CITRON
Creative Industrial Electronic Systems

CiTouchD

Touch Driver for DOS

V1.0.01x

User's Manual

Document revision

Rev.	Description	Reviser.	Date
001	First issue	wh, pk	27. Nov. 1996
002	Adapted to CiTouchD 1.0.008	wh	28. Jan. 1997
003	Adapted to CiTouchD 1.0.009	wh	07. Feb. 1997
004	Adapted to CiTouchD 1.0.010; changed some default values in CITOUCHD.INI	wh	14. Jul. 1997
005	Adapted to new firm address	wh	29. Aug. 1997
006	New logo added	pk	25. 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: \\ntserv1\dokument\manuals\citouchd\ctd_e_r005.doc
Date: 29.08.97 11:40
Document revision: 3
Document reference: h:\dformat.dot\cidoku.dot

1 Table of contents

2 Introduction

The CiTouchD mouse driver (Citron Touch driver for MS-DOS) enables the Citron Infrared Touch (in the following referred to as IRT) to be used as a mouse substitute with the operating system MS-DOS. The requirements to be met for the operation of the CiTouchD are a MS-DOS version 3.0 or higher and at least an i386 microprocessor.
The mouse driver implemented in the CiTouchD provides all functions of a standard MS-DOS mouse driver that are required for both the detection of the cursor position (position of the Touch interruption) and the simulation of a mouse button. Any previously installed driver for a standard mouse is disabled by the CiTouchD driver. Therefore this mouse cannot be used as an input medium anymore.
Before starting the installation, the IRT needs to be connected to a serial interface of the computer. Please note that in order to operate the CiTouchD driver, the serial interface has to be capable of interpreting Interrupts.
If a Citron-LDVGA graphics card is used in conjunction with the LDRI reception 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 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.
Contrary to a regular mouse the CiTouchD 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 CiTouchD the emulation of the mouse buttons is widely user-configurable and programmable.
All parameters that influence the operating methods of the CiTouchD, are saved in an INI-file in ASCII-format. Therefore, all settings can be changed either during the installation or during regular operation.

3 Loading CiTouchD

The CiTouchD driver is loaded by executing the file CITOUCHD.EXE either from the MS-DOS prompt or from the AUTOEXEC.BAT and stays resident in the computer’s memory.
Before starting CiTouchD, the IRT needs to be connected to a serial interface of the computer. Please note that in order to operate the CiTouchD driver, the serial interface has to be capable of interpreting Interrupts.
If a Citron-LDVGA graphics card 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 LDVGA 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.

4 Configuration

The parameters to configure the behaviour of the CiTouchD can be set in three different ways:

1. Passed on as a prompt parameter
2. As entries in the file CITOUCHD.INI
3. Calls from the programming interface (Int 33h or Int 2Fh API)

The CiTouchD uses the parameters in the following order:

First the driver reads its parameters from the file CITOUCHD.INI. In case of a specific entry missing, its default value is used. If the program was started with additional prompt parameters, these parameters overrule the ones from the file CITOUCHD.INI. The prompt parameters will then update the entries in the CITOUCHD.INI. The possible entries of the CITOUCHD.INI are described in chapter "

CITOUCHD.INI“ on page 49.

Calls from the programming interface only change the internal variables of the driver. The contents of the file CITOUCHD.INI is not changed that way.

4.1 Prompt parameters

CiTouchD is able to interpret the following prompt parameters in any order. It distinguishes between small and capital letters:

Parameters	Range	CITOUCH.INI	Default	Description
/? [/h] [/H]	-	-	-	Lists all permitted prompt parameters
?	-	-	-	Installation status of CiTouchD
?D	-	-	-	Requests current settings of the resident share of CiTouchD
?T	-	-	-	Hard- and software revision of connected IRT
/b:	1..3	[Settings] Button =	1	Determines the mouse button to be emulated: 1 = left-hand side mouse button 2 = right-hand side mouse button 3 = both mouse buttons at the same time
/c:	1..4	[Hardware] Interrupt = IO_Base	1	Determines the number of the serial interface to be used by the IRT. This option combines /i: and /p:, assuming the standard PC serial interfaces: COM1 (/c:1) -> Port = 3F8h, IRQ 4 COM2 (/c:2) -> Port = 2F8h, IRQ 3 COM3 (/c:3) -> Port = 3E8h, IRQ 4 COM4 (/c:4) -> Port = 2E8h, IRQ 3
/d	-	-	-	Removes the resident part of the CiTouchD driver from the memory
/k:	0..64	[Citouchd] Stack =	6	Determines size of stack set up by CiTouchD in steps of 256 byte. In case of /k:0, CiTouchD uses the stack of the currently running program.
/i:	0..15	[Hardware] Interrupt =	4	Determines number IRQ to be used by serial interface
/m:	0..7	[Commands] for possible entries refer to next chapter	3D-Touch: 7 2D-Touch: 5	Mouse Button Emulation Modes; depending on the mode additional parameters are required. These are separated by commas without any blanks.
/p:	1..FFFF	[Hardware] IO_Base =	3F8	Determines base address of serial interface. This parameter is interpreted as hexadecimal number. There are no preceding zeros required.
/r:	0..2	[Citouchd] HardReset =	0	Determines whether Touch is reset by the Mouse-API-Functions 00h and 2Fh (reset lasts up to 10 seconds). 0 = No reset 1 = reset in function 2Fh only 2 = reset in function 00h only

/t:	0,1,2, 128	[Hardware] IRT_Mode =	0	Determines communication protocol of connected IRT: 0 = automatic detection 1 = Mode-C (2D-Touch) 2 = CTS1 (3D-Touch) If the correct communication protocol is determined by this parameter, the linking of the IRT takes place faster than with the automatic detection.

The columns of the table above have the following meaning:

Parameters	Syntax of the prompt parameter
Range	Permitted value range
CITOUCHD.INI	[Section] and respective entry in the file CITOUCHD.INI
Default	Value that is used if both the prompt parameter and the entry in the file CITOUCHD.INI are missing
Description	Brief description of the function of the prompt parameter.

4.2 Pre-defined Mouse Button Emulation Modes

There is a total of 8 pre-defined Mouse Button Emulation Modes. One of them is selected by means of the prompt parameter /m: at the start of CiTouchD. If there is no Mouse Button Emulation Mode explicitly selected at the start of CiTouchD, the Emulation Mode determined in the section [Commands] of the file CITOUCHD.INI is used.
In the following the pre-defined modes are described.

/m:0 Enter

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

Parameter: none

Example:
```
c:\>citouchd.exe /m:0
```

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

CITOUCHD.INI:
```
[Commands]
Time1          = 0
Time2          = 0
Time3          = 0
Idle_T1        = 2
T1_Trigger     = 1
Trigger_T2     = 6
T2_UPT3        = 0
UPT3_Idle      = 0
UPT3_Trigger   = 0
```

/m:1 Exit

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

Parameter: none

Example:
```
c:\>citouchd.exe /m:1
```

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

CITOUCHD.INI:
```
[Commands]
    Time1          = 0
    Time2          = 0
    Time3          = 0
    Idle_T1        = 1
    T1_Trigger     = 6
    Trigger_T2     = 1
    T2_UPT3        = 0
    UPT3_Idle      = 0
    UPT3_Trigger   = 0
```

/m:2,t 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.

Parameter: **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.

Example:
```
c:\>citouchd.exe /m:2,300
```

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

CITOUCHD.INI:
```
[Commands]
    Time1          = 0
    Time2          = 0
    Time3          = 0
    Idle_T1        = 2
    T1_Trigger     = 5
    Trigger_T2     = 6
    T2_UPT3        = 0
    UPT3_Idle      = 0
    UPT3_Trigger   = 0
[Settings]
    TapTime        = t
```

/m:3,t1,t2 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.

Parameter

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.

Example

```
c:\>citouchd.exe /m:3,500,300
```

Advantages

- Easy handling
- Cursor motion at pressed mouse button possible

Disadvantages

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

CITOUCHD.INI

```
[Commands]
    Time1          = t1
    Time2          = 0
    Time3          = t2
    Idle_T1        = 2
    T1_Trigger     = 2
    Trigger_T2     = 6
    T2_UPT3        = 0
    UPT3_Idle      = 0
    UPT3_Trigger   = 2
```

/m:4,n 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.

Parameter

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.

