CITRON
Creative Industrial Electronic Systems

# CITOUCHW

## Touch Driver for Windows

## V4.0x

## User's Manual

## Document revision

| Rev. | Description | Reviser | Date |
|---|---|---|---|
| 001 | First issue after translation (preliminary) | pk | 26. Nov 1996 |
| 002 | Adapted to driver revision 4.0.003 | tt | 21.Jul 1997 |
| 003 | Address and telephone number changed | pk | 11. Dec 1997 |
| 004 | New Logo | pk | 19. Feb 1998 |

## Exclusion of liability

The contents of this manual serve for information purposes only. Citron GmbH reserves the right to change the contents of this manual without prior notice. While reasonable efforts have been made in the preparation of this manual to assure its accuracy, errors may occur. Therefore, Citron GmbH assumes no liability resulting from errors or omissions in this manual or from the use of the information contained herein.
Citron GmbH appreciates suggestions with regard to improvements or corrections.



Document information

File name: h:\manuals\citouchw\ctw_e_r5.doc
Date: 19.02.98 10:47
Document revision: 2
Document reference: \\ntserv1\dokument\dformat.dot\cidoku.dot

# 1 Table of contents

# 2 Introduction

The CiTouchW mouse driver (Citron Touch driver for Windows) enables the Citron Infrared Touch (in the following referred to as IRT) to be used as a mouse substitute in conjunction with the graphical surface Microsoft® Windows™. All features of the IRT, such as the Z-axis (only if the IRT is equipped with pressure sensors) or dual touching are supported.
The requirements to be met for the operation of the CiTouchW are a Windows version of 3.1 or higher and at least an i386 microprocessor.
In addition to the IRT, either a PS/2®, bus or serial mouse can be used simultaneously. The mouse driver implemented in the CiTouchW provides the same scope of functions as the standard WINDOWS 3.1 mouse driver.
Contrary to a regular mouse the CiTouchW is capable to work with absolute coordinates. That means, the movement of a finger on the screen surface is not converted into a relative change of the cursor position. The cursor always appears at the exact position the screen is just touched. In order to improve the positioning accuracy, however, it is also possible to work with relative coordinates. The change between the coordinates modes is carried out either statically, i.e. during the configuration or dynamically, i.e. during the regular operation by means of a special finger movement, for example a dual touch.
The biggest challenge at the emulation of a mouse by a Touch is the simulation of the mouse buttons. Not all features of a mouse button can be simulated at the same time and with the same accuracy. However, due to the fact that the requirements vary between the respective application programs (for example precise timing of the mouse click, high safety regarding faulty usage, drag and drop capability ...), in most cases a certain share of the mouse functions is already sufficient for the use of the respective program. At the CiTouchW the emulation of the mouse buttons is widely user-configurable and programmable.
All parameters that influence the operating methods of the CiTouchW are centrally set by one application of the Windows' System Control. The settings can be changed either during the installation or during regular operation.

# 3 Installation

Before starting the installation, the IRT needs to be connected to a serial port of the computer. Please note that in order to operate the CiTouchW driver the serial port has to be capable of interpreting Interrupts.
If a Citron LDVGA graphics board is used in conjunction with the LDRI board, the connecting cable to the display unit already took care of an adequate connection of the IRT. In this case only the base address and the Interrupt channel for the SIO1 of the LDVG graphics board are still to be set. This is carried out by means of the utility LDVINST. For a description of this utility refer to the manual of the LDVGA.
The CiTouchW software is entirely installed under Windows. In order to start the setup program, insert the installation disk into drive A: or B:. Then, at the program manager's menu item *File®Execute* enter "A:\SETUP" (or "B:\SETUP"). After a short waiting time, in which some help files are copied and the destination system is analysed, the actual installation process starts. This process can be divided into three topics:

1. Copying the files of the CiTouchW software and updating the file SYSTEM.INI
2. Preparing the IRT hardware
3. Calibrating the IRT and changing the default parameters of the CiTouchW driver.

The single installation routines proceed automatically in the correct order. Required user inputs are prompted and carried out in dialogue windows. During the installation, Windows is re-started several times. Therefore, it is recommended to close all applications, especially open DOS boxes, prior to the installation of the CiTouchW software.

## *3.1 Copying the CiTouchW files*

The CiTouchW software consists of the following files:

- CITOUCHW.DRV The actual Windows mouse driver
- CTAPPLET.CPL System Control application for the setting of CiTouchW parameters
- CVMD.386 32 bit part of the CiTouchW driver
- CTL3DV2.DLL DLL for the 3D effects of the CTAPPLET dialogue windows
- README.TXT Text file containing the latest changes regarding the files CITOUCHW.DRV and CTAPPLET.CPL.

All files except the file README.TXT are copied into the Windows system directory (usually C:\WINDOWS\SYSTEM). In a Windows network environment, however, the file CTL3DV2.DLL is copied into the Windows directory. In case parts of the CiTouchW software already exist on the destination PC, the user is requested whether the currently installed version is to be kept or the new version is to be copied from the installation disk. Make sure newer file versions, especially of CTL3DV2.DLL, are not accidentally overwritten!
If all files are successfully copied onto the hard disk, the file SYSTEM.INI is updated. The following entries are changed:

```
[boot]
mouse.drv=citouchw.drv

[boot.description]
mouse.drv=CiTouchW V4.0.001

[386Enh]
mouse=cvmd.386
```

The old entries are not deleted but changed into a comment line by adding a semicolon. This way all changes can easily be undone.
If there was a previous version of the CiTouchW driver installed on the destination PC, some entries of the existing file CITOUCHW.INI are transferred into the new CTW300.INI. To find out which entries are concerned, refer to the file README.TXT on the installation disk.
After SYSTEM.INI was updated, Windows needs to be re-started for the changes to become effective. After a safety query, the re-start occurs automatically.

## 3.2 Setup of the IRT hardware

If Windows is restarted, the setup program automatically starts the System Control with the CiTouchW Hardware Setup. In this dialogue window both the base address and the ISA bus Interrupt number of the serial port the IRT is connected to are set.

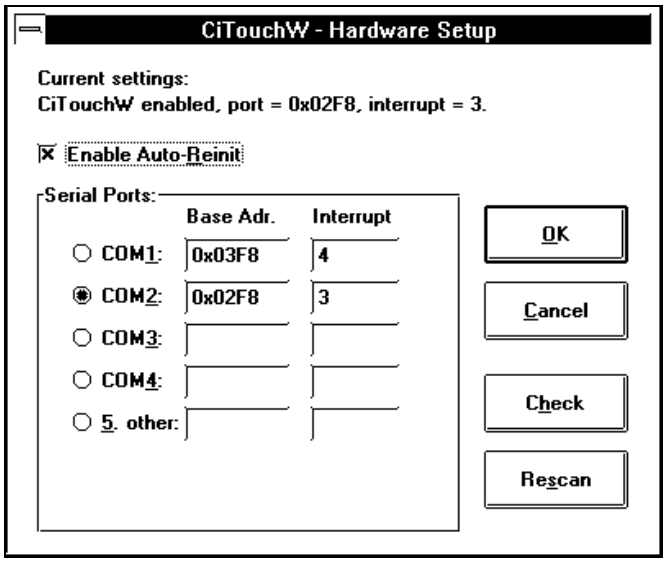


**Illustration 3-1, Dialogue window Hardware Setup**

Since a correctly connected IRT is detected automatically, the default values can usually be acknowledged by pressing the Enter key.
However, if in the line below „Current settings" the dialogue window shows the remark "No IRT found!", the parameters of the serial port need to be set manually. For additional help regarding this case refer to the chapter "Problems during the installation" at page 7. Since for a trouble-free operation the CiTouchW driver requires a correct assignment of COM ports to base addresses and Interrupt channels, all COM ports that are available in the computer need to be entered into the respective fields of this dialogue window. For a more detailed description of this dialogue window refer to chapter "Hardware" at page 25. If with the entered parameters an IRT can be initialized successfully, a restart of Windows, if required, is carried out.

## 3.3 Calibration of the IRT

In order to conclude the installation, it is required to calibrate the CiTouchW software. The calibration will leave the driver independent of the IRT's installation position with regard to the display. After the restart of Windows the calibration window is automatically displayed.

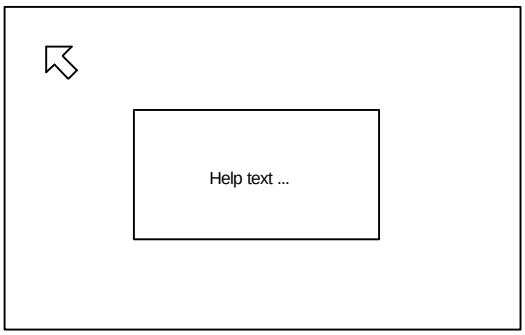


**Illustration 3-2, Calibration in the Flat Panel Mode**

The calibration window covers the entire screen. In its centre there is a help text and in the top left-hand side corner a cursor symbol can be found. Near this cursor symbol the touch zone needs to be interrupted. After the finger has then released the touch zone again, the cursor symbol will vanish and appear again at the bottom right-hand side corner. There the touch zone also needs to be interrupted

with the finger. Upon touching and releasing the touch zone a short beep is emitted from the computer's speaker. The position of the finger upon releasing the touch zone is used as the final calibration position. If an IRT is attached to a flat panel display, the touch zone and the active display area are assumed to accurately coincide with each other. In this case, at the calibration only the position of the IRT's coordinates origin with regard to the one of the display is determined. Therefore, the display needs not to be touched exactly at the cursor symbol. This way an easy yet accurate calibration is carried out. However, if either the IRT is attached to a CRT (cathode ray tube) screen or the sizes of the IRT and the display do not coincide with each other, a more precise calibration is required. To do so there is a second calibration mode available especially for CRT screens. To access this mode, the screen needs to be touched at the help text. The mode change is acknowledged by an acoustic signal. In the CRT calibration mode, the calibration spots are represented by a cross-wire. For a precise calibration to the available screen size and position, the touch zone has to be released from the exact centre of the cross-wire.

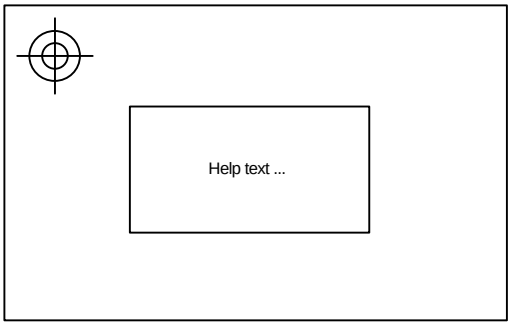


**Illustration 3-3, Calibration in the CRT Mode**

The calibration window closes after the touch zone was released at the second calibration spot. The installation of the CiTouchW software is now successfully finished.
However, the setup program does not return to the Windows desktop right away, but the main dialogue window of the CiTouchW configuration software is opened. As described in chapter „Handling" at page 11, here additional driver parameters can be set according to the user's requirements. For example, a possibly failed calibration can be repeated by selecting the button "Calibrations".

## *3.4 Problems during the installation*

This chapter outlines some hints for the trouble-shooting in case problems occur during the installation of the CiTouchW software. If in spite of these measures the installation still cannot be achieved successfully, please get in touch with us:

Citron GmbH
Steinerne Furt 61
86167 Augsburg
Germany

Fax: ++ 49 821 706914
Tel.: ++ 49 821 705005

Possible problems that may occur in the various installation steps are outlined in the following chapters.

### 3.4.1 Copying step

If the installation is interrupted in the course of copying CiTouchW files, the entire process needs to be repeated. This is the only way a correct installation can be achieved.
In case Windows does not start by itself after the CiTouchW files have been copied, the computer needs to be rebooted manually. Afterwards the installation process should proceed normally as described above. If it does not, the setup of the IRT hardware and the calibration of the CiTouchW need to be carried out manually. For a detailed description of the required procedures please refer to the chapters "Hardware" at page 25 and "Calibration of the IRT" at page 6.

### 3.4.2 Setup of the IRT hardware

If the Windows Program Manager includes an Autostart Group, after a restart of Windows not the "CiTouchW - Hardware Setup" dialogue window is active but the last application of the Autostart Group. In this case use the ALT+TAB key combination to change to the "CiTouchW - Hardware Setup".

If the IRT is not detected automatically, please check the following possible problem sources:

- Is the IRT correctly connected to a serial port of the computer?
- Is the power supply of the IRT turned on?
- Is the used serial port capable of Interrupts and was the correct ISA bus Interrupt number determined in the dialogue window?
  The "CiTouchW - Hardware Setup" assumes a standard PC configuration, i.e. COM1 = 0x3f8 - Interrupt 4 and COM2 = 0x2f8 - Interrupt 3. If the used serial port should not coincide with these parameters, the correct values need to be entered into the respective fields of the "CiTouchW - Hardware Setup" dialogue window.
- Are all serial ports correctly entered at the dialogue window Hardware Setup?
- If a LDVGA board is used: was the serial port (SIO1) activated by means of LDVINST and LDVDRV?
- Was the IRT already initialized by a preceding installation? In this case the automatic IRT detection fails. To solve this problem, interrupt either the connection between the IRT and the serial port of the computer or the IRT's power supply for a few seconds.

If the problem source could be eliminated, the automatic detection of the IRT can be repeated by means of the button "Rescan".

### 3.4.3 Temporary files

During the installation a temporary subdirectory as well as a number of temporary files is created. Normally, after a successfully finished installation the subdirectory and the files will automatically be deleted. If due to an error that occurred during the installation this deletion fails, these files can explicitly be deleted by the user, too. In detail the following files are concerned:

- Files in the Windows directory:
  _MSSETUP.BAT
  _MSRSTRT.EXE
  CTCTRL.EXE
  CTSETUP.INI

- Temporary subdirectory of the root directory of the drive at which Windows is installed:
  The subdirectory's name starts with the characters "~M". The remaining six characters of the directory's name represent a random combination of digits and characters. All files in this particular subdirectory as well as the directory itself may be deleted unhesitatingly.

