value of the measured exposure time
float	manualsetexposuretime	the value of the manual exposure time
float	emulsionnumb	the emulsion setting value
float	dataintfactor	the data intensity value

Retrieve microscope information

This function returns the contents of several microscope variables. It has been implemented from microscope software version 10 onwards.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT GetCMInfo(HWND hWnd, CMINFOTYPE FAR * cminfo, int FAR * iLength)
```

where **iLength** gives the length of **cminfo** (13 for CM software version 10 and 11). This value can be used to check for future extensions of the values in **cminfo**. **cminfo** is a user-defined data type with the following structure and meaning:

integer	CM type	0..7:	CM10/12/20/30/100/120/200/300
integer	Obj lens	0..3:	HC/TWIN/SuperTWIN/UltraTWIN
integer	Diff lens	0..1:	normal/special
integer	Gun	0..2:	Tungsten/Lab6/Feg
single	Magn		magnification as displayed on the CM data monitor
integer	Holder inserted	0..1:	not inserted/inserted
integer	Goniometertype	0..2:	manual/compustage/SRS
integer	Compustage enabled	0..1:	disabled/enabled
integer	Joysticks	bitwise:	...bazyx: 0: disabled; 1: enabled;
integer	Axesavail	bitwise:	...bazyx: 0: not available; 1: available

The compustage variables are only defined when the microscope has a compustage (Goniometertype = 1).

Note: For the CM10/CM100 TWIN (objective-lens 1) is the BioTWIN, for the CM12/CM120 High Contrast (objective-lens 0) is the BioTWIN.

Retrieve CM variables

This function returns the settings of several microscope variables. It has been implemented from microscope software version 11 onwards, with an extension from software version 12 onwards.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT GetCMvar(HWND hWnd, CMVARTYPE FAR * cmvar, int FAR * iLength)
```

where **iLength** gives the length of **cmvar** (54 for CM software version 11). This value can be used to check for future extensions of the values in **cmvar**. **cmvar** is a user-defined data type with following structure and meaning:

long	pageno	number of currently active CM microscope page
long	buttonstate	expresses whether a push button is on (bitwise; 1=on) bits 0..15: 0 D, 1 STIG, 2 DF, 3 ALGN, 4 Scanstop, 5 DualSignal, 6 DualMag, 7 SplitScrn, 8 ExchMode, 9 Ymode, 10 STEM exposure, 11 ACB Set, 12 ACB Auto, 13 Inv Video, 14 Int fine, 15 Wobbler
single	magnification	magnification as displayed on CM data monitor
single	hightension	high tension as displayed on CM data monitor
single	D1	measuring D1 as displayed on CM data monitor
single	D2	measuring D2 as displayed on CM data monitor
single	angle	measuring angle as displayed on CM data monitor
single	spotsize	spot size value (nm) as displayed on CM data monitor
single	intensity	intensity setting (1000 .. 100000)
single	beamshiftx	value of beam shift X for low dose SEARCH state (relative to EXPOSURE state)
single	beamshifty	as above, Y value
single	imageshiftx	as above, image shift X value
single	imageshifty	as above, image shift Y value
integer	screendown	position of main viewing screen, 1 = down, 0 = up
integer	Dfmode	setting of dark field mode, bitwise, bit 0: 0 = X-Y, 1 = CONE; bit 1: 0 = MANUAL, 1 = DYNAMIC
integer	TVdetpos	position of TV detector: 0 = central, 2 = off-axis, 3 = not selected

From CM software version 12 onwards:

single	measx	x value of measurement, before conversion to display values
on		
		MICROCONTROLLER screen
single	measy	as above, y value
single	beamx	x value of beam deflection setting for currently active mode
single	beamy	as above, y value
single	defocus	defocus value as displayed by the microscope (in nm)
integer	focusstep	currently active focus step size (1 to 9)
single	objcurrent	the setting of the objective lens, a value between 0 and 100 000)
single	dftx	x or rot (depending on dark-field mode X-Y or CONE) value of the dark-field tilt display
single	dfty	as above, y or tilt value

Note: In order to allow easy distinction between image magnification and camera length, the camera length is given in nanometers (no overlap in ranges).