Example

```
c:\>citouchd.exe /m:4,1
```

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

CITOUCHD.INI

```
[Commands]
    Time1          = 0
    Time2          = 0
    Time3          = 0
    Idle_T1        = 2
    T1_Trigger     = 4
    Trigger_T2     = 8
    T2_UPT3        = 0
    UPT3_Idle      = 0
    UPT3_Trigger   = 0
[Settings]
    DblErrSkip     = n
```

m:5,n 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.

Parameter

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.

Example

```
c:\>citouchd.exe /m:5,1
```

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

CITOUCHD.INI

```
[Commands]
        Time1           = 0
        Time2           = 0
        Time3           = 0
        Idle_T1         = 2
        T1_Trigger      = 4
        Trigger_T2      = 6
        T2_UPT3         = 0
        UPT3_Idle       = 0
        UPT3_Trigger    = 0
[Settings]
        DblErrSkip      = n
```

m:6,t1,t2 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.

Parameter

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.

Example

```
c:\>citouchd.exe /m:6,500,300
```

Advantages

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

Disadvantages

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

CITOUCHD.INI

```
[Commands]
        Time1           = t1
        Time2           = t2
        Time3           = 0
        Idle_T1         = 2
        T1_Trigger      = 2
        Trigger_T2      = 1
        T2_UPT3         = 6
        UPT3_Idle       = 6
        UPT3_Trigger    = 2
```

m:7,p 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.

Parameter

Pressure Sensitivity (p): 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.

Example

```
c:\>citouchd.exe /m:7,10
```

Advantages

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

Disadvantages

- IRT needs to be equipped with pressure sensors

CITOUCHD.INI

```
[Commands]
        Time1           = 0
        Time2           = 0
        Time3           = 0
        Idle_T1         = 3
        T1_Trigger      = 1
        Trigger_T2      = 7
        T2_UPT3         = 0
        UPT3_Idle       = 0
        UPT3_Trigger    = 0
[Settings]
        Pressure        = p
```

5 Software interface

This chapter comprises descriptions of both the functions of the CiTouchD driver and the software interfaces provided for application programs.

5.1 Basics

In order to utilize the API functions of the CiTouchD driver, it is essential to clearly understand the function of the mouse driver emulation. The following chapters are supposed to contribute to this understanding.

5.1.1 Coordinates systems

Within the Touch driver positions always refer to a virtual graphics screen. The screen's resolution depends on the respective video mode as well as the corresponding graphics card that mode is based on. Since in the various text modes this virtual graphics screen is also used for determining the Touch interruptions and represents the basis for the communication with the Touch interface, the graphic coordinates need to be converted into column and line coordinates for the cursor. Since each column and line equal 8 dots, the graphic coordinates need to be divided by 8.
The origin of the graphics screen is situated in the left-hand side top corner. Since coordinates counted towards the right-hand side and down are positive coordinates, the column and line coordinates of the cursor will always be positive.
The functions 0Fh and 0Bh of the Int33-API supply relative coordinates. Due to reasons of compatibility to the mouse, the unit „mickeys“ (= 1/400 inch) was retained. Function 0Fh allows the ratio between mickeys and 8 virtual pixels to be set.
Relative coordinates are signed. Negative vectors represent a movement towards the left-hand side and up, positive vectors represent a movement towards the right-hand side and down.

5.1.2 Supported video modes

All video modes that are set by the function 00h of the EGA / VGA-BIOS are supported:

Screen mode	Text/Graphics	Resolution	Virtual screen
00h	Text	40*25 characters	320*200 dots
01h	Text	40*25 characters	320*200 dots
02h	Text	80*25 characters	640*200 dots
03h	Text	80*25 characters	640*200 dots
04h	Graphics	320*200 dots	320*200 dots
05h	Graphics	320*200 dots	320*200 dots
06h	Graphics	640*200 dots	640*200 dots
07h	Text	80*25 characters	640*200 dots
0Dh	Graphics	320*200 dots	320*200 dots
0Eh	Graphics	640*200 dots	640*200 dots
0Fh	Graphics	640*350 dots	640*350 dots
10h	Graphics	640*350 dots	640*350 dots
11h	Graphics	640*480 dots	640*480 dots
12h	Graphics	640*480 dots	640*480 dots
13h	Graphics	320*200 dots	320*200 dots

5.1.3 User-defined key emulation

In the CiTouchD 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. Illustration 5-1 shows the exact state-transition diagram of the Button-Machine:
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 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 condition declared in

"T1_Trigger" has to be met at the same time. Contrary to T1, for the states "T2" and "T3" to step to the following state 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.

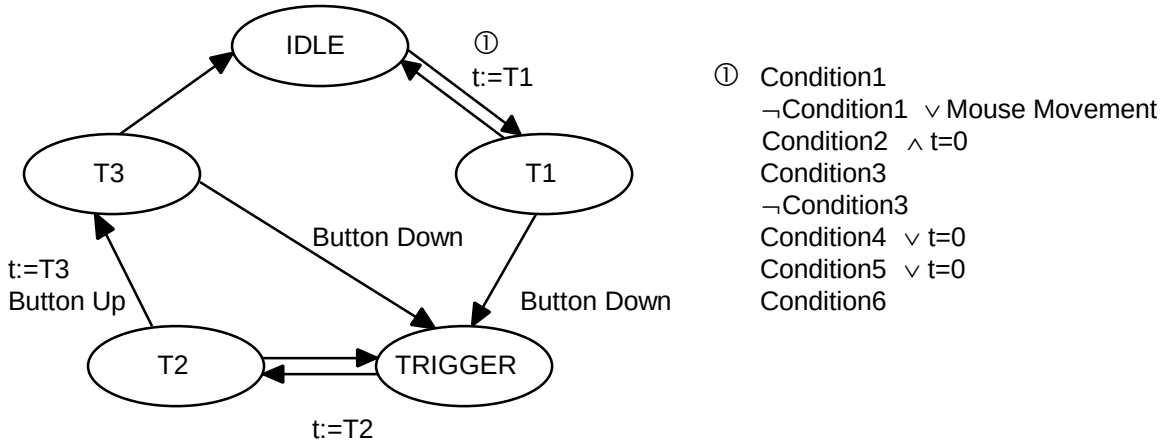


Illustration 5-1, State machine for the Mouse Button Emulation

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

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

```
T1 = 550 ms
T2 = 0 ms
T3 = 550 ms
Idle_T1         = "Enter"
T1_Trigger      = "Enter"
Trigger_T2      = "Leave"
T2_UPT3         = "Never"
UPT3_Idle       = "Never"
UPT3_Trigger    = "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
Idle_T1         = "Enter"
T1_Trigger      = "Z-Press"
Trigger_T2      = "Leave"
T2_UPT3         = "Never"
UPT3_Idle       = "Never"
UPT3_Trigger    = "Never"
```

5.1.4 Additional operating parameters

In order to ease the mouse emulation by the IRT as much as possible, there are additional parameters to adapt the behaviour of the mouse emulation to the necessities of the user and the requirements of the application program. These parameters are either preset in the file CITOUCHD.INI or temporarily changed by an application program calling the extended API functions.

5.1.4.1 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.
The API functions and the corresponding structure elements for the smoothing parameters are as follows:

citGetDriverSettings (function **0Bh**) Requesting the settings
citSetDriverSettings (function **14h**) Changing the settings

The corresponding elements of the structure DRIVERSETTINGS (refer to page 47) are:

dsSmoothX	Smoothing factor of the X-axis
dsSmoothY	Smoothing factor of the Y-axis
dsSmoothAlways	**0** = 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. **1** = 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.
dsCoordSkip	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.

The admissible values of *dsSmoothX* and *dsSmoothY* range from 0 up to *dsSmoothMax*. This value can be requested by means of **citGetDriverConstants** (function **0Eh**).

The following entries in the file CITOUCHD.INI control the smoothing behaviour:

[Settings]

X_Smoothing	corresponds to *dsSmoothX*
Y_Smoothing	corresponds to *dsSmoothY*
SmoothAlways	corresponds to *dsSmoothAlways*
CoordinateSkip	corresponds to *dsCoordSkip*

5.1.4.2 Coordinates mode

The CiTouchD 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.
The speed the cursor moves with when using relative coordinates can be set by means of the following functions:

citGetCalibrationRel (function **0Ah**) Request of calibration of the relative coordinates
citSetCalibrationRel (function **13h**) Change of calibration of the relative coordinates

Both functions take the address of a CALIBRATIONREL (refer to page 44) structure as argument for output data (**citGetCalibrationRel**) or input data (**citSetCalibrationRel**).