# 4 Configuration of the CiTouchW driver

The CiTouchW driver comprises a number of setting possibilities. That way it can be adapted individually to the requirements of the respective user.
All parameters are centrally managed by an application in the program group Control Panel. To start this application, double-click on the CiTouchW icon in the Control Panel.

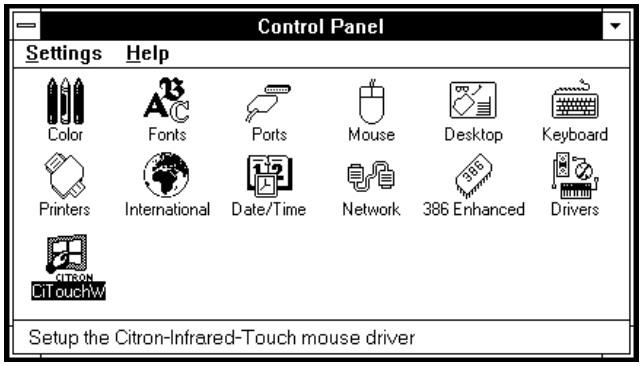


**Illustration 4-1, Control Panel**

The main dialogue window branches out to the various sections. Furthermore all essential parameters of the connected IRT are displayed.

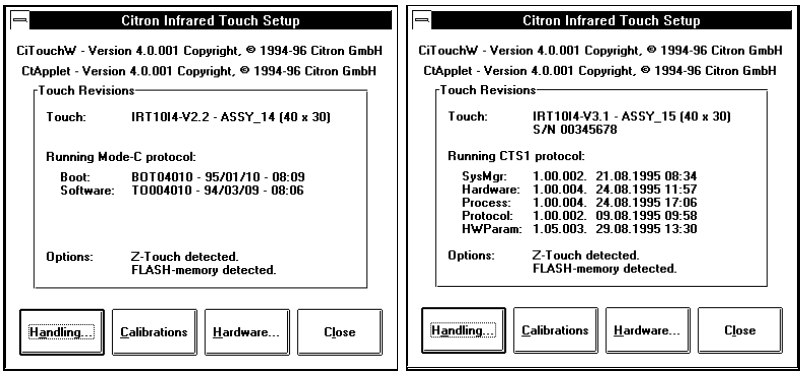


**Illustration 4-2, Main dialogue window with version number (left-hand side Mode-C, right-hand side CTS1)**

The first lines display the version numbers of the CITOUCHW.DRV driver program and the CTAPPLET.CPL System Control application. The box below contains information about the connected IRT. The information displayed here depend on the communication protocol of the connected IRT.

- **Touch:** Exact designation of the IRT hardware. Behind it the numbers in brackets represent the number of X- and Y-beams.
  If the IRT supports the CTS1 communication protocol, the serial number of the IRT is displayed, too.

- **Running Mode-C Protocol:**
  **Boot:** Designation, creation date (YY/MM/DD) and time (HH/MM) of the IRT boot program
  **Software:** Designation, creation date (YY/MM/DD) and time (HH/MM) of the IRT main program

- **Running CTS1 Protocol:**
  **SysMgr:** Version number, creation date (YY/MM/DD) and time (HH/MM) of the System Manager Module.
  **Hardware:** Version number, creation date (YY/MM/DD) and time (HH/MM) of the Hardware Module.

**Process:** Version number, creation date (YY/MM/DD) and time (HH/MM) of the Process Module.

**Protocol:** Version number, creation date (YY/MM/DD) and time (HH/MM) of the Protocol Module.

**HWParam:** Version number, creation date (YY/MM/DD) and time (HH/MM) of the Hardware Parameter Module.

- **Options:** Available memory of the IRT: *"FLASH detected"* or *"EPROM detected"*. If the connected IRT is equipped with pressure sensors the text *"Z-Touch detected"* appears and in the following dialogue windows the Z-axis options are activated.

The buttons situated at the bottom edge of the dialogue window branch out to the several sections to set the CiTouchW driver parameters:

- **Handling:** Sets the Handling of the CiTouchW driver. The Handling comprises the mouse button emulation, the coordinates calculation, various changes made at the Windows desktop as well as the extended driver settings.
- **Calibrations:** Modifies the calibration of the CiTouchW software at any time. For a detailed description about the calibration process please refer chapter "Calibration of the IRT" at page 6.
- **Hardware:** Sets the serial port parameters of the CiTouchW driver.
- **Close:** Closes the dialogue window and returns to the System Control.

Each dialogue window can also be operated via a keyboard without the help of a mouse or the IRT. Just like under Windows, the respective hot-key is emphasized by an underlined letter. The function is then called up by simultaneously pressing the ALT key and the corresponding hot-key. Pressing the ESC key cancels the action, pressing either the CARRIAGE RETURN or the ENTER key carries out the function of the emphasized button (default button).

## *4.1 Handling*

In order to keep the handling of the CiTouchW driver versatile there are numerous parameters to be set. Since it is not possible to display all parameters on a screen with 640x480 pixels, this dialogue window is divided into several pages. All Handling dialogue pages have the menu displayed below in common.

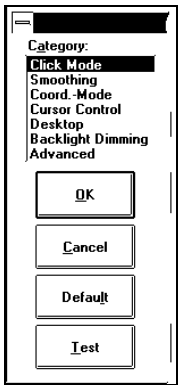


**Illustration 4-3, Common menu of the Handling dialogue pages**

In the **"Category"** list the dialogue pages are selected:

- **Click Mode:** Configures the mouse button emulation.
- **Smoothing:** Sets the smoothing factors for the coordinates calculation.
- **Coord.-Mode:** Sets the mode for the coordinates calculation and the change between absolute and relative coordinates.
- **Cursor Control:** Sets the cursor position in relation to the touch spot and the cursor acceleration.
- **Desktop:** Changes various parameters of the Windows desktop.
- **Backlight Dimming:** Determines the level of background illumination at active and inactive Touch Saver.
- **Advanced:** Sets the extended parameters for the operation of the IRT. Usually these settings do not need to be changed by the user.

Changes of the driver parameters will not have any effect to the CiTouchW function until they are either acknowledged and therewith permanently saved by means of the button **"OK"** or temporarily activated by means of the button **"Test"**. The button **"Cancel"** rejects all changes made and returns to the main dialogue window.
The button **"Default"** resets all CitouchW parameters to their default values. To do so, the files "CTW300.INI" and "CTA300.INI" are renamed in "CTW300.nnn" or "CTA300.nnn", respectively. The term "nnn" represents a consecutive numbering like "000", "001", "002", … . However, except for the baud rate and the most recently detected communication protocol neither the calibration nor the serial port parameters are reset.

In the Test Mode the present driver status is saved and the changes to be made are temporarily activated. The Test Mode dialogue window displayed below appears. If the Test Mode is closed either by means of the button **"Close"** or the **ESC** key, the saved driver status is restored. Therefore it is possible to test various settings without any risk.

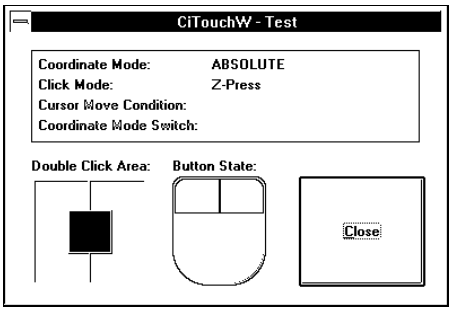


**Illustration 4-4, Test Mode**

The uppermost line of the Test Mode dialogue window displays the current coordinates mode that can either be "ABSOLUTE" or "RELATIVE".
If the corresponding Handling dialogue page was already called up beforehand, the lines below show the condition for a mouse click (**"Click Mode"**), for a cursor movement (**"Cursor Move Condition"**) and for the change between absolute and relative coordinates (**"Coordinate Mode Switch"**).
The field **"Double Click Area"** is divided into two halves. The left-hand side half changes its colour upon each valid double-click with the left-hand side mouse button, the right-hand side half accordingly upon each valid double-click with the right-hand side mouse button. In the centre of the "Double Click Area" its present size is displayed. In order for a double click to be recognized by Windows as such, the second mouse click has to occur within an area of this size around the first mouse click.
Underneath the headline **"Button State"** a mouse symbol is sketched. Its two mouse button symbols change their colour upon each click with the respective left - or right-hand side mouse button. The border line for the cursor acceleration to set in is represented by a red and white frame on the desktop.

### 4.1.1 Click Mode

At the first dialogue page the mouse button emulation is configure. There are a number of pre-defined as well as three user-definable key emulations. The currently active mouse button emulation is graphically displayed at the top right-hand side corner of the window. According to the selected emulation mode the required scroll bars and buttons required appear at the lower right-hand side area of the window.

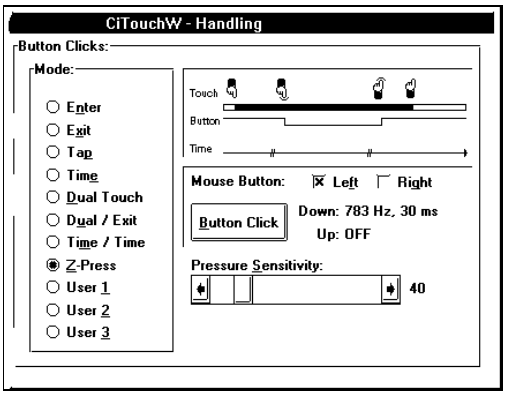


**Illustration 4-5, Dialogue page to select the mouse button emulation**

At the centre of the dialogue window the mouse button to be emulated is marked. Furthermore the current settings of the mouse button clicks are set. The button **"Button Click"** starts a dialogue window that determines both pitch and duration of the beep appearing upon a mouse click.

- **Left Button:** If this quadrangular button is marked, the left-hand side mouse button is emulated
- **Right Button:** If this quadrangular button is marked, the left-hand side mouse button is emulated

The pre-defined Mouse Button Emulation Modes are selected in the box **"Click Mode"**. In the following each emulation mode is individually described.

### Enter

Description: As soon as coordinates are reported, the emulated mouse button is pressed. The key remains pressed until the Touch Zone is released.

Parameters: none

Advantages:
- Accurate timing of the mouse click
- Easy handling
- Cursor motion at pressed mouse button possible

Disadvantages:
- Inaccurate positioning of the mouse click when absolute coordinates are used
- Relative coordinates not useful
- Little safety regarding faulty handling

### Exit

Description: As soon as the Touch Zone is released, a short mouse click occurs.

Parameters: none

Advantages:
- Accurate timing of the mouse click
- Easy handling
- Accurate positioning of the mouse click

Disadvantages:
- Relative coordinates not useful
- Cursor motion at pressed mouse button not possible

### Tap

Description: A mouse click is emulated if the Touch zone is released and then interrupted again within a preset time span. This process is called "Tap". The key remains pressed until the Touch Zone is again released.

Parameters: **Tap Time (t)**: Time span in which the IRT has to be interrupted again in order to create a Tap. The Tap Time can be set in steps of 55 ms between 0 ms and 2 s.

Advantages:
- Efficient safety regarding faulty usage
- Cursor motion at pressed mouse button possible

Disadvantages:
- Inaccurate positioning of the mouse click when absolute coordinates are used
- Relatively complicated handling

### Time

Description

A mouse click is emulated if the Touch Zone is interrupted and no mouse motion occurs for a time span T1. The key remains pressed until the Touch Zone is released. If within a time span T2 the Touch Zone is interrupted again, a second mouse click takes place immediately. This way a double-click can be created.

Parameters

**Time to Click (t1)**: After this time has elapsed, the first mouse click occurs. T1 can be set in steps of 55 ms between 0 ms and 2 s.

**Time to Idle (t2)**: If within this time span the Touch Zone is interrupted again, the second mouse click takes place immediately. T2 can be set in steps of 55 ms between 0 ms and 2 s.

Advantages

- Easy handling
- Cursor motion at pressed mouse button possible

Disadvantages

- Inaccurate timing of the mouse click
- Little safety regarding faulty handling

### Dual Touch

Description

If the Touch Zone is interrupted and a second interruption occurs simultaneously, a mouse click is emulated. The mouse button remains pressed for the duration of this dual touch.

Parameters

**Dual Touch Skip Count (n)**: Determines the number of dual touch messages to be skipped before the mouse click is emulated. The number of messages to be skipped can be set between 0 and 255.

Advantages

- Easy handling
- Accurately localized and timed positioning of the mouse click

Disadvantages

- Little safety regarding faulty handling
- Cursor motion at pressed mouse button not possible

### Dual / Exit

Description

If the Touch Zone is interrupted and a second interruption occurs simultaneously, a mouse click is emulated. Contrary to the "Dual Touch", the mouse button remains pressed until the Touch zone is completely released, i.e. both interruptions are cleared.

Parameters

**Dual Touch Skip Count**: Determines the number of dual touch messages to be skipped before the mouse click is emulated. The number of messages to be skipped can be set between 0 and 255.

Advantages

- Easy handling
- Accurately localized and timed positioning of the mouse click
- Cursor motion at pressed mouse button possible

Disadvantages

- Little safety regarding faulty handling
- Creating double-clicks is very difficult

### Time / Time

Description: A mouse click is emulated if the Touch Zone is interrupted and no mouse motion occurs for a time span T1. If the cursor continues to be motionless, after time span T2 has elapsed the mouse button is briefly released and then immediately pressed again.

Parameters:
- **Time to Click (t1)**: After this time has elapsed, the first mouse click occurs. T1 can be set in steps of 55 ms between 0 ms and 2 s.
- **Time to Second Click (t2)**: After this time span has elapsed, the second mouse click and further ones take place until the Touch Zone is released. T2 can be set in steps of 55 ms between 0 ms and 2 s.