Retrieve FEG parameters

This function returns various data as displayed on the FEG-display page. The data are measured every 2 seconds by the internal CM software. It has been implemented from microscope software version 11 onwards.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT GetFegData (HWND hWnd, FEGDATATYPE FAR * fegdata)
```

where **fegdata** is a user-defined data type with the following structure and meaning:

integer	state	state: the state of the FEG, 1 = off, 2 = shutdown, 3 = Tip IGP on, 4 = standby, 5 = operate, 6 = service, 7 = cold start, 8 = warm start, 9 = test, 10 = error
integer	time	the wait time as displayed on the MICROCONTROLLER
single	filament	filament current
single	extrvolt	extraction voltage
single	gunlens	gun lens voltage
single	emission	emission current
single	igp3	IGP3 current
single	extrlimit	extraction limit

Retrieve mode settings

This function retrieves the mode description data from the microscope. These data describe all settings belonging to the current microscope state.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT GetMode (HWND hWnd, MODETYPE FAR * mode, int FAR * Length)
```

where **mode** is an array of integers of dimension **Length**. Reserve at least 500 integers for the mode to accomodate possible future increases in size.

Retrieve stigmator settings

This function retrieves the settings of the stigmators. These values cannot be interpreted.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT GetStigmators (HWND hWnd, STIGTYPE FAR * stigmators, int FAR * Length)
```

where **stigmators** is an array of integers of dimension **Length**. Reserve at least 1500 integers for the stigmators to accomodate possible future increases in size.

Restore alignment parameters

This function restores previously retrieved microscope column alignment parameters in the microscope.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SetAlignment (HWND hWnd, ALIGNMENTSTYPE FAR * alignment, int Length)
```

where **alignment** is an array of integers of dimension **Length**. Reserve at least 5000 integers for the alignment to accomodate possible future increases in size.

In order to ensure that the alignments are reset properly, restoring the alignment should be preceded by pressing the Alignment button off and followed by pressing the Wobbler on, Diffraction on, Wobbler off and Diffraction off.

Restore mode settings

This function restores previously retrieved mode description data in the microscope.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SetMode (HWND hWnd, MODETYPE FAR * mode, int Length)
```

where **mode** is an array of integers of dimension **Length**. Reserve at least 500 integers for the mode to accomodate possible future increases in size.

Restore objective stigmator settings

From the previous set of retrieved stigmator settings, this function restores the objective stigmator settings in the microscope.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SetObjStigmator(HWND hWnd, STIGTYPE FAR * stigmators, int Length)
```

where **stigmators** is an array of integers of dimension **Length**. Reserve at least 1500 integers for the stigmators to accomodate possible future increases in size.

Restore condenser stigmator settings

From the previous set of retrieved stigmator settings, this function restores the condenser stigmator settings in the microscope.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SetC2Stigmator(HWND hWnd, STIGTYPE FAR * stigmators, int Length)
```

where **stigmators** is an array of integers of dimension **Length**. Reserve at least 1500 integers for the stigmators to accomodate possible future increases in size.

Restore diffraction stigmator settings

From the previous set of retrieved stigmator settings, this function restores the diffraction stigmator settings in the microscope.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SetDifStigmator(HWND hWnd, STIGTYPE FAR * stigmators, int Length)
```

where **stigmators** is an array of integers of dimension **Length**. Reserve at least 1500 integers for the stigmators to accomodate possible future increases in size.

Get rotation alignment

This function retrieves the value of the rotation alignments for the microscope. Together with Set rotation alignment, this function is meant for automated microscope alignment. It has been implemented from microscope software version 11 onwards.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT GetRotationAlignment(HWND hWnd, ROTATIONALIGNMENTTYPE FAR * RotAl)
```

where **RotAl** consists of an array of seven times two singles (x and y direction).

Set rotation alignment

This function sets the rotation alignments of the microscope. When using this function, make sure to use Get rotation alignment first to get the data from the microscope and then change the desired rotation centre (otherwise all other rotation centres get reset to zero or an undefined value). Together with Get rotation alignment, this function is meant for automated microscope alignment. It has been implemented from microscope software version 11 onwards.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SetRotationAlignment(HWND hWnd, ROTATIONALIGNMENTTYPE FAR * RotAl)
```

where **RotAl** consists of an array of seven times two singles (x and y direction).