The coordinates mode the CiTouchD is working in is set by means of the following function:

citGetDriverSettings (function **0Bh**) Request of coordinates mode
citSetDriverSettings (function **14h**) Change of coordinates mode

The corresponding elements of the structure DRIVERSETTINGS (refer to page 47) are:
dsAbsolute **0** = The relative coordinates mode is always used
1 = The absolute coordinates mode is always used.

It is still possible during regular operation to change between the various coordinates modes. The following function determines the condition of a change to take place between the absolute and the relative coordinate's mode:

citGetCommands (function **07h**) Request of condition for mode change
citSetCommands (function **10h**) Change of condition for mode change

The corresponding elements of the structure COMMANDS (refer to page 45) are:

cmModeChange **0** = During regular operation a change between the absolute and the relative coordinates mode is possible.
3 = 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 = At each dual touching a change takes place between the absolute and the relative coordinates mode.
5 = 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 8).

In order to prevent the risk of faulty usage at critical applications, it is possible to confine the conditions for the cursor to change. The following conditions are possible:

- **Enter:** *cmCoordEnterZ* = 0
  cmCoordSignalZ = 0
  A new cursor position is reported as soon as a valid interruption of the Touch Zone occurs.

- **Z-Press/Enter:** *cmCoordEnterZ* = 1
  cmCoordSignalZ = 0
  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:** *cmCoordEnterZ* = 1
  cmCoordSignalZ = 1
  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.

This behaviour is set by means of the following functions:

citGetCoordMode (function **08h**)	Request of parameters
citSetCoordMode (function **11h**)	Change of parameters

Both functions take the address of a COORDMODE (refer to page 46) structure as argument for output data (**citGetCoordMode**) or input data (**citSetCoordMode**).

In the file CITOUCHD.INI the following entries control the coordinates mode:

[Calibration]

XRel_Div	Divisor for the X-coordinates scaling (corresponds to *crDivX)*
XRel_Mul	Multiplier for the X-coordinates scaling (corresponds to *crMulX)*
YRel_Div	Divisor for the Y-coordinates scaling (corresponds to *crDivY)*
YRel_Mul	Multiplier for the Y-coordinates scaling (corresponds to *crMulY)*
SmoothAlways	Corresponds to *dsSmoothAlways*
CoordinateSkip	Corresponds to *dsCoordSkip*

[Commands]

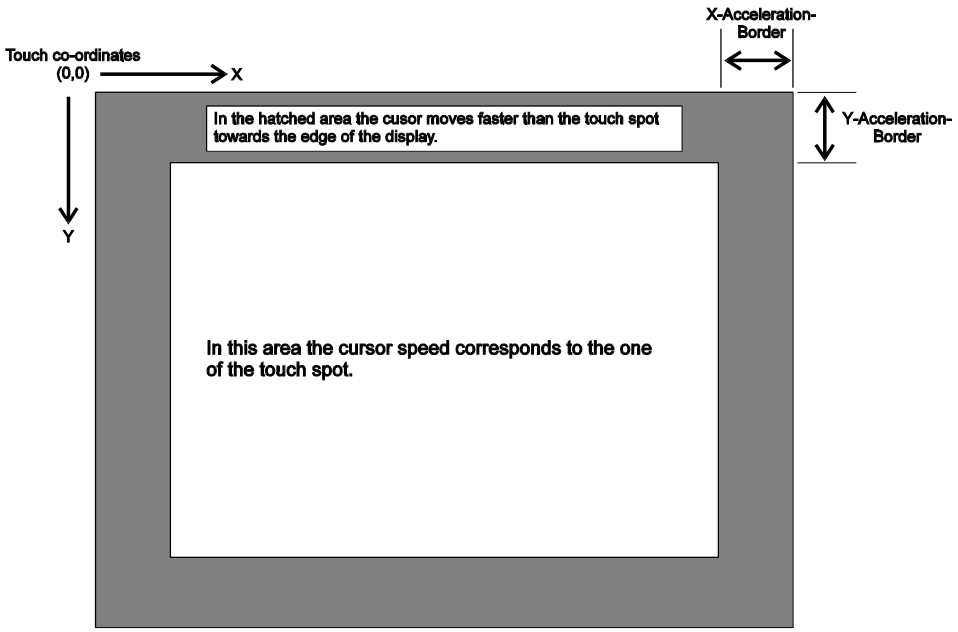
ModeChange	Corresponds to *cmModeChange*

[Settings]

AbsoluteMouse	Corresponds to *dsAbsolute*
CoordEnterZ	Corresponds to *cmCoordEnterZ*
CoordSignalZ	Corresponds to *cmCoordSignalZ*

5.1.4.3 Cursor Control

By means of these parameters a distance between the present cursor position in relation to the touch spot as well as an accelerated movement of the cursor 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 CiTouchD 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.



The following API functions are used to set the acceleration parameters:

citGetAcceleration (function **20h**)	Request of acceleration parameters
citSetAcceleration (function **21h**)	Change of acceleration parameters

Both functions take the address of a ACCELERATION (refer to page 43) structure as argument for output data (**citGetAcceleration**) or input data (**citSetAcceleration**).

The following API functions are used to set the distance between touch spot and cursor position:

citGetDriverSettings (Function **0Bh**)	Request of distance
citSetDriverSettings (Function **14h**)	Change of distance

The corresponding elements of the structure DRIVERSETTINGS (refer to page 47) are:

dsOfsX	X- distance between touch spot and cursor position
dsOfsY	Y- distance between touch spot and cursor position

In the file CITOUCHD.INI the following entries relate to the cursor control:

[Acceleration]

X_Border	Beyond this X-border the acceleration is active (corresponds to *acBorderX)*
Y_Border	Beyond this Y-border the acceleration is active (corresponds to ac*BorderY)*
X_Mul	Acceleration factor for X-direction (corresponds to *acMulX)*
Y_Mul	Acceleration factor for Y-direction (corresponds to a*cMulY)*

[Settings]

X_Offset	Corresponds to *dsOfsX*
Y_Offset	Corresponds to *dsOfsY*

5.1.4.4 Control of background illumination

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 light barriers 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 background illumination of TFT-displays.
For each of the two Touch Saver states, "active" and "inactive", the CiTouchD 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 by means of the following API functions:

citGetTouchSettings (Function **0Ch**)	Request activation time
citSetTouchSettings (Function **15h**)	Change activation time

The corresponding elements of the structure TOUCHSETTINGS (refer to page 48) are:

tsTSaver	Activation time of Touch Saver (0 = immediately active, 65535 = deactivated)
tsTScan	Time interval between two scan operations at activated TouchSaver

The following API functions are used to set the intensity of the background illumination:

citGetDimming (Function **1Dh**)	Request of illumination intensity
citSetDimming (Function **1Eh**)	Change of illumination intensity

Both functions take the address of a DIMMING (refer to page 46) structure as argument for output data (**citGetDimming**) or input data (**citSetDimming**).

In the file CITOUCHD.INI the following entries control the Touch Saver function:

[Settings]

DimmingHigh	Illumination intensity with inactive Touch Saver (regular operation, corresponds to *tsDimmHigh)*
DimmingLow	Illumination intensity with active Touch Saver (corresponds to *tsDimmLow)*
SaverScan	Corresponds to *tsTScan*
SaverTime	Corresponds to *tsTSaver*

5.1.5 Call of the API functions

An application program communicates with the CiTouchD driver via software interrupts. To do so, the program writes the function number into the AL register and then releases the respective software Interrupt. Additional parameters are written into the other registers of the processor. The exact usage of the registers is explained in the respective functional descriptions.

5.2 Emulated mouse driver functions of Int33-API

The Int 33-API (**A**pplication **P**rogram **I**nterface, Interrupt 33h) of the CiTouchD driver is compatible to the Microsoft® mouse driver version 8.0. Therefore, this chapter is confined to the particularities of some functions. A detailed description of each single function of the API is omitted.

5.2.1 Numerical summary

The following table lists the supported functions of the Int33-API in numerical order. The functions marked with a cross (†) can be called from the application program, however, they are not functional. The functions marked with an asterisk (*) vary in details from the Microsoft® mouse driver and are described further below.