Advantages:
- Easy handling
- Easy and accurately positioned creation of double-clicks

Disadvantages:
- Inaccurate timing of the mouse click
- Little safety regarding faulty handling

### Z-Press

Description: This particular item is only available if the IRT is equipped with pressure sensors. As soon as the pressure exerted onto the front screen exceeds a predetermined limit value, a mouse click is emulated. The key remains pressed until this pressure minus a hysteresis, falls below the limit value again.

Parameters:
- **Pressure Sensitivity**: Limit value of the pressure exerted onto the front screen, adjustable between 0 and 255. The actual pressure intensity depends on the installation of the IRT.

Advantages:
- Easy handling
- Cursor motion at pressed mouse button possible
- Efficient safety regarding faulty handling

Disadvantages:
- The IRT needs to be equipped with pressure sensors

### User 1 ... User 3

Description: There are three user-definable emulation modes. The programming is started by means of the button **"Define User"**. Instead of the graphical display, the name of the current user-definable mode is shown at the top right-hand side corner of the dialogue window. For a description of the user-definable programming please refer to the next chapter.

Parameter: none

### 4.1.2 Mouse clicks

The CiTouchW driver is capable of emitting a beep onto the PC speaker any time an emulated mouse button is either pressed or released. Both beep pitch and duration are set in the dialogue window as shown in Illustration 4-6. This dialogue window is started by means of the button **"Button Click"** at the dialogue page **"Click Mode"**.

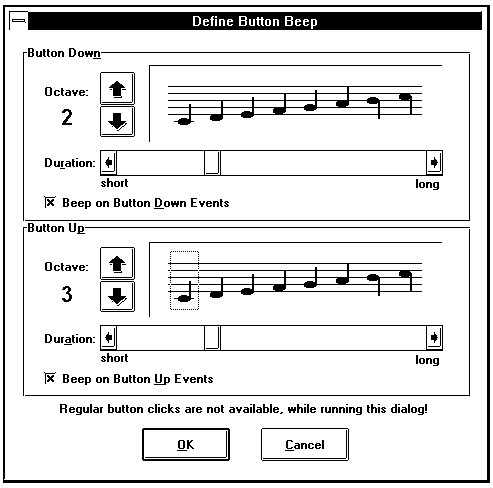


**Illustration 4-6, Definition of mouse button beep**

The beep generation for pressing a mouse button is set at **"Button Down"**, the one for releasing a mouse button at **"Button Up"**. Since both functions comprise the identical elements, in the following there is just one of the functions described in details.
The beep pitch is determined by selecting a note symbol and setting an octave. The currently selected note is coloured blue instead of black. The octave is set within a range of 0 to 5 by means of the two arrow symbols. The beep pitch can vary between 130 Hz and 8372 Hz.
The beep duration is set by means of the scroll bar **"Duration"**. It ranges between 10 ms and 577 ms. The setting is carried out in 10 logarithmically graded steps.
If a note symbol is selected, the corresponding tone is played back as soon as the mouse button is released. Therefore, as long as this dialogue window is displayed no regular mouse clicks are generated.
Whether or not a mouse click is to be generated at all is determined separately for pressing a mouse button by means of **"Beep on Button Down Events"** and for releasing a mouse button by means of **"Beep on Button Up Events"**.
The new settings are only accepted if the Handling dialogue window is eventually terminated with **"OK"**. In order to test the new settings beforehand, the Test Mode described earlier can be used.

### 4.1.3 User-defined key emulation

In the CiTouchW driver mouse button events are created by the so-called Button-Machine. This Button-Machine represents a programmable, asynchronous state machine. The transition from one state to the next one takes place as soon as all the required conditions are met. In the dialogue window for the programming of the user-defined key emulation the simplified state-transition diagram of this Button-Machine is displayed.
After the initialization of the driver, the Button-Machine is in state "IDLE". All mouse buttons are in the released condition. The various states are stepped through in the direction of the arrows.
In the states "T1", "T2" and "T3" a time span can be determined that needs to elapse before the respective state can be left. In addition to the time that has to elapse, to leave state "T1" the next occurring condition in the direction of the arrow as shown in the following illustration needs to be met, too. Contrary to T1, for the states "T2" and "T3" to step to the following status it is sufficient to either meet the declared condition or wait for the time to elapse.
Upon entering the state "TRIGGER", a mouse click is emulated. Upon transition to the state "T3", the mouse button is released again.

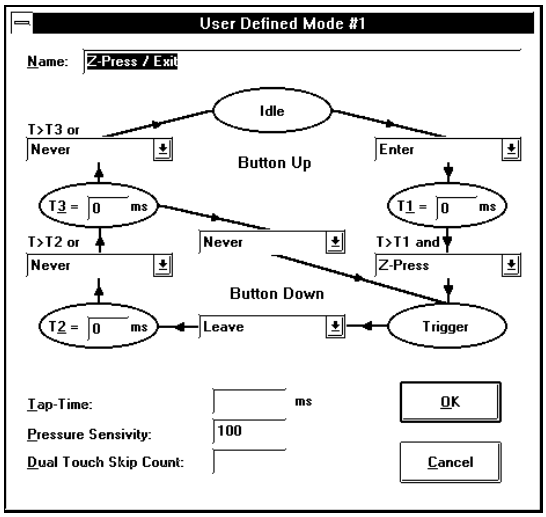


**Illustration 4-7, Dialogue window for the programming of user-defined key emulations**

Possible conditions for state transitions:

- **Never:** This condition never occurs
- **Immediately:** This condition occurs always and immediately
- **Enter:** Coordinates are reported
- **Z-Press:** The pressure limit value was exceeded
- **Dual Touch:** Dual touching was detected
- **Tap:** A "Tap" was detected
- **Leave:** There are no more coordinates reported
- **Z-Release:** The pressure fell below the limit value again
- **No Dual Touch:** There is no more dual touching detected

Illustration 4-8 shows the exact state-transition diagram of the Button-Machine:

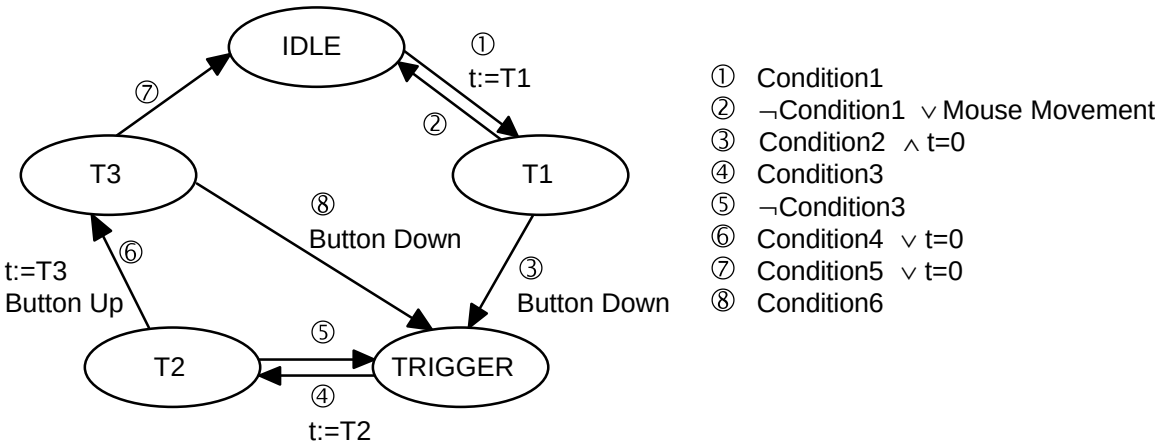


**Illustration 4-8, State machine for the mouse button emulation**

The key emulation mode **"Time"**, for instance, can be programmed as follows:

T1 = 550 ms
T2 = 0 ms
T3 = 550 ms
Condition1 = "Enter"
Condition2 = "Enter"
Condition3 = "Leave"
Condition4 = "Never"
Condition5 = "Never"
Condition6 = "Enter"

Another example is a new emulation mode called **"Z-Press / Exit"**. Similar to the mode "Dual / Exit", the release of the mouse button does not already happen when the pressure falls below the limit value but only after the Touch zone has been completely released. The parameters for this mode are:

T1 = 0 ms
T2 = 0 ms
T3 = 0 ms
Condition1 = "Enter"
Condition2 = "Z-Press"
Condition3 = "Leave"
Condition4 = "Never"
Condition5 = "Never"
Condition6 = "Never"

### 4.1.4 Smoothing

When using absolute coordinates, in the video mode the resolution of the IRT in the Mode-C communication protocol is not sufficient to activate each pixel of the screen. At a screen resolution of 640 x 480 pixels, one IRT coordinates change equals approximately 8 pixels. If the touch spot is now just at the transition of one coordinate to the other, the cursor continuously jumps back and forth a few pixels. In order to prevent this annoying effect, a smoothing factor can be set. To do so, an average value of a certain number of IRT coordinates messages is created. The number of coordinates messages to be taken into consideration for the calculation of this average value can be determined for the X- and Y-axis separately.

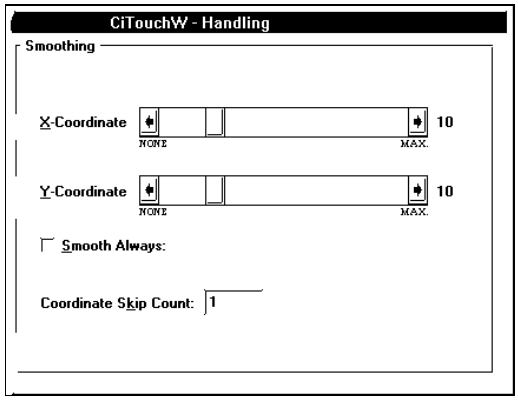


**Illustration 4-9, Dialogue page for the setting of the smoothing factor**

The several operational elements of this dialogue page are:

- **X-Coordinate:** Smoothing factor of the X-coordinate, setting range from 0 to 49
- **Y-Coordinate:** Smoothing factor of the Y-coordinate, setting range from 0 to 49
- **Smooth Always:**
  After each new interruption of the Touch Zone the creation of the average value is started anew. That means, at first the cursor is positioned directly onto the touch spot. The average value is created of additional cursor motions only.
  Is this quadrangular button is marked, however, after leaving the Touch Zone the old average value remains. If the Touch Zone is interrupted a second

time, according to the determined smoothing factor the cursor moves from its old position step by step towards the new touch spot.

- **Coordinate Skip Count:**
  Determines the number of coordinates messages of the IRT to be skipped before a new cursor position is reported. Skipping the first coordinates messages may prove useful, for example, if the IRT is installed rather far away from the screen surface. In that case the IRT would already detect a valid interruption before the finger actually touches the screen surface. On its way between the interruption of the detection and the screen surface the finger generally shifts. The user, however, expects a change of the cursor position only when he actually touches the screen surface. By skipping the first coordinates messages the expected behaviour can be achieved.

### 4.1.5 Coordinates Mode (Coord.-Mode)

The CiTouchW driver is able to work with either absolute or relative coordinates. In case absolute coordinates are used, the cursor jumps directly onto the spot of the interruption of the Touch Zone. In case relative coordinates are used, however, the cursor always moves relatively to its present position. The direction of the cursor motion corresponds to the direction of the moving interruption spot. Since the distance covered by the cursor may be smaller than the one of the actual finger movement, a resolution can be achieved that corresponds to the one of a conventional mouse.
It is possible during regular operation to change between the various coordinates modes. The section "**Mode Switch**" determines the condition of a change between the absolute and the relative coordinate's mode to take place. According to the selected event, at the right-hand side of the box a scroll-bar for the setting of the respective parameter (such as the "Tap Time", for example) appears.

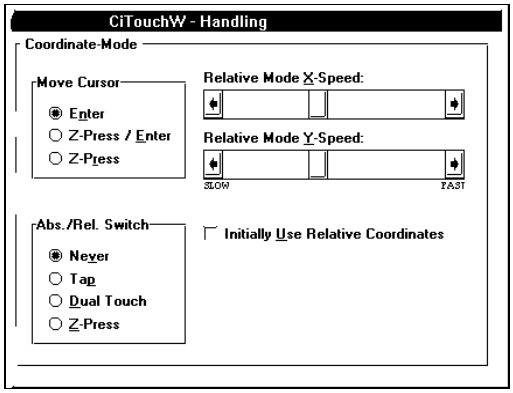


**Illustration 4-10, Dialogue page to set the coordinates mode**

The several operational elements of this dialogue page are:

- **Relative Mode X-Speed:**
  By means of this scroll bar the horizontal speed of the cursor at the usage of relative coordinates is set. Positions toward the left-hand side of the centre represent a slower speed with a higher resolution, positions toward the right-hand side of the centre represent a higher speed with a lower resolution.

- **Relative Mode Y-Speed:**
  By means of this scroll bar the vertical speed of the cursor at the usage of relative coordinates is set. Positions toward the left-hand side of the centre represent a slower speed with a higher resolution, positions toward the right-hand side of the centre represent a higher speed with a lower resolution.

- **Initially Use Relative Coordinates:**
  If this quadrangular button is marked, after the initialization of the CiTouchW driver relative coordinates are used. If it is not marked, the cursor motion is based on absolute coordinates.

**Move Cursor:**

This section determines the condition for the cursor position to change.

- **Enter:** A new cursor position is reported as soon as a valid interruption of the Touch Zone occurs.
- **Z-Press / Enter:** This option is only available if the IRT is equipped with pressure sensors. In order for a new cursor position to be reported after an interruption, the preset limit value of the pressure exerted onto the front screen needs to be exceeded too. For additional changes of the cursor position it is sufficient that the Touch Zone remains interrupted.
- **Z-Press:** This option is only available if the IRT is equipped with pressure sensors. New cursor positions are only reported as long as the preset limit value of the pressure exerted onto the front screen is exceeded.