8.3.6 Retrieving status information

This category of function retrieves status information from the microscope.

Check EDX protection

This function checks whether EDX protection is switched on. If available, it can be on or off on the VACUUM page.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT RetrEDXProt(HWND hWnd, int FAR * EDXprot)
```

where **EDXprot** shows whether EDX protection is on (1) or off (0).

Check external XY deflection

This function checks whether External XY deflection is switched on. If available, it can be switched on on the PARAMETERS 2 page.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT RetrExtXY(HWND hWnd, int FAR * ExtXY)
```

where **ExtXY** shows whether External XY deflection is on (1) or off (0).

Check high tension condition

This function checks whether the high tension can be switched on or not.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT RetrHTCond(HWND hWnd, int FAR * HTcond)
```

where **HTcond** shows whether the high tension can be switched on (1) or not (0).

Get high tension condition

This function checks whether the high tension is on or not.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT GetHTAcknowledge(HWND hWnd, int FAR * HTack)
```

where **HTack** shows whether the high tension is on (odd) or not (even).

This function has been implemented from CM software 12 onwards.

8.3.7 Direct operations

These functions perform a direct operation on the microscope.

Direct operations

This function sets the microscope to a certain mode or changes the value of a certain parameter.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT DirectOperation(HWND hWnd, int diroperation, int param)
```

where **diroperation** is the identification of the direct operation and **param** a parameter (if needed, otherwise it is zero).

The following table gives a summary of possible direct operations, related Function Id's and parameter values:

Operation	Function Id	Parameter
Set D camera length	01	# of steps ¹
Set Focus step size	02	step size
Set HM/SA magnification	03	#of steps ²
Set high tension step	04	step ³
Set LAD camera length	05	# of steps ⁴
Set LM magnification	06	# of steps ⁵
Press Ready	07	number of presses
Set spot size	08	spot size
Switch HT to maximum	09	-
Switch beam blanker on	10	-
Switch beam blanker off	11	-
Switch EDX protection on	12	-
Switch EDX protection off	13	-
Switch External XY defl on	14	-
Switch External XY defl off	15	-
Normalise Imaging Lenses	16	-

- 1 : The number of magnification steps needed to go from the lowest camera length to the one required. Range:0..15.
- 2 : The number of the magnification step from the lowest magnification (starting at 1). Goes first to highest magnification. Range: 1..39
- 3 : The number of H.T. steps on the "PARAMETERS" page needed to go from the lowest high tension to the one required. Range: 0..5.
- 4 : The number of magnification steps needed to go from the lowest camera length to the one required. Range: 0..20.
- 5 : The number of the magnification steps from the lowest magnification (starting at 1). Goes first to lowest magnification. Range: 1..39.

Direct operation 16 has been implemented from microscope software version 11 onwards.

Note: The difference between Set HM/SA magnification and Set LM magnification is purely in the way the microscope reacts, going first to the highest magnification for the former and then coming back down, while Set LM magnification first goes to the minimum and then comes back up. The same parameter will give the same magnification in both cases.

EnableUserInput

EnableUserInput enables and disables the user interface of the CM microscope (in other words, when disabled the user cannot change anything on the microscope using turnknobs, pushbuttons and softkeys). It has been implemented from microscope software version 11 onwards.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT EnableUserInput(HWND hWnd, int bEnable)
```

where **fEnable** is 0 for false (no user input) and 1 for true (user input possible: the standard way).

Note: When using this function, always make sure to turn it off afterwards.

ResetDisplay

This function forces the CM microscope to update the display on the MICROCONTROLLER. Updates of the display have low priority on the microscope so the remote control may appear to be functioning incorrectly. Use the ResetDisplay function after switching Free HT off (the microscope software does not update the HT value correctly). It has been implemented from microscope software version 11 onwards.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT ResetDisplay(HWND hWnd)
```

Set instrument mode

This function switches the microscope to one of the possible modes.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT InstrumentMode(HWND hWnd, int instrmode)
```

where the possible modes are (in their order of appearance on the CM data monitor):

Mode	Identification
Scanning	6
(HR-)TEM	8
Low Dose	7
Nanoprobe	3
Rocking Beam	5
Microprobe Scan	1
Nanoprobe Scan	2
Post Specimen Scan	4