Number	Designation	Description
00h	int33ResetAndStatus	Reset of driver and touch hardware
01h	int33ShowCursor	Show mouse cursor on screen
02h	int33HideCursor	Hide mouse cursor
03h	int33GetPosBtn	Determine mouse position and status of emulated mouse button
04h	int33SetCursorPos	Move mouse cursor
05h	int33GetBtnPress	Determine the number of operations of emulated mouse button
06h	int33GetBtnRelease	How often was the emulated mouse button released ?
07h	int33SetMinMaxHor	Set horizontal moving range of mouse cursor
08h	int33SetMinMaxVer	Set vertical moving range of mouse cursor
09h	int33SetGraphicsCursor	Define mouse cursor in video mode
0Ah	int33SetTextCursor	Define mouse cursor in text mode
0Bh	int33ReadMotionCounters	Read motion values
0Ch	int33InstallEvent	Install Event Handler
0Dh	int33LightPenOn	Activate emulation of light pen †
0Eh	int33LightPenOff	Deactivate emulation of light pen †
0Fh	int33SetMickeyPixelRatio	Set ratio between mickeys and pixels
10h	int33ConditionalOff	Define area in which mouse function is excluded
13h	int33SetDoubleThreshold	Set threshold for doubling of mouse speed. *
14h	int33SwapEvents	Swap Event Handlers
15h	int33GetStateSize	Determine size of touch status buffer
16h	int33SaveState	Save touch status
17h	int33RestoreState	Restore touch status
18h	int33SetAltEvent	Install alternative Event Handler
19h	int33GetAltEvent	Determine address of alternative Event Handlers
1Ah	int33SetSensitivity	Set mouse sensitivity *
1Bh	int33GetSensitivity	Request mouse sensitivity
1Ch	int33SetInterruptRate	Set Interrupt rate of mouse hardware *
1Dh	int33SetCrtPage	Set screen page for mouse cursor
1Eh	int33GetCrtPage	Determine screen page of mouse cursor
1Fh	int33Disable	Deactivate Touch driver
20h	int33Enable	Activate Touch driver
21h	int33SoftReset	Software reset of CiTouchD driver
22h	int33SetLanguage	Set language for messages *
23h	int33GetLanguage	Request language for messages
24h	int33Inquire	Inquire version, mouse type and IRQ *
25h	int33InquireEx	Inquire general information
26h	int33GetMouseScreen	Request expansion of the virtual mouse screen
27h	int33GetCursorMask	Request bit masks of screen cursor *

28h	int33SetVideoMode	Set video mode †
29h	int33InquireVideoModes	Inquire a list of available video modes *
2Ah	int33GetCursorInfo	Request information about mouse cursor
2Bh	int33SetAcceleration	Set acceleration curve *
2Ch	int33GetAcceleration	Read out current acceleration curve *
2Dh	int33GetSelectAccel	Set / request current acceleration curve *
2Fh	int33HardReset	Reset of Touch hardware
30h	int33InquireBallpoint	Determine / request settings of ballpoint mouse *
31h	int33GetVirtualScreen	Request expansion of the virtual screen
32h	int33InquireFunc	Inquire supported functions
33h	int33InquireAll	Inquire settings
34h	int33GetIni	Determine path to CITOUCHD.INI

5.2.2 Particularities of the Int 33 API emulation

Some functions of the Int 33 API vary in details from the Microsoft® mouse driver. These functions are described in the following.

Function 13h Set threshold for doubling of mouse speed. *

Input: AX = 0013h
DX = Threshold for doubling of mouse speed

Response: none

Particularities: The threshold for doubling of mouse speed in DX is internally saved, besides that, however, ignored.

Function 1Ah Set mouse sensitivity *

Input: AX = 001Ah
BX = Number of horizontal mickeys that correspond to 8 pixels
CX = Number of vertical mickeys that correspond to 8 pixels
DX = Threshold for doubling of mouse speed

Response: none

Particularities: The threshold for doubling of mouse speed in DX is internally saved, besides that, however, ignored.

Function 1Ch Set Interrupt rate of mouse hardware *

Input: AX = 001Ch
BX = Interrupt rate

Response: none

Particularities: The rate that was passed is internally saved, besides that, however, ignored.

Function 22h Set language for messages *

Input:
AX = 0022h
BX = Code number of language

Response:
none

Particularities:
Only code number 0 (= English) is supported. All other code numbers are disregarded.

Function 24h Inquire version, mouse type and IRQ *

Input:
AX = 0024h

Response:
BH = Pre-comma part of version number
BL = Post-comma part of version number
CH = Mouse type (always 2 = serial mouse)
CL = IRQ number

Particularities:
Since CiTouchD basically supports each IRQ number available on a PC, values greater than 7 can be returned to CL, too.

Function 27h Request bit masks of screen cursor *

Input:
AX = 0027h

Response:
AX = AND mask at software cursor / start line at hardware cursor
BX = XOR mask at software cursor / end line at hardware cursor
CX = Length of horizontal motion in mickeys
DX = Length of vertical motion in mickeys

Particularities:
Since an IRT works with absolute coordinates the values in CX and DX are always 0.

Function 29h Inquire a list of available video modes *

Input:
AX = 0029h
CX = 0 (request first video mode) or <> 0 (request next modes)

Response:
BX = Segment address of a string
CX = Code number of video mode
DX = Offset address of a string

Particularities:
In order to indicate that it is not possible to request modes, the value 0 is always returned to CX.

Function 2Bh Set acceleration curve *

Input:
AX = 002Bh
BX = Number of acceleration curve to be activated
ES = Segment address of identification data
SI = Offset address of identification data

Response:
AX = FFFFh (error) or 0021h (OK)
BX = Number of mouse buttons

Particularities:
Since acceleration curves in conjunction with an IRT do not make sense, a FFFFh is always returned to AX.

Function 2Ch Read out current acceleration curve *

Input: AX = 002Ch

Response: AX = Function status: <> 0 (error) or 0000h (OK)
BX = Number of current acceleration curve (0..3)
ES = Segment address of buffer
SI = Offset address of buffer

Particularities: Since acceleration curves in conjunction with an IRT do not make sense, a FFFFh is always returned to AX.

Function 2Dh Set / request current acceleration **curve ***

Input: AX = 002Dh
BX = -1 (request current acceleration curve) or
BX = 1..4 (activate this acceleration curve)

Response: AX = Function status: <> 0 (error) or 0000h (OK)
BX = Number of current acceleration curve (0..3)
ES = Segment address of buffer with designation of curve
SI = Offset address of buffer with designation of curve

Particularities: Since acceleration curves in conjunction with an IRT do not make sense, a FFFFh is always returned to AX.

Function 30h Determine / request settings of ballpoint **mouse ***

Input: AX = 0030h
BX = Rotation angle
CX = Command code

Response: AX = Function status: -1 (no ballpoint mouse) or <> 0 (OK)
BX = Rotation angle
CX = Active buttons

Particularities: Since CiTouchD does not support ballpoint mice, a FFFFh is always returned to AX.

5.3 Extended API functions

The extended API functions are accessible via the so-called Multiplex Interrupt 2Fh. By means of this Multiplex Interrupt various parameters and operational modes of the driver are requested or set, respectively. To do so, the identification number of the CiTouchD driver is written into the AH register and the number of the addressed function into the AL register. These are special API functions of the CiTouchD driver and are described in the following.
The mouse emulation of the CiTouchD driver can be entirely parameterized. These parameters are read from either the prompt at the program start or the file CITOUCHD.INI. The contents of the file CITOUCHD.INI are described at page 49. In order to change the parameter of the run time, a number of API functions are provided that are all implemented via an Int 2Fh (Multiplex Interrupt) interface. These functions are described in the following.