**Abs. / Rel. Switch:**

This section determines the condition for the coordinates mode to change between absolute and relative.

- **Never**: During regular operation a change between the absolute and the relative coordinates mode is not possible.
- **Tap:** At each "Tap" a change takes place between the absolute and the relative coordinates mode. For the description of a "Tap" please refer to page 13.
- **Dual Touch:** At each dual touching a change takes place between the absolute and the relative coordinates mode.
- **Z-Press:** This option is only available if the IRT is equipped with pressure sensors. If the preset limit value of the pressure exerted onto the front screen is exceeded, a change takes place between the absolute and the relative coordinates mode.

## 4.1.6 Cursor Control

On this dialogue page a distance between the present cursor position in relation to the touch spot and an accelerated cursor motion in relation to the finger can be set.
According to the installation of the IRT, it may not always be possible to reach the outermost edge of the screen with a finger. By means of the cursor acceleration of the CiTouchW driver, outside an area of adjustable size the cursor position moves faster than the finger towards the edge of the screen. This way the outermost edge of the screen can be reached at any rate. The cursor acceleration is exclusively used in conjunction with absolute coordinates.

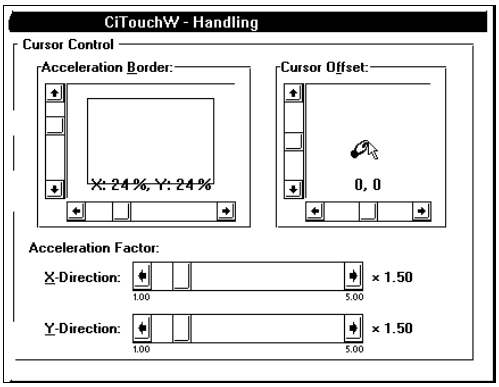


**Illustration 4-11, Dialogue page for the cursor control**

The several operational elements of this dialogue page are:

- **Acceleration Border :**
  By means of these two scroll bars the border is set beyond which the cursor acceleration sets in. The actual position of the acceleration border is displayed by a red frame. The numbers represent the distance of the border to the screen's edge in relation to the maximum possible distance (100% = centre of screen, 0% = edge of screen).

- **Cursor Offset:**
  By means of these two scroll bars the cursor position in reference to the actual touch spot is set. Regardless of the calibration of the CiTouchW driver, this distance (offset) is then kept. The field between the two scroll bars shows two symbols, a hand to indicate the touch spot position and an arrow to indicate the actual cursor position in relation to the touch spot. The numbers underneath the symbols represent the cursor offset in pixels.

- **Acceleration Factor:**
  By means of these two scroll bars the acceleration factor of the cursor motion is set. An acceleration factor of 2, for example, means that the cursor has already reached the screen's edge even though the finger only covered half of the distance between the acceleration border and the screen's edge.

### 4.1.7 Desktop Settings

At this dialogue page the various desktop settings can be changed. Changes regarding the desktop settings do not have an immediate effect on the CiTouchW driver, however improve the user-friendliness of Windows in conjunction with the CiTouchW driver.

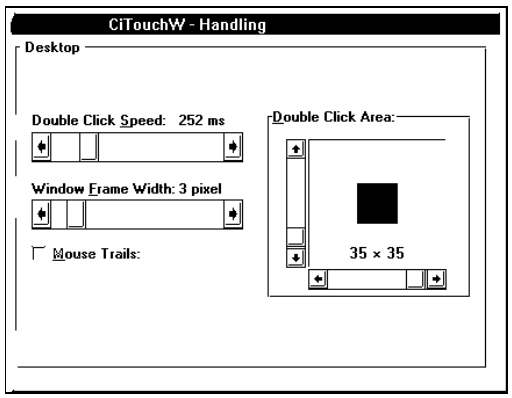


**Illustration 4-12, Dialogue page with the Windows desktop settings**

- **Double Click Area:**

  In order for a double click to be recognized by Windows as such, the second mouse click has to occur within a certain area around the first mouse click. This area is called the "Double Click Area". Due to the IRT's limited resolution it is rather difficult to hit exactly the same spot twice within a short time span. Therefore, the "Double Click Area" can be increased by means of the two scroll bars. The actual size of the "Double Click Area" is indicated by the black rectangle situated between the two scroll bars. The numbers underneath the rectangle represent the size of the "Double Click Area" in pixels.

- **Double Click Speed:**

  By means of this scroll bar the time span is set in which two mouse clicks have to occur in order to be recognized by Windows as a double click. The time can be set in steps of 16 ms between 100 ms and 900 ms.

- **Window Frame Width:**

  When absolute coordinates are used, it is rather difficult to precisely hit the frame of a window. Therefore, the frame width can be increased by means of this scroll bar.

- **Mouse Trails:**

  In order to improve the visibility of the cursor, most of the video drivers provide a mouse trail function. After a motion the cursor does not immediately disappear, but fades out slowly at the previous position. This option is turned on by marking this quadrangular button. In this case a scroll bar appears. By means of this scroll bar the number of cursors to be simultaneously displayed can be set between 1 and 7.

### 4.1.8 Control of the background illumination (Backlight Dimming)

The IRT comprises a "Touch Saver" function that is automatically activated if the Touch Zone was not interrupted for a certain adjustable period of time. If the Touch Saver is active, the scan rate of the beams is decreased, too. Therefore, with activated Touch Saver the IRT responds slower to Interruptions of the Touch Zone than with a deactivated Touch Saver. In addition to that the IRT comprises a PWM output that, for example, can be used to control the dimming level of the background illumination of TFT-displays.
For each of the two Touch Saver states, "active" and "inactive", the CiTouchW driver assigns a certain mark-to-space ratio and therewith a varying background illumination intensity to the PWM output. The activation time of the Touch Saver can be set on this dialogue page.

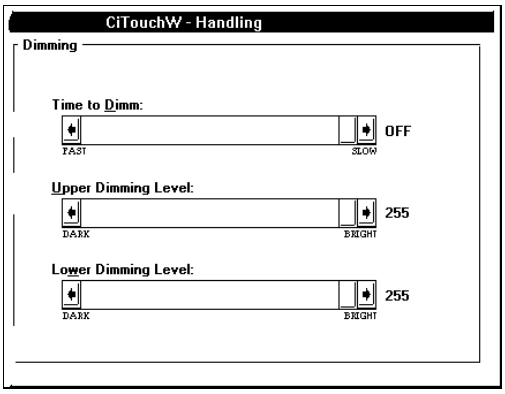


**Illustration 4-13, dialogue page for the control of the dimming level**

- **Time to Dimm:**
  This scroll bar sets the activation time of the Touch Saver in steps of 1 s. The minimum activation time equals 1 second, the maximum activation time 546 minutes and 6 seconds. The Touch Saver can be completely deactivated by shifting the scroll bar's slide all the way towards the right. However, if the IRT is operated in the Mode-C protocol, the activation time of the Touch Saver can only be set in steps of 30 seconds.

- **Upper Dimming Level:**
  This scroll bar sets the dimming level of the background illumination at deactivated Touch Saver (represents the normal operation). The "Upper Dimming Level" can be decreased to minimum value of 165 only. This way, during regular operation the background illumination intensity of a TFT display can never completely go out.

- **Lower Dimming Level:**
  This scroll bar sets the dimming level of the background illumination at activated Touch Saver. If the "Lower Dimming Level" is set to a value below 165, it may occur that the background illumination completely goes out after a certain period of time.

### 4.1.9 Advanced settings

At this dialogue page the IRT can be optimized with regard to the used computer. Changing these settings, however, requires a detailed understanding about the function of the IRT as described in its User's Manual. Normally it is not necessary to change these parameters. Therefore, the several input fields are just briefly explained here.

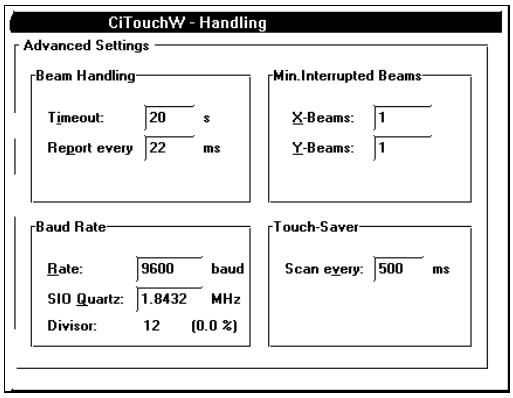


**Illustration 4-14, Dialogue page of the advanced settings**

The several operational elements of this dialogue page are:

**Beam Handling:**

- **Timeout:** If an interruption of any IRT's beam lasts longer than the time span determined here in seconds, it is excluded from the coordinates calculation.
- **Report every:** This entry determines the number of coordinates messages to be reported by the IRT upon a valid interruption of a beam.

**Min. Interrupted Beams:**

- **X-Beams:** This entry determines the number of X-beams that have to be interrupted simultaneously in order to be detected by the IRT as a valid interruption.
- **Y-Beams:** This entry determines the number of Y-beams that have to be interrupted simultaneously in order to be detected by the IRT as a valid interruption.

**Baud Rate:**

- **Rate:** This entry sets the baud rate for the serial data transmission between IRT and computer. If transmission errors should occur rather often, the baud rate should be reduced. Possible baud rates are 1200, 2400, 4800, 9600 and 19200 baud.
- **SIO-Quartz:** In case there is another quartz used in the computer for the generation of the baud rate than the usual 1.8432 MHz quartz, the correct quartz frequency in MHz needs to be entered here.
- **Divisor:** This field indicates the calculated divisor for the baud rate. Behind it in brackets the deviation of the actual baud rate from the nominal one is indicated in percent.

**Touch-Saver:**

- **Scan every:** If the Touch Saver is active, the scan rate is reduced to the time value entered here.

## *4.2 Hardware*

In order to establish a serial connection to the IRT, the CiTouchW driver requires several parameters that can be set in this dialogue window. Even after a successful installation, it is possible to change the interface parameters at any time. However, if parameters were changed, for its reinitialization the driver requires a restart of Windows.
The program tries to automatically detect all serial ports that are available in the computer. To do so, first the computer's BIOS is evaluated. This way up to 4 serial ports can be detected. However, the BIOS does not hold information about the respective Interrupts assigned to the interfaces. Therefore, the computers' common default settings are assumed:

COM1 = 0x3f8, Interrupt 4
COM2 = 0x2f8, Interrupt 3
COM3 = 0x3e8, no Interrupt
COM4 = 0x2e8, no Interrupt

Afterwards, the contents of the file SYSTEM.INI are searched for configuration data of serial ports. This way both the base address and the assigned Interrupt can be detected.
The information about the available serial ports gained that way, are provided to the user in four sets of two input fields each. If a serial port was either not detected at all or the automatic Interrupt assignment is wrong, the correct parameters need to be entered manually into the respective fields.

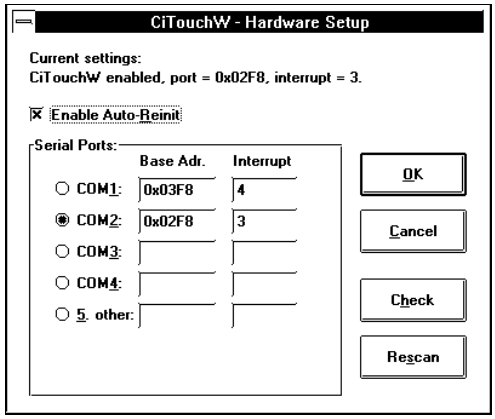


**Illustration 4-15, Dialogue window for setting the serial ports parameters**

The several operational elements of this dialogue page are:

- **Current Settings:** This line indicates the current settings of the CiTouchW driver. The following entries are possible:
  **"No IRT found":** At none of the found available serial ports an IRT was detected.
  **"IRT probably at port = ???, interrupt = ?.":** At the serial port with the stated parameters an IRT was detected. However, the CiTouchW driver is not initialized.
  **"IRT found, port = ???, interrupt = ?.":** In the file CTW300.INI an entry for serial port parameters with the stated values was found. However, no IRT could be initialized with these parameters.
  **"CiTouchW enabled, port = ???, interrupt = ?.":** The CiTouchW driver is initialized and there is an IRT connected to the serial port with the stated parameters.

- **OK:** Activating this button starts a process of reinitializing the CiTouchW driver with the set parameters. If this process fails, the dialogue window will not close. Otherwise, after a query is acknowledged, Windows is restarted.

- **Cancel:** Activating this button rejects all changes made and returns to the main dialogue window.

- **Check:** Activating this button starts a trial to reinitialize the IRT with the set parameters. The result of this trial is indicated in a message window.

- **Rescan:** When this button is activated the computer is again searched for serial ports and a connected IRT.

- **Enable Auto-Reinit:**
  If this quadrangular button is marked, the IRT is automatically linked again after the connection between IRT and computer was interrupted.

**Serial Ports:**

- **Base Adr.:** The base address of the detected serial port is either displayed in or entered into this column.

- **Interrupt:** The ISA bus Interrupt number of the serial port is either displayed in or entered into this column. A serial Interrupt is mandatory for the operation of the CiTouchW driver!

To which port the IRT is then eventually connected is determined by marking the circular button at the beginning of the respective line.

**Important**: For a satisfactory operation of the CiTouchW driver it is mandatory that both the base addresses and the Interrupt channels for all available serial ports in the computer are set correctly. Possible entries for physically non-existent ports must be deleted!

If a Citron LDVGA graphics board is used for the IRT connection and all 4 COM ports of the computer are occupied by other interface boards, the parameters used for the LDVGA board can be entered at item "5. other". The parameters of the other COM ports still have to be correct though!

# 5 CiTouchW-API

Besides functioning as a mouse driver, the CiTouchW driver additionally provides an interface to application programs. This interface enables both the setting of the entire scope of driver parameters and a direct access to the IRT.
To be able to use the CiTouchW-API in own application programs, both the file "CITOUCHW.LIB" needs to be linked up and the header file "CITOUCHW.H" needs to be embedded into the program. These two files can be found on the installation disk in the subdirectory "\API".