Set TV detector position

This function defines the position for the TV detector on the microscope. It has been implemented from microscope software version 11 onwards.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SetTVdetPos (HWND hWnd, int pos)
```

where **pos** gives the position as 0 central, 2 off-axis and 3 not selected.

Switch Free high tension control

This function switches Free High Tension control on or off (depending on availability on the microscope; standard on all microscopes from CM software version 11 onwards). It has been implemented from microscope software version 11 onwards.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT SwitchFreeHT (HWND hWnd, int onoff)
```

where **onoff** is 1 for on, 0 for off.

Change Free high tension

This function changes the high tension of the microscope by the specified amount (in kV) where positive is up and negative is down. Free HT control must be on for this function to work. It has been implemented from microscope software version 11 onwards.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT ChangeFreeHT (HWND hWnd, float deltaht)
```

where **deltaht** is a single.

8.3.8 Control of the CompuStage

This category of functions is implemented from version 10 microscope software onwards (but in version 10 the beta tilt axis is read out incorrectly).

CompuStage ask position

This function returns the current position of the CompuStage.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT CSAskPos(HWND hWnd, float FAR * x, float FAR * y, float FAR * z, float FAR * a, float FAR * b)
```

where **x**, **y**, **z**, **a**, **b** are 5 singles containing a 5 axis CompuStage position in the order X, Y, Z, A, B. The values of the positions X, Y and Z are given in μm ; A and B are in degrees.

CompuStage go to position

This function will perform a Recall of a Goto of the CompuStage.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT CSGotoPos(HWND hWnd, float x, float y, float z, float a, float b, int m)
```

where **x**, **y**, **z**, **a**, **b** are 5 singles containing a 5 axis CompuStage position in the order X, Y, Z, A, B. The values of the positions X, Y and Z are given in μm ; A and B are given in degrees. The integer **m** defines the method of moving (XYonly, recall, xgoto, ygoto, zgoto, agoto, bgoto). Both recalls (recall and XYonly) are done using the algorithms which are used on the CM to perform a recall (B to 0, A to 0, etc.). There is no check possible to determine whether the position is a correct one before performing the goto. The function will return the result as soon as the recall has been completed. The xgoto, ... methods (which can be used in any combination) define on which axes to move. When recall is 0, the normal procedure on the microscope will be used, when it is off the stage will drive directly on the axes defined.

Note: Although it is possible to make the method into combinations of recall and various axes, the microscope will still treat it as a safe recall of all five axes. Therefore always first Ask the current position and leave the axes that are not to be changed in the Goto call.

Warning: Recall should be switched off only while settings tilts within a safe range. It should not be used with X or Y movements because of the high speed of the stage under those cases.

Values:	hex	decimal	binary
XYonly	\$C0	192	11000000
recall	\$40	64	01000000
bgoto	\$10	16	00001000
agoto	\$8	8	00001000
zgoto	\$4	4	00000100
ygoto	\$2	2	00000010
xgoto	\$1	1	00000001

The method is set bitwise.

Examples of the coding of method:	bit no	method =
Perform a full recall to the given position:	01 0 00000	recall
Perform a xy only recall to the given position:	11 0 00000	xyonly
Perform a goto using only axes z and b:	00 0 10100	zgoto or bgoto

CompuStage read registers

This function will return the contents of the registers on the CM.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT CSGetRegs(HWND hWnd, CSREGISTERSTYPE FAR * csregisters)
```

Register settings for the CompuStage are controlled in one block, using the user-defined data type register which is a structure containing the fields:

```
integer filled (0: empty; 1: filled; 2: do not change state)
single x
single y
single z
single a
single b
```

and **csregisters** which is a user-defined data type consisting of an array of 25 registers (of the user-defined data type defined above). The first element contains the value of register 1; the last one of register 25. The value of the filled field in the register is 0 if the register was cleared and 1 if the register was filled.

CompuStage write registers

This function will fill the registers on the CM with the given values.