5.3.1 Determining the identification number of CiTouchD

Upon each call of an extended API function via the Multiplex Interrupt, the identification number of the CiTouchD driver has to be written into the AH register. The Multiplex Interrupt owes its name to the fact that by its means not only one single (DOS) program can be accessed, but that it is open-ended to all TSR

programs that require a communication interface to the outside. After a TSR program was called up for the first time and is installed consequently, further calls from the DOS prompt can be used to either set certain parameters in the running copy of this program or to remove it from the memory again. The newly called up program establishes the contact with its already installed part via the Multiplex Interrupt.
Any program that wants to use the Multiplex Interrupt (abbreviated: MUX), first needs to give itself an 8 bit identification number. The identification numbers from 00h up to BFh are reserved by Microsoft® for their own DOS programs. However, the range from C0h up to FFh is available and can be used by application programs, i.e. by own TSR programs.
Since the identification number can be freely chosen between C0h and FFh, it is may occur that several programs use the same number. In order to prevent this problem, any program that wants to use the Multiplex Interrupt needs to provide a function for an installation check. This function is carried out if at a Multiplex Interrupt the identification number of the respective program is written into the AH register and the value 00h into the AL register. If one of the already installed Handlers detects its identification number, it has to return a value unequal 00h (generally FFh) to the AL register. This way the CiTouchD driver runs through all identification numbers from C0h upwards and takes the first free one.
Since due to this behaviour the identification number of the CiTouchD is not certain in advance, any program that wants to use the driver functions provided via the Multiplex Interrupt 2Fh needs to request the currently valid identification number beforehand. In order to search for the correct identification number, starting at the identification number C0h (to be stored in register AH) the program calls the function 00h (to be stored in register AL) of the Multiplex Interrupt 2Fh as long until the ES:DI register pair points onto the string "**CiTouchD"**. The identification number that went with the successful function call equals the current identification number of the CiTouchD driver. Relating to this identification number, the functions of the CiTouchD that are provided via the Multiplex Interrupt can now be called.
If the search for the identification number remains without success until number FFh, the CiTouchD driver is not installed.
The following examples in the programming languages C, Pascal and Assembler utilize this method in order to request the existence of a LDVGA graphics board by means of the function 25h (citDetectLDVGA) which is described further below.

Example in C:

```
/*****************************************************************************
  detectLDVGA - Check if LDVGA graphics card is present (Microsoft C example)
*****************************************************************************/


#include <conio.h>
#include <dos.h|>
#include <stdio.h>
#include <string.h>

#define IDENSTR     "CiTouchD"
#define IDSTRLEN    8
#define PRESENCE    0x00     /* Literal for interrupt 2f/function 00h */
#define DETECTLDVGA 0x25     /* Literal for interrupt 2f/function 25h/37d */

int main( void )
{
    union _REGS ir, or;
    struct _SREGS sregs;
    int i;
    unsigned char idenNr; /* Identify number of CiTouchD */
    void far *pIdenStr;

    for( i=0xC0; i<0x100; i++ )    /* ID number from 0xC0...0xFF are allowed    */
    {                              /* for non system TSRs using MUX interrupt   */
        ir.h.al = PRESENCE;        /* Verify presence function                  */
        ir.h.ah = i;               /* Identify number                           */
        _int86x( 0x2F, &ir, &or, &sregs );

        /* Look, if a tsr with multiplex support is installed under
           this identify number                                                 */
        if( or.h.al == 0xFF )
        {
            /* point to given identifier string                                 */
            pIdenStr = (void far*) ( ((unsigned long)sregs.es << 16) + or.x.di);

            /* Look, if identifier string matches searched one                  */
            if( !_fstrncmp( pIdenStr, IDENSTR, IDSTRLEN ) )
```

```
            {
                break;
            }
        }
    }

    if( i == 256 )
        printf( "\nCiTouchD not installed!\n" );
    else
    {
        idenNr = ( unsigned char )i;       /* store identify number            */

        /* detect LDVGA */
        ir.h.al = DETECTLDVGA;
        ir.h.ah = idenNr;
        _int86x( 0x2F, &ir, &or, &sregs );
        if( or.x.ax )
            printf( "LDVGA detected.\n" );
        else
            printf( "No LDVGA detected.\n" );
    }

    return( 0 );
}
```

Example in Pascal:

```
(*************************************************************************
detectLDVGA -   Check if LDVGA graphics card is present (Turbo Pascal example)
 *************************************************************************)

uses dos;                               (* For 'Intr' *)

const
  IdenStr        = 'CiTouchD';
  PRESENCE       = $00;
  citDetectLDVGA = $25;

var
  Regs       : Registers;
  IdenNr     : Byte;
  IdenStrPtr : Pointer;
  TmpIden    : String;
  i          : Integer;
  j          : Byte;
  Found      : Boolean;

begin

  Found := False;
  i := $C0;                             (* ID number from $C0...$FF are
                                           allowed *)

  repeat
    Regs.AL := PRESENCE;                (* Verify presence function        *)
    Regs.AH := Byte( i );               (* Identify number                 *)
    Intr( $2F, Regs );

    (* Look, if a tsr with multiplex support is installed under this identfy number
*)
    if Regs.AL = $FF then begin
      IdenStrPtr := Ptr( Regs.ES, Regs.DI ); (* Point to given identifier string *)

      (* Convert given null terminated C-String in Pascal string and compare *)
      TmpIden := '';
      for j := 1 to 8 do begin
        TmpIden := TmpIden + Char( Ptr( Regs.ES, Regs.DI )^ );
        Regs.DI := Regs.DI + 1;
      end;

      if TmpIden = IdenStr then
        Found := TRUE;
    end;
    i := i + 1;
  until Found OR (i > $FF);
```

```
  if Not Found then
    writeln( 'CiTouchD not installed!' )
  else begin
    idenNr := Byte( i - 1);             (* store identify number *)

    (* detect LDVGA *)
    Regs.AL := citDetectLDVGA;
    Regs.AH := idenNr;
    Intr( $2F, Regs );
    if Regs.AX <> 0 then
      writeln('LDVGA found!')
    else
      writeln('No LDVGA detected.');
  end;
end.
```

Example in Assembler:

```
;////////////////////////////////////////////////////////////////////////////
;// S w i t c h A   -   Switch back to TBT/Assembler example
;// Assemble with MASM or TASM
;////////////////////////////////////////////////////////////////////////////

        .MODEL  small, pascal
        .DOSSEG

CR          EQU 13t
LF          EQU 10t
IDLEN       EQU 8t
PRESENCE    EQU 000h
DETECTLDVGA EQU 025h

        .STACK
        .DATA

MsgErr      BYTE    CR, LF, "CiTouchD not installed!", CR, LF, '$'
MsgFound    BYTE    CR, LF, "LDVGA found!", CR, LF, '$'
MsgNoLDVGA    BYTE    CR, LF, "No LDVGA detected.", CR, LF, '$'

IDStr   BYTE    'CiTouchD', 0           ;// Multiplex indentifier string
IDNr    BYTE    ?                       ;// Storage for found MUX code

        .CODE

Start   PROC    NEAR

        mov     ax, DGROUP              ;// Initialize DS
        mov     ds, ax


;// Start searching with number 0C0h...0FFh
        mov     bx, 0C0h                ;// only bl is of interest
                                        ;// (bx is used for easier overflow
                                        ;//  test)
        cld                             ;// String compare from left to right

SearchIDLoop:
        mov     al, PRESENCE            ;// verify presence function
        mov     ah, bl                  ;// next identifier number to search for
        int     02Fh

;// Look if a tsr with multiplex support is installed under this identify number
        cmp     al, 0FFh
        jne     SearchIDLoopNext

        ;// Look, if identify string matches
        ;// (ES:DI points to 'CiTouchD' if right tsr)
        mov     cx, IDLEN
        mov     si, OFFSET IDStr        ;// DS:SI points to compare string
        REPE    cmpsb
        jnz     SearchIDLoopNext        ;// No match -> carry on search
        mov     IDNr, bl                ;// else store ID number and
        jmp     Detect                  ;// search LDVGA

SearchIDLoopNext:
```

```
        inc     bx
        cmp     bx, 0100h               ;// look if search for all possible
                                        ;// ID numbers
        jne     SearchIDLoop            ;// -> if not next try

;// ID String not found -> quit with error message
        mov     dx, OFFSET MsgErr       ;// DS:DX points to message
        jmp     PrintOut

;// check presence of LDVGA via multiplex function DETECTLDVGA
Detect:
        mov     al, DETECTLDVGA
        mov     ah, IDNr                ;// use examined ID number
        int     02Fh
        or      ax, ax                  ;// have we found the LDVGA
        jz      NotFound                ;// -> NO
        mov    dx, OFFSET MsgFound      ;// DS:DX points to message
        jmp    PrintOut

NotFound:
        mov    dx, OFFSET MsgNoLDVGA    ;// DS:DX points to message
PrintOut:
        mov    ah, 09h                  ;// Request print string function 09h
        int    21h

MainExit:
        mov     ax, 04C00h              ;// end program
        int     21h

Start   ENDP

        END    Start
```