## *5.1 Summary of the API functions*

The following charts list a brief summary of all CiTouchW API functions in functional order.

### 5.1.1 Requesting CiTouchW parameters

By means of these commands all configuration parameters of the CiTouchW driver can be read out.

| Command | Returned data | Page |
|---|---|---|
| **citGetAcceleration** | Parameters for cursor acceleration | 29 |
| **citGetButtonBeep** | Parameters for mouse clicks | 30 |
| **citGetCalibrationAbs** | Calibration parameters for absolute coordinates | 30 |
| **citGetCalibrationRel** | Calibration parameters for relative coordinates | 30 |
| **citGetCommands** | Parameters for mouse button emulation | 30 |
| **citGetCoordMode** | Parameters for coordinates calculation | 31 |
| **citGetDimming** | Mark-to-space ratio of IRT's PWM output | 31 |
| **citGetDriverConstants** | Unchanging driver parameters | 31 |
| **citGetDriverSettings** | Variable driver parameters | 31 |
| **citGetFlags** | Current driver status | 32 |
| **citGetSerialHardware** | Parameters of serial interface | 32 |
| **citGetTouchHardware** | Unchanging parameters of the IRT | 32 |
| **citGetTouchSettings** | Variable parameters of the IRT | 32 |
| **citGetVersion** | Version number of CITOUCHW.DRV | 33 |
| **Inquire** | General mouse parameters (supported by any Windows-compatible mouse driver) | 38 |
| **MouseGetIntVect** | Used Interrupt vector (supported by any Windows-compatible mouse driver) | 38 |

### 5.1.2 Changing CiTouchW parameters

By means of these commands all configuration parameters of the CiTouchW driver can be changed.

| Command | Returned data | Page |
|---|---|---|
| **citSetAcceleration** | Parameters for cursor acceleration | 35 |
| **citSetButtonBeep** | Parameters for mouse clicks | 35 |
| **citSetCalibrationAbs** | Calibration parameters for absolute coordinates | 35 |
| **citSetCalibrationRel** | Calibration parameters for relative coordinates | 36 |
| **citSetCommands** | Parameters for mouse button emulation | 36 |
| **citSetCoordMode** | Parameters for coordinates calculation | 36 |
| **citSetDimming** | Mark-to-space ratio of IRT's PWM output | 37 |
| **citSetDriverSettings** | Unchanging driver parameters | 37 |
| **citSetSerialHardware** | Parameters of serial interface | 37 |
| **citSetTouchSettings** | Unchanging parameters of the IRT | 38 |
| **citReloadIni** | Complete reinitialization of the driver | 34 |

### 5.1.3 Communication with the IRT

The CiTouchW driver provides an easy-to-handle interface for both transmitting commands to the IRT and receiving of reports from the IRT.
Commands can be transmitted by means of the function **citSend()** at any time. However, before an application program is able to communicate with the IRT, the receive channel needs to be opened by means of the function **citOpen()**. However, a CiTouchW driver with an open receive channel does not report cursor motions or mouse clicks anymore. Therefore it is essential not to forget the call-up of **citClose()** after having received the desired reports! Only complete reports are sent from the IRT to the application program. This way it is not necessary for the application program having to recognize reports' limits. However, only those IRT functions can be used that do not require a new initialization of the IRT. Therefore, especially a reprogramming of the IRT's FLASH Memory is not possible.
There are two methods provided for the reception of IRT reports: *Polling* and *Notification*.
At the *Polling* the function **citReceiveStatus()** has to request continuously whether there is a complete report from the IRT available. Is this the case, the report can be read by means of the function **citReceive()**.
At the *Notification*, however, a message is sent to the application program as soon as a complete report was received. Both the number of this message and the window handle of the application window that is to receive the message are determined by the user when calling up the function **citOpen()**. At page 43 the structure of the notification is described.
If there is another report received by the IRT although the receive buffer was not read-out yet, the newly received report is disregarded. This guarantees that an explicitly requested report cannot be overwritten by subsequent coordinates messages of the IRT.
Possible commands for the IRT and the structure of IRT reports vary between the Mode-C communication protocol and the CTS1 protocol. The respective commands are described in the user's manual of the IRT.

| Command | Function | Page |
|---|---|---|
| **citCheckBreak** | Checks a serial port for 100ms Breaks | 29 |
| **citClose** | Closes the receive channel | 29 |
| **citOpen** | Opens the receive channel between IRT and computer | 33 |
| **citReceive** | Receives a complete report from the IRT | 34 |
| **citReceiveStatus** | Determines the status of the receive channel | 34 |
| **citSend** | Transmits one byte to the IRT | 34 |

### 5.1.4 General help functions

Besides the interfaces required for the mouse button emulation, the CiTouchW driver also provides several useful functions.

| Command | Function | Page |
|---|---|---|
| **citDetectLDVGA** | Checks whether an LDVGA graphics board exists | 29 |
| **citPlaySound** | Emits a beep to the PC speaker | 33 |

## *5.2 Reference of the extended API functions*

In the following the structure of a reference entry is demonstrated:

### Example

| | |
|---|---|
| Prototype: | **Function prototype** |
| Parameter: | Description of the parameters |
| Response: | Description of the return parameter |
| Description: | Detailed description of the function |

## citCheckBreak

Prototype: **BOOL WINAPI citCheckBreak(WORD wPort)**

Parameter:
WORD wPort
Base address of serial port to be tested

Response:
TRUE There were 100ms Breaks detected
FALSE There were no Breaks detected

Description: A not initialized IRT transmits BREAK signals in time intervals of 100 ms. By means of this function a serial port can be searched for these signals.

---

## citClose

Prototype: **BOOL WINAPI citClose(void)**

Parameter: none

Response:
TRUE Receive channel could be closed
FALSE Receive channel could not be closed. Possible causes for that are either an already closed receive channel or a driver that was not initialized.

Description: For the CiTouchW driver to restore its normal function as a mouse driver, the receive channel, that was opened earlier to be able to receive reports, has now to be closed again.

---

## citDetectLDVGA

Prototype: **BOOL WINAPI citDetectLDVGA(void)**

Parameter: none

Response:
TRUE A Citron LDVGA board was detected
FALSE There is no Citron LDVGA board available.

Description: This function detects whether a Citron Long-Distance-VGA board is available.

---

## citGetAcceleration

Prototype: **WORD WINAPI citGetAcceleration(LPACCELERATION lpAC)**

Parameter:
LPACCELERATION lpAC
Pointer onto an ACCELERATION structure for the returning of the parameters for the cursor acceleration.

Response: Number of bytes written into ACCELERATION structure

Description: The parameters for the cursor acceleration are written into the ACCELERATION structure that was provided. For a description of this structure, please refer to page 39.

## citGetButtonBeep

Prototype: **WORD WINAPI citGetButtonBeep(LPBUTTONBEEP lpBB)**

Parameter: LPBUTTONBEEP lpBB
Pointer onto a BUTTONBEEP structure for the returning of mouse click parameters.

Response: Number of bytes written into BUTTONBEEP structure

Description: The parameters for the creation of a mouse click are written into the BUTTONBEEP structure that was provided. For a description of this structure, please refer to page 39.

---

## citGetCalibrationAbs

Prototype: **WORD WINAPI citGetCalibrationAbs(LPCALIBRATIONABS lpCA)**

Parameter: LPCALIBRATIONABS lpCA
Pointer onto a CALIBRATIONABS structure for the returning of the absolute calibration parameters.

Response: Number of bytes written into the CALIBRATIONABS structure

Description: The calibration parameters of the absolute coordinates mode are written into the CALIBRATIONABS structure that was provided. For a description of this structure, please refer to page 40.

---

## citGetCalibrationRel

Prototype: **WORD WINAPI citGetCalibrationRel(LPCALIBRATIONREL lpCR)**

Parameter: LPCALIBRATIONREL lpCR
Pointer onto a CALIBRATIONREL structure for the returning of the relative calibration parameters.

Response: Number of bytes written into the CALIBRATIONREL structure

Description: The calibration parameters of the relative coordinates mode are written into the CALIBRATIONREL structure that was provided. For a description of this structure, please refer to page 40.

---

## citGetCommands

Prototype: **WORD WINAPI citGetCommands(LPCOMMANDS lpCMD)**

Parameter: LPCOMMANDS lpCMD
Pointer onto a COMMANDS structure for the returning of the mouse button emulation parameters.

Response: Number of bytes written into the COMMANDS structure

Description: The parameters for the mouse button emulation are written into the COMMANDS structure that was provided. For a description of this structure, please refer to page 40. For a description of the function of the mouse button emulation refer to chapter "User-defined key emulation" at page 16.

## citGetCoordMode

Prototype: **WORD WINAPI citGetCoordMode(LPCOORDMODE lpCM)**

Parameter: LPCOORDMODE lpCM
Pointer onto a COORDMODE structure for the returning of the coordinates output parameters.

Response: Number of bytes written into the COORDMODE structure

Description: The parameters for the coordinates output are written into the COORDMODE structure that was provided. For a description of this structure, please refer to page 41.

---

## citGetDimming

Prototype: **WORD WINAPI citGetDimming(LPDIMMING lpDIM)**

Parameter: LPDIMMING lpDIM
Pointer onto a DIMMING structure for the returning of dimming parameters.

Response: Number of bytes written into DIMMING structure

Description: The parameters for the control of the PWM output of the IRT are written into the DIMMING structure that was provided. For a description of this structure, please refer to page 41.

---

## citGetDriverConstants

Prototype: **WORD WINAPI citGetDriverConstants(LPDRIVERCONSTANTS lpDC)**

Parameter: LPDRIVERCONSTANTS lpDC
Pointer onto a DRIVERCONSTANTS structure for the returning of driver constants.

Response: Number of bytes written into DRIVERCONSTANTS structure

Description: The fixed driver parameters are written into the DRIVERCONSTANTS structure that was provided. For a description of this structure, please refer to page 41.

---

## citGetDriverSettings

Prototype: **WORD WINAPI citGetDriverSettings(LPDRIVERSETTINGS lpDS)**

Parameter: LPDRIVERSETTINGS lpDS
Pointer onto a DRIVERSETTINGS structure for the returning of the driver parameters.

Response: Number of bytes written into the DRIVERSETTINGS structure

Description: The variable driver parameters are written into the DRIVERSETTINGS structure that was provided. For a description of this structure, please refer to page 42.

## citGetFlags

Prototype: **WORD WINAPI citGetFlags(void)**

Parameter: none

Response: Current status of the CiTouchW driver

Description: The current driver status is carried in the return word as bit flags. For the inquiry of single bit flags, in the file CITOUCHW.H there are constants defined with the prefix "IF_". With these constants the return word can be AND-operated. For a description of the various flags refer to the file CITOUCHW.H. This file also contains describing comments next to each definition.

---

## citGetSerialHardware

Prototype: **WORD WINAPI citGetSerialHardware(LPSERIALHARDWARE lpSH)**

Parameter: LPSERIALHARDWARE lpSH
Pointer onto a SERIALHARDWARE structure for the returning of the serial interface parameters.

Response: Number of bytes written into the SERIALHARDWARE structure

Description: The serial interface parameters are written into the SERIALHARDWARE structure that was provided. For a description of this structure, please refer to page 42.

---

## citGetTouchHardware

Prototype: **WORD WINAPI citGetTouchHardware(LPTOUCHHARDWARE lpTH)**

Parameter: LPTOUCHHARDWARE lpTH
Pointer onto a TOUCHHARDWARE structure for the returning of the unchanging IRT parameters.

Response: Number of bytes written into the TOUCHHARDWARE structure

Description: The unchanging IRT parameters are written into the TOUCHHARDWARE structure that was provided. For a description of this structure, please refer to page 43.

---

## citGetTouchSettings

Prototype: **WORD WINAPI citGetTouchSettings(LPTOUCHSETTINGS lpTS)**

Parameter: LPTOUCHSETTINGS lpTS
Pointer onto a TOUCHSETTINGS structure for the returning of the variable IRT parameters.

Response: Number of bytes written into the TOUCHSETTINGS structure

Description: The variable IRT parameters are written into the TOUCHSETTINGS structure that was provided. For a description of this structure, please refer to page 43.

## citGetVersion

Prototype: **WORD WINAPI citGetVersion(void)**

Parameter: none

Response: Version number of the CITOUCHW.DRV program

Description: This function should not be used anymore. To inquire the version number, the VERSIONINFO resource of the CITOUCHW.DRV driver should be read instead. To do so, Windows provides a number of help functions in the dynamic link library VER.DLL.

---

## citOpen

Prototype: **BOOL WINAPI citOpen(HWND hWnd, WORD wMsg)**

Parameter:

HWND hWnd
Either the handle of the window that is to receive the notification messages or NULL if no messages are to be sent.
WORD wMsg
Message to be sent as a notification to the window

Response:

TRUE, in case the receive channel could be opened.
FALSE, in case the receive channel could not be opened. Possible causes for that are either an already opened receive channel or a driver that was not initialized.

Description: Before an application program is able to receive reports from the IRT, the receive channel needs to be opened beforehand. However, a CiTouchW driver with an open receive channel does not report cursor motions or mouse clicks anymore. Therefore it is essential not to forget the call-up of **citClose ()** after having received the desired reports!

---

## citPlaySound

Prototype: **void WINAPI citPlaySound(WORD frequency, WORD duration)**

Parameter:

WORD frequency
Beep pitch in Hertz [Hz]
WORD duration
Duration in milliseconds [ms]

Response: none

Description: This function emits a beep with the determined pitch and duration to the PC speaker. The function returns as soon as the beep emission has started. This way the computer is not blocked for the duration of the beep.