The function declaration in the CMREMOTE32.DLL header file is:

```
EXPORT CSSetRegs(HWND hWnd, CSREGISTERSTYPE FAR * csregisters)
```

Register settings for the CompuStage are controlled in one block, using the user-defined data type register which is a structure containing the fields:

integer	filled (0: empty; 1: filled; 2: do not change state)
5 singles	x,y,z,a,b

and **csregisters** which is a user-defined data type consisting of an array of 25 registers (of the user-defined data type defined above). The first element contains the value of register 1; the last one of register 25.

The value of the filled field in the registers must be 0 if the register must be cleared and 1 if the register must be filled with the values. It must have the value 2 if the contents of the register concerned must not be changed.

9 EXPLANATION OF ERROR CODES

After a call is made to the dynamic link library, an error status is returned. This variable can be used in your application program to detect erroneous situations. Messages 1 to 9 are error messages from SECS2_32.EXE, 10 and 11 from the DLL itself and messages with numbers above 50 come from the microscope. Some of these messages should never occur as refer to handling of the messages by the DLL to SECS2_32.EXE.

The possible return values from the dynamic link library are:

0	: Function performed correctly.	
1	: Message Not Sent	an error message from SECS2_32.EXE which indicates that SECS2 was unable to send a message (usually when no connection can be made to the microscope). Message was not sent.
2	: Time Out	an error message from SECS2_32.EXE indicating that the time the microscope took to respond to a message was longer than the maximum response time set (can occur when no connection with the microscope is established).
3	: SECS No Memory	indicates that SECS2_32.EXE does not have enough memory to be loaded. Should not occur anymore, retained for backward compatibility.
4	: Queue Full	the message queue from SECS2_32.EXE to the microscope is full. Should not occur anymore (SECS2_32 doesn't have a message queue; instead the CMREMOTE32.DLL will wait until it gets access or times out), retained for backward compatibility.
5	: SECS Down	SECS2_32.EXE has shut down.
10	: Wrong Response	The answer received by the Dynamic Link Library was not a response to the last message it sent. It means that two different messages have become mixed up.
11	: Illegal Parameter	A parameter in the remote control message is out of range or of the wrong type.
12	: SecsTimeOut	CMREMOTE32.DLL timed out on connecting to SECS2_32 and trying to get its message handled.

51	: Unrecognized Device Id	A message belonging with the SECS protocol. Should not occur when the CMREMOTE32.DLL Dynamic Link Library and SECS2_32.EXE are used.
53	: Invalid Stream	A message belonging with the SECS protocol. Should not occur when the CMREMOTE32.DLL Dynamic Link Library and SECS2_32.EXE are used.
55	: Invalid Function	The microscope does not know the function requested. This can happen with older microscope software that does not contain all the latest functions.
57	: Invalid Data	The data in the remote control message are not valid (e.g. out of range).
59	: Transaction Time Out	Microscope has responded initially but the data took too long in transmission.
61	: Data Too Long	Too many data were sent in the message. Should not occur when the CMREMOTE32.DLL Dynamic Link Library and SECS2_32.EXE are used.
115	: Invalid Instrument Status	The requested instrument status is impossible on the microscope. Usually means that the particular mode or status (software or hardware option) is not present.

10 EXPLANATION OF SECS2 MESSAGES

10.1 INTRODUCTION

The SECS2_32 program allows logging of the messages passing from the CMREMOTE32 Dynamic Link Library to the CM microscope and back. During program development and debugging, it may be useful to use this facility to check whether the program actually passes the correct values to or from the DLL. In order to start logging, double-click on SECS2_32.EXE (if it is minimized) and select **Record, Messages**. The messages are logged and can be inspected and/or saved in a text file.

In order to understand the messages, a description is given here of the format and meaning of the messages. The protocol of the messages always follows these steps:

1. SECS2_32.EXE tries to establish contact with the microscope.
2. If it is available, the microscope tells SECS2_32.EXE that is OK to send the message.
3. SECS2_32.EXE sends the message (or displays **Message NOT sent** if unsuccessful).
4. The microscope acknowledges receiving the message.
5. The microscope tries to establish contact with the PC.
6. The PC tells the microscope that it is OK to send the message.
7. The microscope sends a message back.
8. SECS2_32.EXE acknowledges receiving the message.