5.3.2 Summary of the extended API functions

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

5.3.2.1 Requesting CiTouchD parameters

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

No.	Designation	Returned data	Page
05h	**citGetFlags**	Current driver status	31
06h	**citGetSerialHardware**	Parameters of serial interface	31
07h	**citGetCommands**	Parameters for Mouse Button Emulation	31
08h	**citGetCoordMode**	Parameters for coordinates calculation	32
09h	**citGetCalibrationAbs**	Calibration parameters for absolute coordinates	32
0Ah	**citGetCalibrationRel**	Calibration parameters for relative coordinates	32
0Bh	**citGetDriverSettings**	Variable driver parameters	32
0Ch	**citGetTouchSettings**	Variable parameters of the IRT	33
0Dh	**citGetTouchHardware**	Unchanging parameters of the IRT	33
0Eh	**citGetDriverConstants**	Unchanging driver parameters	33
1Bh	**citGetVersion**	Versions number of CITOUCHD.EXE	38
1Dh	**citGetDimming**	Mark-to-space ratio of IRT's PWM output	38
20h	**citGetAcceleration**	Parameter for cursor acceleration	40
22h	**citGetButtonBeep**	Parameter for mouse clicks	40
27h	**citGetCitdFlags**	Extended driver status	42

5.3.2.2 Changing CiTouchD parameters

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

No.	Designation	Changed parameters	Page
0Fh	**citSetSerialHardware**	Parameters of serial interface	33

10h	**citSetCommands**	Parameters for Mouse Button Emulation	34
11h	**citSetCoordMode**	Parameters for coordinates calculation	34
12h	**citSetCalibrationAbs**	Calibration parameters for absolute coordinates	34
13h	**citSetCalibrationRel**	Calibration parameters for relative coordinates	35
14h	**citSetDriverSettings**	Variable driver parameters	35
15h	**citSetTouchSettings**	Variable parameters of the IRT	35
1Eh	**citSetDimming**	Mark-to-space ratio of IRT's PWM output	40
21h	**citSetAcceleration**	Parameter for cursor acceleration	40
23h	**citSetButtonBeep**	Parameter for mouse clicks	41
28h	**citSetCitdFlags**	Setting of the Hardware-Reset-Handling	42

5.3.2.3 Communication with the IRT

The CiTouchD driver provides an easy-to-handle interface for both transmitting commands to the IRT and receiving of reports from the IRT.
Before an application program is able to communicate with the IRT, the reception channel needs to be opened by means of the function **citOpen()**. However, a CiTouchD driver with an open reception 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.
Two methods are 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, an Event Handler of the application program is called up as soon as a complete report was received. The address of the Event Handlers is determined by the user when calling up the function **citOpen()**. At page 48 the structure of the notification is described.
If there is another report received by the IRT although the reception buffer was net 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.

No.	Designation	Function	Page
16h	**citSend**	Transmits one byte to the IRT	35
17h	**citReceive**	Receives a complete report from the IRT	36
18h	**citOpen**	Opens the reception channel between IRT and computer	36
19h	**citClose**	Closes the reception channel	36
1Ah	**citReceiveStatus**	Determines the status of the reception channel	38
1Ch	**citCheckBreak**	Checks a serial interface for 100ms Breaks	38

5.3.2.4 General help functions

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

No.	Command	Function	Page
00h	**citPresence**	Requests installation status	30
01h	**citGetPSP**	Determines PSP of resident driver	30
25h	**citDetectLDVGA**	Checks whether an LDVGA graphics board exists	41
26h	**citPlaySound**	Emits a sound to the PC speaker	41

5.3.3 Reference of the extended API functions

In the following a reference of the extended API functions is listed in numerical order. These functions can be accessed via the Int 2Fh (Multiplex Interrupt). How to determine the required identification number of the CiTouchD driver was already described earlier in this manual.

Function 00h — citPresence

Input:
AL = 00h
AH = Identification number of the CiTouchD driver

Response:
AL = 0FFh Identification number valid, ES:DI points onto a string with 8 characters of length.
AL = 00h No resident program with this identification number available. In this case is ES:DI invalid.

Description:
If CiTouchD is installed, this function returns the value FFh to the AL register and a pointer onto the identification string „CiTouchD“ (ASCIIZ-Format) to the ES:DI register pair. Exclusively in this case the other functions of the Multiplex Interrupt can be called up, too. The value to be written into register AH depends on the computer's configuration. For a detailed description refer to the chapter "Determining the identification number" at page 25.

Function 01h — citGetPSP

Input:
AL = 01h
AH = Identification number of the CiTouchD driver

Response:
ES:DI = Segment and offset of the resident part of CiTouchD

Description:
This function provides the PSP of the resident part of the driver.

Function 05h — citGetFlags

Input:
AL = 05h
AH = Identification number of the CiTouchD driver

Response:
AX = Current status of the CiTouchD driver

Description:
The current driver status is carried in the return word as bit flags. The single bits have the following meaning:

Bit 3	= IF_SMOOTH_ALWAYS	Smoothing is also active between two touchings
Bit 4	= IF_FIRST_COORDINATE	First coordinates message
Bit 5	= IF_BEEP_MODE	Acoustic signals as response to mouse clicks
Bit 6	= IF_ABSOLUTE_MODE	Absolute coordinates are used
Bit 7	= IF_Z_TOUCH	A 3D-IRT was detected
Bit 8	= IF_ENABLED	The driver is activated
Bit 11	= IF_INT_ON	The hardware Interrupts are enabled
Bit 12	= IF_BREAK	Breaks were received by IRT
Bit 13	= IF_AUTOREINIT	Automatic reinitialization is enabled
Bit 14	= IF_ON_SLAVEPIC	The used Interrupt vector is situated at the second PIC of the computer.
Bit 15	= IF_IRT_EXISTS	An IRT was detected

Function 06h — citGetSerialHardware

Input:
AL = 06h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a SERIALHARDWARE structure for the returning of the serial interface parameters.

Response:
AX = Number of bytes written into the SERIALHARDWARE structure

Description:
The serial interface parameters are written into the SERIALHARDWARE structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 47.

Function 07h — citGetCommands

Input:
AL = 07h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a COMMANDS structure for the returning of the Mouse Button Emulation parameters.

Response:
AX = Number of bytes written into the COMMANDS structure

Description:
The parameters for the Mouse Button Emulation are written into the COMMANDS structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 45. For a description of the function of the Mouse Button Emulation refer to chapter "User-defined key emulation" at page 12.

Function 08h citGetCoordMode

Input:
AL = 08h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a COORDMODE structure for the returning of the coordinates output parameters.

Response:
AX = Number of bytes written into the COORDMODE structure

Description:
The parameters for the coordinates output are written into the COORDMODE structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 46.

Function 09h citGetCalibrationAbs

Input:
AL = 09h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a CALIBRATIONABS structure for the returning of the absolute calibration parameters.

Response:
AX = Number of bytes written into the CALIBRATIONABS structure

Description:
The calibration parameters of the absolute coordinates mode are written into the CALIBRATIONABS structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 44.

Function 0Ah citGetCalibrationRel

Input:
AL = 0Ah
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a CALIBRATIONREL structure for the returning of the relative calibration parameters.

Response:
AX = Number of bytes written into the CALIBRATIONREL structure

Description:
The calibration parameters of the relative coordinates mode are written into the CALIBRATIONREL structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 44.

Function 0Bh citGetDriverSettings

Input:
AL = 0Bh
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a DRIVERSETTINGS structure for the returning of the driver parameters.

Response:
AX = Number of bytes written into the DRIVERSETTINGS structure

Description:
The variable driver parameters are written into the DRIVERSETTINGS structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 47.

Function 0Ch citGetTouchSettings

Input:
AL = 0Ch
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a TOUCHSETTINGS structure for the returning of the variable IRT parameters.

Response:
AX = Number of bytes written into the TOUCHSETTINGS structure

Description:
The variable IRT parameters are written into the TOUCHSETTINGS structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 48.

Function 0Dh citGetTouchHardware

Input:
AL = 0Dh
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a TOUCHHARDWARE structure for the returning of the unchanging IRT parameters.