## citReceive

Prototype: **WORD WINAPI citReceive(LPBYTE lpBuffer)**

Parameter:
LPBYTE lpBuffer
Pointer onto a buffer the received report is copied to.

Response: Status of the receive buffer

Description: By means of this function a report that was completely received by the IRT can be read out. If there is no report available it is not waited for but immediately returned with an according status code. The maximum required size for the receive buffer can be requested by means of the function **citGetDriverConstants()**.
Possible return values are defined as constants in the file CITOUCHW.H with the prefix "CRS_". The file also contains describing comments.
Reports in the CTS1 protocol are already decoded, i.e. they do not contain DC2/DC4 and SYN sequences anymore.

---

## citReceiveStatus

Prototype: **WORD WINAPI citReceiveStatus(void)**

Parameter: none

Response: Status of the receive buffer

Description: The current status of the receive buffer is requested. Possible return values are defined as constants in the file CITOUCHW.H with the prefix "CRS_". The file also contains describing comments.

---

## citReloadIni

Prototype: **BOOL WINAPI citReloadIni(void)**

Parameter: none

Response:
TRUE, if the driver could be reinitialized
FALSE, if the driver could not be reinitialized

Description: The file CTW300.INI is read anew and the CiTouchW driver is completely reinitialized.

---

## citSend

Prototype: **void WINAPI citSend(BYTE byData)**

Parameter:
BYTE byData
Data byte to be transmitted to the IRT

Response: none

Description: A byte is transmitted to the IRT. Supposed that the transmission buffer of the interface module is full, the transmission will wait until this module is empty.
Commands in the CTS1 protocol have to be encoded beforehand, i.e. the application program has to add the DC2/DC4 and SYN sequences.

## citSetAcceleration

Prototype: **BOOL WINAPI citSetAcceleration(LPACCELERATION lpAC)**

Parameter: LPACCELERATION lpAC
Pointer onto an ACCELERATION structure with the new parameters for the cursor acceleration.

Response: TRUE, if the new parameters could be set.
FALSE, if the new parameters could not be set.

Description: The parameters for the cursor acceleration are changed according to the ACCELERATION structure that was provided. For a description of this structure, please refer to page 39.
If the parameters could not be changed, the old status remains.

---

## citSetButtonBeep

Prototype: **BOOL WINAPI citSetButtonBeep(LPBUTTONBEEP lpBB)**

Parameter: LPBUTTONBEEP lpBB
Pointer onto a BUTTONBEEP structure with the new parameters for the mouse click creation.

Response: TRUE, if the new parameters could be set.
FALSE, if the new parameters could not be set.

Description: The parameters for the creation of a mouse click are changed according to BUTTONBEEP structure that was provided. For a description of this structure, please refer to page 39.
If the parameters could not be changed, the old status remains.

---

## citSetCalibrationAbs

Prototype: **BOOL WINAPI citSetCalibrationAbs(LPCALIBRATIONABS lpCA)**

Parameter: LPCALIBRATIONABS lpCA
Pointer onto a CALIBRATIONABS structure with the new absolute calibration parameters.

Response: TRUE, if the new parameters could be set.
FALSE, if the new parameters could not be set.

Description: The calibration parameters of the absolute coordinates mode are changed according to the CALIBRATIONABS structure that was provided. For a description of this structure, please refer to page 40.
If the parameters could not be changed, the old status remains.

## citSetCalibrationRel

Prototype: **BOOL WINAPI citSetCalibrationRel(LPCALIBRATIONREL lpCR)**

Parameter: LPCALIBRATIONREL lpCR
Pointer onto a CALIBRATIONREL structure with the new relative calibration parameters.

Response: TRUE, if the new parameters could be set.
FALSE, if the new parameters could not be set.

Description: The calibration parameters of the relative coordinates mode are changed according to the CALIBRATIONREL structure that was provided. For a description of this structure, please refer to page 40.
If the parameters could not be changed, the old status remains.

---

## citSetCommands

Prototype: **BOOL WINAPI citSetCommands(LPCOMMANDS lpCMD)**

Parameter: LPCOMMANDS lpCMD
Pointer onto a COMMANDS structure with the new parameters for the mouse button emulation.

Response: TRUE, if the new parameters could be set.
FALSE, if the new parameters could not be set.

Description: The parameters for the mouse button emulation are changed according to the COMMANDS structure that was provided. For a description of this structure, please refer to page 40. For a description of the function of the mouse button emulation refer to chapter "User-defined key emulation" at page 16.
If the parameters could not be changed, the old status remains.

---

## citSetCoordMode

Prototype: **BOOL WINAPI citSetCoordMode(LPCOORDMODE lpCM)**

Parameter: LPCOORDMODE lpCM
Pointer onto a COORDMODE structure with the new parameters for the coordinates output.

Response: TRUE, if the new parameters could be set.
FALSE, if the new parameters could not be set.

Description: The parameters for the coordinates output are changed according to the COORDMODE structure that was provided. For a description of this structure, please refer to page 41.
If the parameters could not be changed, the old status remains.

## citSetDimming

Prototype: **BOOL WINAPI citSetDimming(LPDIMMING lpDIM)**

Parameter: LPDIMMING lpDIM
Pointer onto a DIMMING structure with the new parameters for the control of the IRT's PWM output.

Response: TRUE, if the new parameters could be set.
FALSE, if the new parameters could not be set.

Description: The parameters for the control of the IRT's PWM output are changed according to the DIMMING structure that was provided. For a description of this structure, please refer to page 41.
If the parameters could not be changed, the old status remains.

---

## citSetDriverSettings

Prototype: **BOOL WINAPI citSetDriverSettings(LPDRIVERSETTINGS lpDS)**

Parameter: LPDRIVERSETTINGS lpDS
Pointer onto a DRIVERSETTINGS structure with the new driver parameters.

Response: TRUE, if the new parameters could be set.
FALSE, if the new parameters could not be set.

Description: The new driver parameters are changed according to the DRIVERSETTINGS structure that was provided. For a description of this structure, please refer to page 42.
If the parameters could not be changed, the old status remains.

---

## citSetSerialHardware

Prototype: **BOOL WINAPI citSetSerialHardware(LPSERIALHARDWARE lpSH)**

Parameter: LPSERIALHARDWARE lpSH
Pointer onto a SERIALHARDWARE structure with the new serial interface parameters.

Response: TRUE, if the new parameters could be set.
FALSE, if the new parameters could not be set.

Description: The serial interface parameters are changed according to the SERIALHARDWARE structure that was provided. For a description of this structure, please refer to page 42.
If the parameters could not be changed, the old status remains.

## citSetTouchSettings

Prototype: **BOOL WINAPI citSetTouchSettings(LPTOUCHSETTINGS lpTS)**

Parameter: LPTOUCHSETTINGS lpTS
Pointer onto a TOUCHSETTINGS structure with the new IRT parameters.

Response: TRUE, if the new parameters could be set.
FALSE, if the new parameters could not be set.

Description: The new IRT parameters are changed according to the TOUCHSETTINGS structure that was provided. For a description of this structure, please refer to page 43.
If the parameters could not be changed, the old status remains.

---

## Inquire

Prototype: **WORD WINAPI Inquire(LPMOUSEINFO lpMI)**

Parameter: LPMOUSEINFO lpMI
Pointer onto a MOUSEINFO structure for the returning of the mouse driver parameters.

Response: Number of bytes copied into the MOUSEINFO structure.

Description: This function is supported by any Windows-compatible mouse driver. For a description of the MOUSEINFO structure, please refer to page 42.

---

## MouseGetIntVect

Prototype: **WORD WINAPI MouseGetIntVect(void)**

Parameter: none

Response: The Interrupt vector used by CiTouchW

Description: This function is supported by any Windows-compatible mouse driver.

## *5.3 Reference of the structures*

In order to pass parameters to the API functions, in the file CITOUCHW.H there are numerous structures defined. In the following these structures are listed in alphabetical order.

### ACCELERATION

This structure contains the parameters for the cursor acceleration.

| Type | Name | Description |
|---|---|---|
| short | acMulX | Acceleration factor for X-coordinates |
| short | acMulY | Acceleration factor for Y-coordinates |
| WORD | acBorderX | Width of acceleration range (in Touch coordinates !) |
| WORD | acBorderY | Height of acceleration range (in Touch coordinates !) |

The items *acMulX* and ac*MulY* are signed fixed-point numbers. Their value range is calculated by the formulas

$$acMulX_{\max} = \frac{32767 * dcFixedBias}{caMulX}$$

$$acMulY_{\max} = \frac{32767 * dcFixedBias}{caMulY}.$$

The scaling factor *dcFixedBias* for the fixed-point numbers can be requested by means of **citGetDriverConstants()**. The items *caMulX* or *caMulY*, respectively, can be requested by means of **citGetCalibrationAbs()**.
The items *acBorderX* and *acBorderY* are declared in Touch coordinates. Their value range is

$$\mathrm{acBorderX}_{\max} = 32767$$
$$\mathrm{acBorderY}_{\max} = 32767$$

### BUTTONBEEP

This structure contains the parameters for the mouse button click creation.

| Type | Name | Description |
|---|---|---|
| WORD | bbFreqDown | Frequency of beep in Hertz [Hz] when key is pressed |
| WORD | bbTimeDown | Duration of beep in milliseconds [ms] when key is pressed |
| WORD | bbFreqUp | Frequency of beep in Hertz [Hz] when key is released |
| WORD | bbTimeUp | Duration of beep in milliseconds [ms] when key is released |
| WORD | bbClickMode | A combination of BC_??? constants that determines at what point of time a beep is created. |

## CALIBRATIONABS

This structure contains the parameters for the calibration of absolute coordinates. The formula that is used for the calibration of absolute coordinates represents a simple straight line equation.

$$y = mx + t$$

Windows expects coordinates with a value range from 0 up to 65535.

| Type | Name | Description |
|---|---|---|
| DWORD | caMulX | "*m*" for the X-coordinate |
| short | caAddX | "*t*" for the X-coordinate |
| DWORD | caMulY | "*m*" for the Y-coordinate |
| short | caAddY | "*t*" for the Y-coordinate |
| WORD | caOrientation | Orientation of the IRT in relation to the display. Corresponding to the position of the IRT's connecting plug in relation to the left-hand side top corner of the display, one of the OR_???? constants is used. |

The items *caMulX* and *caMulY* are unsigned fixed-point numbers. The scaling factor for the fixed-point numbers can be requested by means of **citGetDriverConstants()**.

## CALIBRATIONREL

This structure contains the parameters for the calibration of relative coordinates. The following formula is used for the calibration of relative coordinates:

$$y = x \cdot \frac{m}{d}$$

| Type | Name | Description |
|---|---|---|
| short | crMulX | "*m*" for the X-coordinate |
| WORD | crDivX | "*d*" for the X-coordinate |
| short | crMulY | "*m*" for the Y-coordinate |
| WORD | crDivY | "*d*" for the Y-coordinate |

## COMMANDS

This structure contains the parameters for mouse button emulation. The creation of a mouse click is described at the chapter "User-defined key emulation" at page 16.

| Type | Name | Description |
|---|---|---|
| WORD | cmdT1 | First time constant in steps of 1 ms |
| WORD | cmdT2 | Second time constant in steps of 1 ms |
| WORD | cmdT3 | Third time constant in steps of 1 ms |
| BYTE | cmdIdleT1 | Condition 1 |
| BYTE | cmdT1Trigger | Condition 2 |
| BYTE | cmdTriggerT2 | Condition 3 |
| BYTE | cmdT2UpT3 | Condition 4 |
| BYTE | cmdUpT3Idle | Condition 5 |
| BYTE | cmdUpT3Trigger | Condition 6 |
| BYTE | cmdModeChange | Condition for the change between absolute and relative coordinates. |
| BYTE | cmdReserved1 | reserved |

Possible values for the condition bytes are defined in the file CITOUCHW.H as constants with the prefix "BM_". This file also contains describing comments. For the field "cmdModeChange", the constants with the prefix "MC_" can be used.

## COORDMODE

The structure contains the parameters for the coordinates creation.

| Type | Name | Description |
|---|---|---|
| BOOL | cmCoordEnterZ | TRUE The first coordinates message after interrupting the Touch Zone requires the preset limit value for the pressure exerted onto the front screen to be exceeded.<br>FALSE The first coordinates message immediately occurs after interrupting the Touch Zone. |
| BOOL | cmCoordSignalZ | TRUE Additional coordinates messages also require the preset limit value for the pressure exerted onto the front screen to be exceeded.<br>FALSE For additional coordinates messages it is sufficient that the Touch Zone remains interrupted. |

## DIMMING

The structure contains the parameters for the control of the PWM output of the IRT.

| Type | Name | Description |
|---|---|---|
| WORD | blDimmingHigh | Mark-to-space ratio at deactivated Touch Saver |
| WORD | blDimmingLow | Mark-to-space ratio at activated Touch Saver |
| BOOL | blSaverActive | TRUE at activated Touch Saver |

The values for *blDimmingHigh* and *blDimmingLow* range from 0 up to 255. If both values equal 0, the mark-to-space ratio is set to its maximum regardless of the status of the Touch Saver. The activation time for the Touch Saver is determined by means of **citSetTouchSettings()**.
The value for *blSaverActive* is set by means of **citGetDimming()**.The function **citSetDimming()** disregards this parameter, therefore it is neither able to activate nor deactivate the Touch Saver!