All messages consist of series of bytes (hex format; if you do not know how to convert these to decimal numbers, use the Windows Calculator with **View** switched to scientific), with a header (an initial series of ten bytes) and the actual message which can vary in length from zero to several thousands of bytes. The header contains the following:

- 2 bytes for a device identification, 434D (hex for the characters CM) for PC> CM, C34D for CM > PC.
- 2 bytes with the function identification number (see below)
- 2 bytes (block number)
- four system bytes that make up a function serial number that allows identification of the original and returned messages (the DLL will return the error wrong response if the serial numbers do not match).

Each message has a function identification. Below is a list of the functions numbers, in hex format, as sent from the PC. Messages returning from the CM have the first function byte changed to the original one with hex 80 subtracted, while the second byte is increased by 1 if no error occurred. Thus 8101 (Equipment Available) will return 0102 while C349 (CM info) returns 434A. If an error occurred, then the function number carries instead the type of remote control error (09 and then the errors as listed in chapter 8 but with 50 subtracted), with the original header following for identification.

8101	Equipment available
C041	Pushbutton on
C043	Pushbutton off
C045	Pushbutton press
C047	Disable user input
C049	Enable user input
C141	Softkey
C241	Turnknob
C343	Get HT condition
C345	Get EDX protection
C347	Get External XY deflection
C349	Get CM Info
C34B	Get CM Variables
C34D	Get FEG data
C34F	Get HT Acknowledge
C445	Current read out
C447	Pressure read out
C449	Screen current
C44B	Video Left
C44D	Video Right
C44F	Emission current
C541	Get Mode
C543	Set Mode
C545	Get Stigmator settings
C547	Set objective stigmator
C549	Set condenser stigmator
C54B	Set diffraction stigmator
C54D	Get alignment
C54F	Set alignment
C559	Get rotation alignment
C55B	Set rotation alignment
C55D	Get CM Camera Values

Direct operations:

C741 Set D
C743 Set Focus step
C745 Set HM
C747 Set HT
C749 Set LAD
C74B Set LM
C74D Set Press Ready
C74F Set Spotsize
C751 Set TV detector position
C753 Switch Free HT on/off
C755 Change Free HT
C781 Set HT to maximum
C783 Set Microprobe Scan
C785 Set Nanoprobe Scan
C787 Set Nanoprobe
C789 Set Post-specimen Scan
C78B Set Rocking Beam
C78D Set Scanning
C78F Set Low Dose
C791 Set Instrument Mode TEM
C793 Beam blanker on
C795 Beam blanker off
C797 EDX protection on
C799 EDX protection off
C79B External XY deflection on
C79D External XY deflection off
C79F Normalize Imaging Lenses

SRS functions:

C841 Goto SRS register
C843 Goto SRS position
C847 Set SRS register
C849 Read SRS register
C84F Read SRS position
C865 Select SRS register
C88B Calibrate SRS

CA5D Reset display

CompuStage functions:

CB41 CompuStage Ask position
CB43 CompuStage Goto position
CB45 Get CompuStage registers
CB47 Set CompuStage registers

10.2 MESSAGE STRUCTURES

The important message structures are listed below (many functions have no or irrelevant data structures since the function identification is the only important parameter). For clarity a slash separates the individual bytes in the list below (but does not occur in the SECS message log).

Equipment Available

PC > CM, header only

CM > PC, message: 01 02 41 length (06) model (6 bytes) 41
length (06) sw version (6 bytes)

Pushbutton on,off,press

PC > CM, message: 66 00 length (01) pushbutton id byte

CM > PC, header only

Softkey

PC > CM, message: 66 00 length (02) softkey id no. of presses

CM > PC, header only

Turnknob

PC > CM, PC message: 02 00 03 65 01 turnknob id (byte) 41
01 direction (hex character, R for right 52, or L for Left 4C) 69 02
turn counts (word = 2 bytes)

CM > PC, header only

Get HT condition, EDX protection, External XY deflection, Get HT Acknowledge

PC > CM, header only

CM > PC, CM message: 65 length (01) 00 (false) or FF (true)

Get CM Cam Values

PC > CM, header only

CM > PC, message: 21 length Exposure number (4x char = byte)
Stock (byte) Series size (byte) TEM Exposure LED (byte) Double
exposure (byte) Series exposure (byte)
Through-focus series (byte) Auto exposure time (byte)
Manual exposure time (byte) Timer exposure time (byte)
PVP running (byte) measured exposure time (real)
Manual exposure time (real) Emulsion setting (real)
Data intensity (real)