Response:
AX = Number of bytes written into the TOUCHHARDWARE structure

Description:
The unchanging IRT parameters are written into the TOUCHHARDWARE structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 48.

Function 0Eh citGetDriverConstants

Input:
AL = 0Eh
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a DRIVERCONSTANTS structure for the returning of driver constants.

Response:
AX = Number of bytes written into DRIVERCONSTANTS structure

Description:
The unchanging driver parameters are written into the DRIVERCONSTANTS structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 46.

Function 0Fh citSetSerialHardware

Input:
AL = 0Fh
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a SERIALHARDWARE structure with the new serial interface parameters.

Response:
AX = 0001h The new parameters could be set..
AX = 0000h The parameters could not be changed.

Description:
The serial interface parameters are changed according to the SERIALHARDWARE structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 47. If the parameters could not be changed, the old status remains.

Function 10h citSetCommands

Input:
AL = 10h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a COMMANDS structure with the new parameters for the Mouse Button Emulation.

Response:
AX = 0001h The new parameters could be set.
AX = 0000h The parameters could not be changed.

Description:
The parameters for the Mouse Button Emulation are changed according to the COMMANDS structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 45. For a description of the function of the Mouse Button Emulation refer to chapter "User-defined key emulation" at page 12. If the parameters could not be changed, the old status remains.

Function 11h citSetCoordMode

Input:
AL = 11h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a COORDMODE structure with the new parameters for the coordinates output.

Response:
AX = 0001h The new parameters could be set.
AX = 0000h The parameters could not be changed.

Description:
The parameters for the coordinates output are changed according to the COORDMODE structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 46. If the parameters could not be changed, the old status remains.

Function 12h citSetCalibrationAbs

Input:
AL = 12h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a CALIBRATIONABS structure with the new absolute calibration parameters.

Response:
AX = 0001h The new parameters could be set.
EX = 0000h The parameters could not be changed.

Description:
The calibration parameters of the absolute coordinates mode are changed according to the CALIBRATIONABS structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 44. If the parameters could not be changed, the old status remains.

Function 13h citSetCalibrationRel

Input:
AL = 13h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a CALIBRATIONREL structure with the new relative calibration parameters.

Response:
AX = 0001h The new parameters could be set.
AX = 0000h The parameters could not be changed.

Description:
The calibration parameters of the relative coordinates mode are changed according to the CALIBRATIONREL structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 44. If the parameters could not be changed, the old status remains.

Function 14h citSetDriverSettings

Input:
AL = 14h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a DRIVERSETTINGS structure with the new driver parameters.

Response:
AX = 0001h The new parameters could be set.
AX = 0000h The parameters could not be changed.

Description:
The new driver parameters are changed according to the DRIVERSETTINGS structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 47. If the parameters could not be changed, the old status remains.

Function 15h citSetTouchSettings

Input:
AL = 15h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a TOUCHSETTINGS structure with the new IRT parameters.

Response:
AX = 0001h The new parameters could be set.
AX = 0000h The parameters could not be changed.

Description:
The new IRT parameters are changed according to the TOUCHSETTINGS structure the pointer of which was passed. For a description of this structure refer to page 48. If the parameters could not be changed, the old status remains.

Function 16h citSend

Input:
AL = 16h
AH = Identification number of the CiTouchD driver
BL = 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.

Function 17h citReceive

Input:
AL = 17h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a buffer the received report is copied to.

Response:
AX = Status of the reception buffer
Possible return values:

00h = CRS_IDLE	No report available
01h = CRS_READY	A report was received
02h = CRS_OVER	Further reports were received. However, the reported that was received first still completely exists in the buffer.
04h = CRS_CLOSE	The driver was not opened by **citOpen** (function **18h**).

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 reception buffer can be requested by means of the function **citGetDriverConstants** (function **0Eh**). Reports in the CTS1 protocol are already decoded, i.e. they do not contain DC2/DC4 and SYN sequences anymore.

Function 18h citOpen

Input:
AL = 18h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto either an Event Handler or NUL in case there are no notifications to be transmitted.

Response:
AX = 0001h The reception channel could be opened.
AX = 0000h The reception channel could not be opened. Possible causes for that are either an already opened reception channel or a driver that was not initialized.

Description:
Before an application program is able to receive reports from the IRT, the reception channel needs to be opened beforehand. However, a CiTouchD driver with an open reception channel does not report cursor motions or mouse clicks anymore. Therefore it is essential not to forget the call-up of **citClose** (function **19h**) after having received the desired reports!

Function 19h citClose

Input:
AL = 19h
AH = Identification number of the CiTouchD driver

Response:
AX = 0001h Reception channel could be closed
AX = 0000h Reception channel could not be closed. Possible causes for that are either an already closed reception channel or a driver that was not initialized.

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

Function 1Ah citReceiveStatus

Input:
AL = 1Ah
AH = Identification number of the CiTouchD driver

Response:
AX = Status of the reception buffer
Possible return values:

00h = CRS_IDLE	No report available
01h = CRS_READY	A report was received
02h = CRS_OVER	Further reports were received. However, the reported that was received first still completely exists in the buffer.
04h = CRS_CLOSE	The driver was not opened by **citOpen** (function **18h**).

Description:
The current status of the reception buffer is requested.

Function 1Bh Versions number of CITOUCHD.EXE

Input:
AL = 1Bh
AH = Identification number of the CiTouchD driver

Response:
DH = Main version number
DL = Sub version number
AH = Revision number
AL = Stage

Description:
The version number of CiTouchD is requested. However, by means of the function int33Inquire the Microsoft® mouse driver compatible version is returned.

Function 1Ch citCheckBreak

Input:
AL = 1Ch
AH = Identification number of the CiTouchD driver

Response:
AX = 0001h There were 100ms Breaks detected
AX = 0000h There were no 100ms Breaks detected

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

Function 1Dh citGetDimming

Input:
AL = 1Dh
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a DIMMING structure for the returning of dimming parameters.

Response:
AX = 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 the pointer of which was passed in ES:DI. For a description of this structure refer to page 46.

Function 1Eh citSetDimming

Input:
AL = 1Eh
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a DIMMING structure with the new parameters for the control of the IRT's PWM output.

Response:
AX = 0001h The new parameters could be set.
AX = 0000h The parameters could not be changed.

Description:
The parameters for the control of the IRT's PWM output are changed according to the DIMMING structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 46. If the parameters could not be changed, the old status remains.

Function 20h citGetAcceleration

Input:
AL = 20h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto an ACCELERATION structure for the returning of the parameters for the cursor acceleration.

Response:
AX = Number of bytes written into ACCELERATION structure

Description:
The parameters for the cursor acceleration are written into the ACCELERATION structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 43.

Function 21h citSetAcceleration

Input:
AL = 21h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto an ACCELERATION structure with the new parameters for the cursor acceleration.

Response:
AX = 0001h The new parameters could be set.
AX = 0000h The parameters could not be changed.

Description:
The parameters for the cursor acceleration are changed according to the ACCELERATION structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 43. If the parameters could not be changed, the old status remains.

Function 22h citGetButtonBeep

Input:
AL = 22h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a BUTTONBEEP structure for the returning of mouse click parameters.

Response:
AX = Number of bytes written into BUTTONBEEP structure

Description:
The parameters for the creation of a mouse click are written into the BUTTONBEEP structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 43.

Function 23h citSetButtonBeep

Input:
AL = 23h
AH = Identification number of the CiTouchD driver
ES:DI = Pointer onto a BUTTONBEEP structure with the new parameters for the mouse click creation.

Response:
AX = 0001h The new parameters could be set.
AX = 0000h The parameters could not be changed.

Description:
The parameters for the creation of a mouse click are changed according to BUTTONBEEP structure the pointer of which was passed in ES:DI. For a description of this structure refer to page 43. If the parameters could not be changed, the old status remains.

Function 25h citDetectLDVGA

Input:
AL = 25h
AH = Identification number of the CiTouchD driver

Response:
AX = 0001h A Citron LDVGA board was detected
AX = 0000h There is no Citron LDVGA board available.