## DRIVERCONSTANTS

This structure contains the unchanging parameters (constants) of the CiTouchW driver.

| Type | Name | Description |
|---|---|---|
| WORD | dcSmoothMax | Upper limit for the smoothing factors. The smoothing factors have to be smaller than the value declared here. |
| WORD | dcFixedBias | Scaling factor of the fixed-point format for the calibration of absolute coordinates. |
| WORD | dcReportMax | Maximum size of buffer for the receive of IRT reports. |
| WORD | dcKeyNum | The IRT is operated in the key mode in which one single key is defined by means of the number declared here. |
| WORD | dcKeyMode | Operating mode of the key in the Mode-C protocol. |

## DRIVERSETTINGS

This structure contains the variable parameters of the CiTouchW driver.

| Type | Name | Description |
|---|---|---|
| WORD | dsSmoothX | Smoothing factor for the X-axis |
| WORD | dsSmoothY | Smoothing factor for the Y-axis |
| short | dsOfsX | X-distance between touch spot and cursor position |
| short | dsOfsY | Y-distance between touch spot and cursor position |
| WORD | dsTapTime | Time span in which the IRT has to be interrupted a second time in order to create a Tap. The Tap Time is declared in steps of 55 ms. |
| WORD | dsCoordSkip | Number of coordinates messages to be disregarded before a new cursor position is reported. |
| WORD | dsDberrSkip | Number of dual touching messages to be disregarded before a dual touching is actually detected. |
| WORD | dsButtonNum | Number of mouse button to be emulated.<br>1 = left-hand side mouse button<br>2 = right-hand side mouse button<br>3 = both mouse buttons simultaneously |
| BOOL | dsAutoInit | TRUE if after an interruption of the connection between the IRT and the computer the linking is to be automatically restored. |
| BOOL | dsButtonClick | TRUE if an acoustic signal is to occur as response to a mouse click |
| BOOL | dsAbsolute | TRUE if absolute coordinates are to be used. |
| BOOL | dsSmoothAlways | TRUE if the coordinates smoothing should also cover the release and once more interruption of the Touch Zone. |

## MOUSEINFO

This structure contains the global parameters of a Windows-compatible mouse driver.

| Type | Name | Description |
|---|---|---|
| BYTE | msExists | Unequals zero in case a mouse exists |
| BYTE | msRelative | Unequals zero in case relative coordinates are reported |
| WORD | msNumButtons | Number of mouse buttons |
| WORD | msRate | Maximum number of Interrupts per second |
| WORD | msXThresh | X-threshold for mouse acceleration |
| WORD | msYThresh | Y--threshold for mouse acceleration |
| WORD | msXRes | X-resolution of the mouse |
| WORD | msYRes | Y- resolution of the mouse |
| WORD | msMouseComPort | Serial port used by the mouse |

## SERIALHARDWARE

This structure contains the parameters of the serial interface.

| Type | Name | Description |
|---|---|---|
| WORD | shPort | Base address of the serial interface module |
| short | shInterrupt | Respective ISA Bus Interrupt channel |
| WORD | shBaudDiv | Divisor for the baud rate |

## TOUCHHARDWARE

This structure contains the unchanging parameters of the IRT.

| Type | Name | Description |
|---|---|---|
| WORD | thBeamsX | Number of physically present X-beams |
| WORD | thBeamsY | Number of physically present Y-beams |
| WORD | thResolutionX | Maximum X-coordinate provided by the IRT |
| WORD | thResolutionY | Maximum Y-coordinate provided by the IRT |
| WORD | thProtocol | Communication protocol used by the IRT. This field can take over one of the values defined by the TP_??? constants. |
| Char | thDesignator[33] | Designation of the IRT (0-terminated string) |
| char | thAssy[17] | ASSY number of the IRT (0-terminated string) |
| char | thMem | 'E' = EPROM, 'F' = FLASH Memory |
| BYTE | thReserved1 | reserved |
| char | thComment[257] | At the Mode-C protocol: optional comment (0-terminated string)<br>At the CTS1 protocol: Serial number of the IRT (0-terminated string) |

## TOUCHSETTINGS

This structure contains the variable parameters of the IRT.

| Type | Name | Description |
|---|---|---|
| WORD | tsMinBeamsX | Minimum number of interrupted X-beams |
| WORD | tsMinBeamsY | Minimum number of interrupted Y-beams |
| WORD | tsBeamTimeout | Blank-out time for defective beams in steps of 1 s. |
| WORD | tsTCont | Time interval between two coordinates messages in steps of 1 ms. |
| WORD | tsPressLevel | Pressure sensitivity |
| WORD | tsTSaver | Time span to elapse before the Touch Saver is activated in steps of 1 s. |
| WORD | tsTScan | Time interval between two scan operations at activated Touch Saver in steps of 1 ms. |

### *5.4 Notification by the CiTouchW*

If upon the call of **citOpen()** as the first parameter a valid window handle was passed, CiTouchW transmits a notification to this window as soon as a complete report was received from the IRT. The number of this notification is passed to **citOpen()** as the second parameter (e.g. WM_USER+1). The notification parameters are set to the following values:

| | |
|---|---|
| **wParam** | Status of receive buffer (like **citReceive()**) |
| **LOWORD(lParam)** | Number of bytes in the receive buffer |
| **HIWORD(lParam)** | 0 |

# 6 Appendix

All parameters for the programs CITOUCHW.DRV and CTAPPLET.CPL are saved in the files CTW300.INI and CTA300.INI in the Windows directory. In order to reset all settings to their default values, the respective *.INI file can be deleted. The programs then operate with their internally saved default values and write possible changes to their *.INI file only if required.
The structure of the files CTW300.INI and CTA300.INI is identical with the one of any other *.INI file under Windows.
Possible values for the various entries are indicated in italics. The limits of the value ranges are displayed as in the following example:

X_Mul = *-32768..+32767*

Entries with a toggle function are indicated with "Yes | No". "Yes" represents the value 1, "No" accordingly the value 0.
If there is a selection possible between several values, these values are separated by a vertical line, for example:

AutoReinit = *Yes* | *No*

## *6.1 CTW300.INI*

The CiTouchW driver saves its parameters to the file CTW300.INI. In the following the various sections and entries of the file CTW300.INI are listed in alphabetical order. If an entry is omitted, the default value that is internally saved in the driver is used.
In this context, the entries "IO_Base" and "Interrupt" in the section "[Hardware]" represent a particularity. If one of those entries is omitted, the CiTouchW driver will not carry out an attempt to link an IRT.

Valid sections of the file CTW300.INI:

| Section | Function |
|---|---|
| **[Acceleration]** | Parameters for the cursor acceleration |
| **[Calibration]** | Calibration of absolute and relative coordinates |
| **[Commands]** | Mouse button emulation |
| **[Hardware]** | Parameters of the serial interface |
| **[Settings]** | General settings for the driver and the IRT |
| **[Sound]** | Settings for the mouse click creation |

### 6.1.1 [Acceleration]

This section contains the parameters for the cursor acceleration when absolute coordinates are used.
In the section **[Acceleration]** the following entries are possible:

X_Border = *0..32767*

This entry determines the width of the area in which the cursor is accelerated in reference to the finger movement. The default value is **7864**, which equals 24%.

Y_Border = *0..32767*

This entry determines the height of the area in which the cursor is accelerated in reference to the finger movement. The default value is **7864**, which equals 24%.

X_Mul = *0..(32767*dcFixedBias)/caMulX*

This entry determines the factor with which the cursor is accelerated in comparison to the finger movement in X-direction. The default value **384**, which equals a factor of 1,5.

Y_Mul = *0..(32767*dcFixedBias)/caMulY*

This entry determines the factor with which the cursor is accelerated in comparison to the finger movement in Y-direction. The default value **384**, which equals a factor of 1,5.

### 6.1.2 [Calibration]

This section contains the calibration parameters for absolute and relative coordinates.
In the section **[Calibrations]** the following entries are possible:

X_Add = *-32767..32767*

This entry determines the offset of the straight line equation for the calibration of absolute X-coordinates. The default value is **0**.

X_Mul = *0..65535*

This entry determines the slope of the straight line equation for the calibration of absolute X-coordinates. X_Mul represents an unsigned fixed-point number. The scaling factor can be requested by means of the function **citGetDriverConstants()**. The default value is **256**.

XRel_Div = *0..65535*

This entry determines the divisor for the scaling of relative X-coordinates.
The default value is **1**.

XRel_Mul = *-32768..+32767*

This entry determines the multiplier for the scaling of relative X-coordinates.
The default value is **-1**.

Y_Add = *-32767..32767*

This entry determines the offset of the straight line equation for the calibration of absolute Y-coordinates. The default value is **0**.

Y_Mul = *0..65535*

This entry determines the slope of the straight line equation for the calibration of absolute Y-coordinates. Y_Mul represents an unsigned fixed-point number. The scaling factor can be requested by means of the function citGetDriverConstants **()**. The default value is **256**.

YRel_Div = *0..65535*

This entry determines the divisor for the scaling of relative Y-coordinates.
The default value is **1**.

YRel_Mul = *-32768..+32767*

This entry determines the multiplier for the scaling of relative Y-coordinates.
The default value is **-1**.

### 6.1.3 [Commands]

This section contains the parameters for the mouse button emulation. For a description of the mouse button emulation refer to chapter "User-defined key emulation" at page 16.
The values for the conditions of the status transitions have the following meaning:

| Value | Meaning |
|---|---|
| **0** | Never |
| **1** | Immediately |
| **2** | Enter |
| **3** | Z-Press |
| **4** | Dual Touch |
| **5** | Tap |
| **6** | Leave |
| **7** | Z-Release |
| **8** | No Dual Touch |

In the section **[Commands]** the following entries are possible:

Idle_T1 = *0..8*

This entry determines the condition for the transition from status "IDLE" to the status "T1". The default value is **2**.

ModeChange = *0* | *3* | *4* | *5*

Condition for the change between absolute and relative coordinates. The default value is **0**.

T1_Trigger = *0..8*

This entry determines the condition for the transition from status "T1" to the status "TRIGGER". The default value is **4**.

T2_UPT3 = *0..8*

This entry determines the condition for the transition from status "T2" to the status "T3". The default value is **0**.

Time1 = *0..65535*

This entry determines the time constant for the status "T1". The value entered here corresponds to the desired time constant in milliseconds. The default value is **0**.

Time2 = *0..65535*

This entry determines the time constant for the status "T2". The value entered here corresponds to the desired time constant in milliseconds. The default value is **0**.

Time3 = *0..65535*

This entry determines the time constant for the status "T3". The value entered here corresponds to the desired time constant in milliseconds. The default value is **0**.

Trigger_T2 = *0..8*

This entry determines the condition for the transition from status "TRIGGER" into the status "T2". The default value is **6**.

UPT3_Idle = *0..8*

This entry determines the condition for the transition from status "T3" into the status "IDLE". The default value is **0**.

UPT3_Trigger = *0..8*

This entry determines the condition for the transition from status "T3" into the status "TRIGGER". The default value is **0**.

### 6.1.4 [Hardware]

This section contains the parameters of the serial interface. If one of the entries "IO_Base" or "Interrupt" is either omitted or set to 0, there will be no attempt to link the IRT. The driver remains in the memory, though. This way, the API functions described in chapter "CiTouchW-API" at page 27 can be used.
In the section **[Hardware]** the following entries are possible:

BaudDivisor = *0..65535*

This entry determines the divisor for the baud rate generator. The following formula is used to calculate the baud rate:

$$BaudRate = \frac{f_{Quarz}}{16 \cdot BaudDivisor};$$ $f_{Quarz}$ = 1.8432 MHz at a standard PC

The default value for BaudDivisor is **12**, corresponding to **9600 baud**.

Interrupt = *0..15*

This entry determines the number of the ISA Bus Interrupt channel of the serial interface. The default value is **0**.

IO_Base = *0..65535*

This entry determines the base address of the serial interface module. The default value is **0**.

IRT_Mode = *0|1|2*

This entry contains the most recently detected communication protocol of the IRT. This way the IRT can be linked faster. In case this entry is either omitted or faulty, the CiTouchW driver attempts to detect the current communication protocol automatically. The following assignment of values to detected communication protocols is used:

| Value | Meaning |
|---|---|
| 0 | No protocol detected |
| 1 | Mode-C protocol |
| 2 | CTS1 protocol |

### 6.1.5 [Settings]

This section contains all variable operational parameters of both the driver and the IRT.
In the section **[Settings]** the following entries are possible:

AbsoluteMouse = *Yes* | *No*

This entry determines whether after an initialization of the driver absolute or relative coordinates are used. If the entry "ModeChange" in the section **[Commands]** contains any other value than 0, the coordinates mode can be dynamically changed during regular operation. The default value is **Yes** (1).

AutoReinit = *Yes* | *No*

This entry determines whether after an interruption of the connection between the IRT and the computer the linking should take place automatically. The default value is **Yes** (1).

BeamTimeout = *0..65535*

This entry determines the blank-out time for defective beams. If an interruption of any IRT's beam lasts longer than the time span determined here in seconds, it is excluded from the coordinates calculation. The value 0 prevents the blanking-out of beams. The default value is **20**.

Button = *1..3*

This entry determines the mouse button to be emulated. 1 represents the left-hand side mouse button, 2 the right-hand side mouse button and 3 both mouse buttons at the same time. The default value is **1**.

ButtonClick = *Yes* | *No*

This entry determines whether an acoustic signal at the PC speaker is to occur as a response to a mouse click. The default value is **Yes** (1).

ContTime = *0..65535*

This entry determines the time interval between two coordinates messages of the IRT. The value entered here corresponds to a time interval in milliseconds. In this context the time required for the transmission of a coordinates message has to be regarded (depends on the baud rate).
The default value is **22**.

CoordEnterZ = *Yes* | *No*

This entry determines whether after the interruption of the Touch Zone the first reported cursor position additionally requires the pressure exerted onto the screen to exceed its limit value. However, for that the IRT needs to be equipped with a Z-axis. The default value is **No** (0).

CoordinateSkip = *0..65535*

This entry determines the number of coordinates messages of the IRT to be disregarded after the interruption of the Touch Zone. The default value is **1**.

CoordSignalZ = *Yes* | *No*

This entry determines whether after the interruption of the Touch Zone further cursor positions additionally require the pressure exerted onto the screen to exceed its limit value. However, for that the IRT needs to be equipped with a Z-axis. The default value is **No** (0).