Get CM info

PC > CM, header only

CM > PC, message: 21 length CMtype (byte) Obj lens (byte) Diff
lens (byte) Gun (byte) Magnification (real = 4 bytes) Holder
inserted (byte) Goniometer (byte) Enabled (byte) Joysticks (byte)
Axes available (byte)

Get CM variables

PC > CM, header only

CM > PC, message: 21 length CM page (word = 2 bytes) Buttonstate
(word) Magnification (real = 4 bytes) HT (real) D1 (real) D2
(real) Angle (real) Spotsize (real) Intensity (real) Beam shift
x, y (2x real) Image shift x,y (2x real) Mainscreen (word) Dark
field (word) TV detector (word) Measuring (2x real) Beam shift (2x
real) Defocus (real) Focus step (integer) Objective-lens setting
(real)
Dark-field tilts (2x real)

Get FEG data

PC > CM, header only

CM > PC, message: 21 length FEG state (byte) Wait time (byte)
Filament (real = 4 bytes) Extr volt (real) Gun lens (real) Emission
current (real) IGP3 (real) Extr limit (real)

Current read out

PC > CM, message: 66 number of currents to read out (byte) current
ids (26x byte)

CM > PC, message: 91 length (68) currents (26x real = 4 bytes)

Pressure read out

PC > CM, message: 66 00 pressure ids (4x byte)

CM > PC, message: 91 length (10) pressures (4x real = 4 bytes)

Screen current

PC > CM, header only

CM > PC, message: 91 length (04) current (real = 4 bytes)

Video left/right

PC > CM, header only

CM > PC, message: 91 length (04) video level (byte) 3 bytes (no
function)

Emission current

PC > CM, header only

CM > PC, message: 91 length (04) current (real = 4 bytes)

Get mode

PC > CM, header only

CM > PC, message: 22 length (word = 2 bytes) data

Set mode

PC > CM, message: 22 length (word = 2 bytes) data

CM > PC, header only

Get stigmator settings

PC > CM, header only

CM > PC, message: 92 length (word = 2 bytes) data

Set stigmator settings

PC > CM, message: 92 length (word = 2 bytes) data

CM > PC, header only

Get alignment

PC > CM, header only

CM > PC, message: 22 length (word = 2 bytes) data

Set alignment

PC > CM, message: 22 length (word = 2 bytes) data

CM > PC, header only

Get rotation alignment

PC > CM, header only

CM > PC, message: 91 length (38) 7x2 reals (= 4 bytes)

Set rotation alignment

PC > CM, message: 92 length (00 38) 7x2 reals (= 4 bytes)

CM > PC, header only

Direct operations with a parameter

PC > CM, message: 66 00 length of parameter (byte) parameter
(byte or real = 4 bytes)

CM > PC, header only

SRS functions

All SRS functions send or receive messages of the same format, even though the message sent from the PC or CM may be empty.

Message: 21 length (0E) x,y positions (long = 4 bytes),
word (2 bytes, no meaning) byte (no meaning) register (byte) word
(no meaning) (to get positions in micrometers divide by 1000, bytes are
in sequence, not reversed as for CompuStage)

Where appropriate x,y and register data are filled, either sent from the PC or sent by the CM

CompuStage Ask position

PC > CM, header only

CM > PC, message: 21 length (byte) position (5x long = 4 bytes) (to
understand the positions, reverse the sequence of bytes, then divide by 1000 to get
micrometers/degrees)

CompuStage Goto position

PC > CM, message: 22 length (2 bytes) method (byte) position (5x
long = 4 bytes)

CM > PC, header only

CompuStage Get registers

PC > CM, header only

CM > PC, message: 22 length (2 bytes) 25x register = filled (byte) +
position (5x long = 4 bytes) (to understand the positions, reverse the sequence of
bytes, then divide by 1000 to get micrometers/degrees)

CompuStage Set registers

PC > CM, message: 22 length (2 bytes) 25x register = filled (byte) +
position (5x long = 4 bytes)

CM > PC, header only