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

Function 26h citPlaySound

Input:
AL = 26h
AH = Identification number of the CiTouchD driver
BX = Sound pitch in Hertz [Hz]
CX = Duration in milliseconds [ms]

Response:
none

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

Function 27h — Extended driver status

Input:
AL = 27h
AH = Identification number of the CiTouchD driver

Response:
AX = Current operating mode of the CiTouchD driver

Description:
The return value carries the operating mode as a bit sequence. The single bits have the following meaning:
Bit 0 = CTDF_VESA ->VGA board supports VESA standard
Bit 1 = CTDF_CHANGEMODE ->Video Mode was changed
Bit 2 = CTDF_TEXT -> Graphics board works in the text mode
Bit 3 = CTDF_SOFT -> Usage of the software text cursor
Bit 4 = CTDF_EGA -> EGA board found
Bit 5 = CTDF_VGA -> VGA board found
Bit 6 = CTDF_MONO -> monochrome EGA/VGA video mode
Bit 7 = CTDF_EXTVIDEO -> Function of extended Video BIOS
Bit 8 = CTDF_STACK -> CiTouchD uses own stack
Bit 9 = CTDF_HARDRESET0 -> Touch Reset in function Int33h/00h
Bit 10 = CTDF_HARDRESET2F-> Touch Reset in function Int33/2Fh
Bit 11 = CTDF_HIDE -> No own cursor in WINDOWS
Bit 12 = CTDF_SYS -> CiTouchD loaded in CONFIG.SYS
Bit 13 = CTDF_CONDOFF -> Conditional off is active

This version of CiTouchD does not support CTDF_EGA, CTDF_EXTVIDEO and CTDF_SYS yet. These bits are always 0.

Function 28h — Setting of the Hardware-Reset-Handling

Input:
AL = 28h
AH = Identification number of the CiTouchD driver
BX = Flags to be changed

Response:
AX = 0 in case the Read-Only-Flags are to be changed.

Description:
By changing the flags CTDF_HARDRESET0 Bit 9 and CTDF_HARDRESET2F Bit 10, the response of the Touch to a reset caused by the Int33h functions 00h and 2Fh can be changed.
All other flags are read-only and have to be set to 0 in BX.

5.3.4 Reference of the structures

In order to pass parameters to the extended API functions, numerous structures are defined. In the following these structures are listed in alphabetical order. Please note, that the API functions do not expect additional fill-up bytes between the structure elements. The following data types are used:

Type	Length	Value rage	Description
char	1 byte	-128…127	Signed integer value
BYTE	1 byte	0…255	Unsigned integer value
short	2 byte	-32768…32767	Signed integer value
WORD	2 byte	0…65535	Unsigned integer value
BOOL	2 byte	0, 1	Unsigned integer value. Values unequal 0 are interpreted as TRUE, the value 0 is interpreted as FALSE.
DWORD	4 byte	0…4294967295	Unsigned integer value

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** (function **0Eh**). The items *caMulX* or *caMulY*, respectively, can be requested by means of **citGetCalibrationAbs** (function **09h**).
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 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	The following constants determine the conditions for a beep to be created: 00h = BC_OFF — No beep creation 01h = BC_DOWN — Beep creation when emulated key is pressed 02h = BC_UP — Beep creation when emulated key is released

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.

$y = mx + t$, with y = calibrated coordinates and x = coordinates provided by IRT

CiTouchD expects calibrated 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 following values is used: 00h = OR_TOPLEFT – left-hand side top corner 01h = OR_TOPRIGHT – right-hand side top corner 02h = OR_BOTTOMRIGHT – right-hand side bottom corner 03h = OR_BOTTOMLEFT – left-hand side bottom corner

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** (function **0Eh**).

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 12.

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 conditions from 1 to 6:

Value	Designation	Meaning
00h	BM_NEVER	Never TRUE
01h	BM_IMMED	Always TRUE
02h	BM_ENTER	TRUE as soon as the Touch Zone is interrupted
03h	BM_ZPRESS	TRUE as soon as a certain pressure onto the Touch Zone is exceeded
04h	BM_DUAL	TRUE as soon as a dual touching of the Touch Zone occurs
05h	BM_TAP	TRUE when the Touch Zone is released and then interrupted again within a certain period of time ("tap")
06h	BM_LEAVE	TRUE as soon as Touch Zone is released
07h	BM_ZRLSE	TRUE as soon as the pressure exerted onto the Touch Zone falls below its limit value again
08h	BM_NODUAL	TRUE as soon as the dual touching of the Touch Zone ends

The following values can be used for the field "cmdModeChange":

Value	Designation	Meaning
00h	MC_NEVER	Never change mode
03h	MC_ZPRESS	TRUE as soon as a certain pressure onto the Touch Zone is exceeded
04h	MC_DUAL	TRUE as soon as a dual touching of the Touch Zone occurs
05h	MC_TAP	TRUE when the Touch Zone is released and then interrupted again within a certain period of time ("tap")

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.
		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.
		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** (function **15h**).
The value for *blSaverActive* is set by means of **citGetDimming** (function **1Dh**). The function **citSetDimming** (function **1Eh**) 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 CiTouchD 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 reception 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 CiTouchD 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. 1 = left-hand side mouse button 2 = right-hand side mouse button 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.

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-light barriers
WORD	thBeamsY	Number of physically present Y-light barriers
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) 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-light barriers
WORD	tsMinBeamsY	Minimum number of interrupted Y-light barriers
WORD	tsBeamTimeout	Blank-out time for defective light barriers 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.

5.3.5 Notification by the CiTouchD

If upon the call of **citOpen** (function **18h**) a pointer to an Event Handler was passed, CiTouchD calls up this Event Handler as soon as a complete report was received from the IRT. Please note that the Event Handler is called within the Interrupt-Service-Routine of CiTouchD and therefore should return as quickly as possibly. Especially DOS or BIOS functions, respectively, may not be called! Upon the call of the Event Handler the registers are set to the following values:

AX	Status of reception buffer (like at **citReceive**, function **17h**)
CX	Number of bytes in the reception buffer
ES:DI	Pointer onto the reception buffer, read-only-register!

The Event Handler has to return one of the following values to AX:

00h Reception buffer was not read, status of buffer not to be changed.
01h Reception buffer was read, contents of buffer to be erased.

6 CITOUCHD.INI

The CiTouchD driver saves its parameters in the file CITOUCHD.INI. When CiTouchD is started the values saved here are used unless they are overwritten by parameters entered at the prompt.

In the following the various sections and entries of the file CITOUCHD.INI are listed in alphabetical order. If an entry is omitted, either the default value that is saved in the driver or the respective prompt parameter is used.

Sections of the file CITOUCHD.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
[Citouchd]	Settings for the behaviour of CiTouchD as mouse driver

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 comparison 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 comparison 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 **256**, which equals a factor of 1,0.

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 **256**, which equals a factor of 1,0.

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** (function **0Eh**). 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 "Pre-defined Mouse Button Emulation Modes" at page 6.
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.
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:

$$\text{BaudRate} = \frac{f_{\text{Quarz}}}{16 \cdot \text{BaudDivisor}};$$ generally f_{Quarz} = 1.8432 MHz

The default value for BaudDivisor is **6** or 19200 baud if f_{Quarz} = 1.8432 MHz.

Interrupt = *0..15*

This entry determines the number of the ISA Bus Interrupt channel of the serial interface. The default value is **4**.

IO_Base = *0..65535*

This entry determines the base address of the serial interface module. The default value is **0x3f8**.

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 CiTouchD 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 light barriers. If an interruption of any IRT's light barrier 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 light barriers. 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. 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-light barriers 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-light barriers 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 **500**.

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**.

6.1.7 [Citouchd]

This section contains the parameters for the behaviour of the CiTouchD as mouse driver.
In the section **[Citouchd]** the following entries are possible.

Stack = 0..64

This entry determines the amount of stack that CiTouchD reserved for itself. The value entered here corresponds to the stack size in blocks of 256 byte. In case of Stack = 0 CiTouchD works with the stack of the currently running program. The default value is **6**.

HardReset = 0 | 1| 2

This entry determines whether a reset of the Touch is to be carried out in the mouse API functions (Software-Interrupt 33h) 00h (int33ResetAndStatus) and 2Fh (int33HardReset). Since (in case of failure) this process may take up to 10 seconds the reset can be deactivated. The default value is **0**.

Value	Meaning
0	A reset of the Touch is carried out neither in function 00h nor in function 2Fh.
1	A reset of the Touch is carried out in function int33HardReset only.
2	A reset of the Touch is carried out either in function 00h or in function 2Fh.

7 INDEX

F

H

I

L

M

N

P

S

T

U

V

X

Y

Z