DblErrSkip = *0..65535*

This entry determines the number of dual touching messages of the IRT to be disregarded before a dual touching is actually detected. The default value is **1**.

DimmingHigh = 0..255

This entry determines the mark-to-space ratio of the IRT's PWM output at deactivated Touch Saver. The default value is **0**.

DimmingLow = 0..255

This entry determines the mark-to-space ratio of the IRT's PWM output at activated Touch Saver. The default value is **0**.

MinXBeams = *1..5*

This entry determines the number of X-beams that have to be interrupted simultaneously in order to be detected as a valid interruption. The default value is **1**.

MinYBeams = *1..5*

This entry determines the number of Y-beams that have to be interrupted simultaneously in order to be detected as a valid interruption. The default value is **1**.

Pressure = *0..255*

This entry determines the required pressure to be exerted onto the front screen in order to release the pressure-controlled events of the driver. The value 0 deactivates the Z-axis of the IRT. The default value is **20**.

SaverScan = *1..65535*

This entry determines the scan rate of the IRT at activated Touch Saver. The actual scan rate corresponds to the value entered here in milliseconds. The default value is **500**.

SaverTime = *0..65535*

This entry determines the time span to elapse before the Touch Saver is activated. The activation time corresponds to the value entered here in seconds. At a value of 0 the Touch Saver is immediately activated. At a value of 65535 the Touch Saver is never activated. The default value is **65535**.

SmoothAlways = *Yes* | *No*

This entry determines whether the calculation of the average value of absolute coordinates should also cover the release and once more interruption of the Touch Zone. The default value is **No** (0).

TapTime = *0..65535*

This entry determines the time interval in which the Touch Zone has to be interrupted a second time in order to create a Tap. The time interval corresponds to the value entered here in milliseconds. The default value is **220**.

X_Offset = *-32768..+32767*

This entry determines the X-distance between the touch spot and the current cursor position. Positive values shift the cursor position towards the right-hand side, negative values towards the left-hand side. The default value is **0**.

X_Smoothing = *0..dcSmoothMax-1*

This entry determines the number of coordinates messages of which an average value is calculated for the smoothing of absolute X-coordinates. The upper limit can be requested by means of the function **citGetDriverConstants()**. The default value is **10**.

Y_Offset = *-32768..+32767*

This entry determines the Y-distance between the touch spot and the current cursor position. Positive values shift the cursor position downwards, negative values upwards. The default value is **0**.

Y_Smoothing = *0..dcSmoothMax-1*

This entry determines the number of coordinates messages of which an average value is calculated for the smoothing of absolute Y-coordinates. The upper limit can be requested by means of the function **citGetDriverConstants()**. The default value is **10**.

### 6.1.6 [Sound]

This section contains the parameters for the creation of the mouse click.
In the section **[Sound]** the following entries are possible:

ButtonClick = *0* | *1* | *2* | *3*

This entry determines the conditions for a beep to be emitted. To do so, one or a combination of the following values can be used:

| Value | Meaning |
|---|---|
| 0 | No beep emitted |
| 1 | Beep emitted when mouse button is pressed |
| 2 | Beep emitted when mouse button is released |
| 3 | Beep emitted when mouse button is either pressed or released. |

The default value is **1**.

FreqDown = *0..65535*

This entry determines the pitch of the beep to be emitted when the mouse button is pressed in steps of 1 Hz. The default value is **783**.

FreqUp = *0..65535*

This entry determines the pitch of the beep to be emitted when the mouse button is released in steps of 1 Hz. The default value is **1046**.

TimeDown = *0..65535*

This entry determines the duration of the beep to be emitted when the mouse button is pressed in steps of 1 millisecond. The default value is **30**.

TimeUp = *0..65535*

This entry determines the duration of the beep to be emitted when the mouse button is released in steps of 1 millisecond. The default value is **30**.

## *6.2 CTA300.INI*

The System Control application saves its parameters in the file CTA300.INI. In the following the various sections and entries of the file CTA300.INI are listed in alphabetical order. If an entry is omitted, the default value that is saved in CTAPPLET.CPL is used.

Valid sections of the file CTA300.INI:

| Section | Function |
|---|---|
| **[Advanced]** | Settings of the dialogue page "Advanced Settings" |
| **[ButtonClick.Double]** | Parameters for the mouse button emulation "Dual Touch" |
| **[ButtonClick.DoubleExit]** | Parameters for the mouse button emulation "Double / Exit" |
| **[ButtonClick.Enter]** | Parameters for the mouse button emulation "Enter" |
| **[ButtonClick.Exit]** | Parameters for the mouse button emulation "Exit" |
| **[ButtonClick.Tap]** | Parameters for the mouse button emulation "Tap" |
| **[ButtonClick.Time]** | Parameters for the mouse button emulation "Time" |
| **[ButtonClick.TimeTime]** | Parameters for the mouse button emulation "Time / Time" |
| **[ButtonClick.User1]** | Parameters for the mouse button emulation "User 1" |
| **[ButtonClick.User2]** | Parameters for the mouse button emulation "User 2" |
| **[ButtonClick.User3]** | Parameters for the mouse button emulation "User 3" |
| **[ButtonClick.ZPress]** | Parameters for the mouse button emulation "Z-Press" |

### 6.2.1 [Advanced]

This section contains those parameters of the dialogue page "Advanced Settings", that are not already saved in the file CTW300.INI.
In the section **[Advanced]** the following entries are possible:

SioQuartz.High = *0..65535*

This entry holds the High-Word of the frequency (/100) for the generation of the serial port's baud rate. The default value is **0x0000**.

SioQuartz.Low = *0..65535*

This entry holds the Low-Word of the frequency (/100) for the generation of the serial port's baud rate. The default value is **0x4800**.

### 6.2.2 [ButtonClick.???]

These sections contain the parameters of the mouse button emulation. All sections with the prefix "ButtonClick." contain at least the entries listed in the following.
For possible values for the conditions of status transitions please refer to the chapter "[Commands]" at page 46. The various conditions themselves are described in chapter "User-defined key emulation" at page 16.

IdleT1 = *0..8*

This entry determines the condition for the transition from status "IDLE" to the status "T1". The default values are:

| Section | Default value for IdleT1 |
|---|---|
| **[ButtonClick.Double]** | 2 |
| **[ButtonClick.DoubleExit]** | 2 |
| **[ButtonClick.Enter]** | 2 |
| **[ButtonClick.Exit]** | 1 |
| **[ButtonClick.Tap]** | 2 |
| **[ButtonClick.Time]** | 2 |
| **[ButtonClick.TimeTime]** | 2 |
| **[ButtonClick.User1]** | 0 |
| **[ButtonClick.User2]** | 0 |
| **[ButtonClick.User3]** | 0 |
| **[ButtonClick.ZPress]** | 3 |

T1 = *0..65535*

This entry determines the time constant for the status "T1". The time constant corresponds to the value entered here in milliseconds. The default values are:

| Section | Default value for T1 |
|---|---|
| **[ButtonClick.Double]** | 0 |
| **[ButtonClick.DoubleExit]** | 0 |
| **[ButtonClick.Enter]** | 0 |
| **[ButtonClick.Exit]** | 0 |
| **[ButtonClick.Tap]** | 0 |
| **[ButtonClick.Time]** | 400 |
| **[ButtonClick.TimeTime]** | 990 |
| **[ButtonClick.User1]** | 0 |
| **[ButtonClick.User2]** | 0 |
| **[ButtonClick.User3]** | 0 |
| **[ButtonClick.ZPress]** | 0 |

T1Trigger = *0..8*

This entry determines the condition for the transition from status "T1" to the status "TRIGGER". The default values are:

| Section | Default value for T1Trigger |
|---|---|
| **[ButtonClick.Double]** | 4 |
| **[ButtonClick.DoubleExit]** | 4 |
| **[ButtonClick.Enter]** | 1 |
| **[ButtonClick.Exit]** | 6 |
| **[ButtonClick.Tap]** | 5 |
| **[ButtonClick.Time]** | 2 |
| **[ButtonClick.TimeTime]** | 2 |
| **[ButtonClick.User1]** | 0 |
| **[ButtonClick.User2]** | 0 |
| **[ButtonClick.User3]** | 0 |
| **[ButtonClick.ZPress]** | 1 |

T2 = *0..65535*

This entry determines the time constant for the status "T2". The time constant corresponds to the value entered here in milliseconds. The default values are:

| Section | Default value for T2 |
|---|---|
| **[ButtonClick.Double]** | 0 |
| **[ButtonClick.DoubleExit]** | 0 |
| **[ButtonClick.Enter]** | 0 |
| **[ButtonClick.Exit]** | 0 |
| **[ButtonClick.Tap]** | 0 |
| **[ButtonClick.Time]** | 0 |
| **[ButtonClick.TimeTime]** | 660 |
| **[ButtonClick.User1]** | 0 |
| **[ButtonClick.User2]** | 0 |
| **[ButtonClick.User3]** | 0 |
| **[ButtonClick.ZPress]** | 0 |

T2UpT3 = *0..8*

This entry determines the condition for the transition from status "T2" to the status "T3". The default values are:

| Section | Default value for T2UpT3 |
|---|---|
| **[ButtonClick.Double]** | 0 |
| **[ButtonClick.DoubleExit]** | 0 |
| **[ButtonClick.Enter]** | 0 |
| **[ButtonClick.Exit]** | 0 |
| **[ButtonClick.Tap]** | 0 |
| **[ButtonClick.Time]** | 0 |
| **[ButtonClick.TimeTime]** | 6 |
| **[ButtonClick.User1]** | 0 |
| **[ButtonClick.User2]** | 0 |
| **[ButtonClick.User3]** | 0 |
| **[ButtonClick.ZPress]** | 0 |

T3 = *0..65535*

This entry determines the time constant for the status "T3". The time constant corresponds to the value entered here in milliseconds. The default values are:

| Section | Default value for T3 |
|---|---|
| **[ButtonClick.Double]** | 0 |
| **[ButtonClick.DoubleExit]** | 0 |
| **[ButtonClick.Enter]** | 0 |
| **[ButtonClick.Exit]** | 0 |
| **[ButtonClick.Tap]** | 0 |
| **[ButtonClick.Time]** | 600 |
| **[ButtonClick.TimeTime]** | 0 |
| **[ButtonClick.User1]** | 0 |
| **[ButtonClick.User2]** | 0 |
| **[ButtonClick.User3]** | 0 |
| **[ButtonClick.ZPress]** | 0 |

TriggerT2 = *0..8*

This entry determines the condition for the transition from status "TRIGGER" to the status "T2". The default values are:

| Section | Default value for TriggerT2 |
|---|---|
| **[ButtonClick.Double]** | 8 |
| **[ButtonClick.DoubleExit]** | 6 |
| **[ButtonClick.Enter]** | 6 |
| **[ButtonClick.Exit]** | 1 |
| **[ButtonClick.Tap]** | 6 |
| **[ButtonClick.Time]** | 6 |
| **[ButtonClick.TimeTime]** | 1 |
| **[ButtonClick.User1]** | 0 |
| **[ButtonClick.User2]** | 0 |
| **[ButtonClick.User3]** | 0 |
| **[ButtonClick.ZPress]** | 7 |

UpT3Idle = *0..8*

This entry determines the condition for the transition from status "T3" to the status "IDLE". The default values are:

| Section | Default value for UpT3Idle |
|---|---|
| **[ButtonClick.Double]** | 0 |
| **[ButtonClick.DoubleExit]** | 0 |
| **[ButtonClick.Enter]** | 0 |
| **[ButtonClick.Exit]** | 0 |
| **[ButtonClick.Tap]** | 0 |
| **[ButtonClick.Time]** | 0 |
| **[ButtonClick.TimeTime]** | 6 |
| **[ButtonClick.User1]** | 0 |
| **[ButtonClick.User2]** | 0 |
| **[ButtonClick.User3]** | 0 |
| **[ButtonClick.ZPress]** | 0 |

UpT3Trigger = *0..8*

This entry determines the condition for the transition from status "UpT3" to the status "TRIGGER". The default values are:

| Section | Default value for UpT3Trigger |
|---|---|
| **[ButtonClick.Double]** | 0 |
| **[ButtonClick.DoubleExit]** | 0 |
| **[ButtonClick.Enter]** | 0 |
| **[ButtonClick.Exit]** | 0 |
| **[ButtonClick.Tap]** | 0 |
| **[ButtonClick.Time]** | 2 |
| **[ButtonClick.TimeTime]** | 2 |
| **[ButtonClick.User1]** | 0 |
| **[ButtonClick.User2]** | 0 |
| **[ButtonClick.User3]** | 0 |
| **[ButtonClick.ZPress]** | 0 |

The sections **[ButtonClick.User1]** up to **[ButtonClick.User3]** additionally contain the following entries:

DblErrSkip = *0..65535*

With regard to the respective user-defined Mouse Button Emulation Mode, this entry holds the number of dual touching messages to be disregarded before a dual touching is accepted. The number entered at "DblErrSkip" in the section **[Settings]** of the file CTW300.INI represents the default value.

Name = *String*

This entry determines the name of the respective user-defined Mouse Button Emulation Mode. Default is an empty string.

TapTime = *0..65535*

With regard to the respective user-defined Mouse Button Emulation Mode, this entry determines the maximum time span that may elapse between leaving and interrupting the Touch Zone in order to create a Tap. The time span corresponds to the value entered here multiplied by 55 milliseconds. The number entered at "TapTime" in the section **[Settings]** of the file CTW300.INI represents the default value.

ZPress = *0..255*

With regard to the respective user-defined Mouse Button Emulation Mode, this entry determines the required pressure to be exerted onto the front screen in order to cause a pressure event. The number entered at "Pressure" in the section **[Settings]** of the file CTW300.INI represents the default value.

# 7 Index