

## Index

|  |    |
|--|----|
| Index  | 1  |
| TOUCHPANEL.TDD                                   | 2  |
| Installation of the driver (Standard settings)   | 2  |
| Installation of the driver (Individual settings) | 3  |
| Secondary addresses                              | 6  |
| User Function Codes                              | 7  |
| Touchpanel                                       | 10 |
| Read out current measurement                     | 14 |
| Touch panel Keyboard                             | 17 |
| Currently pressed button                         | 21 |
| Touch panel Keyboard key codes                   | 23 |
| Keyboard-Auto-Repeat                             | 27 |
| Beep   | 28 |
| Key attributes                                   | 32 |
| Read out alternative Keyboard buffer             | 33 |
| Slider definition                                | 36 |
| Slider output                                    | 38 |
| XY-Slider definition                             | 39 |
| XY-Slider output                                 | 41 |
| Incremental Touch area                           | 42 |
| XY-Slider output                                 | 44 |
| Read out and reset increments                    | 46 |
| Read out slider activity                         | 47 |
| Graphic with Touch panel                         | 48 |
| Graphic Delay                                    | 50 |
| Distance between 2 coordinates                   | 51 |
| Maximum variation                                | 54 |
| Pressed length                                   | 57 |
| Calibration of Touchpanel                        | 58 |
| TP_CALIBRATE.INC                                 | 60 |
| Example 1: Graphic Toolkit                       | 70 |
| Example 2: TEC-1000                              | 73 |
| Example 3: TP-1000                               | 76 |
| Documentation History                            | 78 |

## TOUCHPANEL.TDD

This driver controls Touch panels and supports easy touch-keyboard handling like the function INDEX\_2D.

For using ANALOG1.TDD or ANALOG2.TDD together with TOUCHPANEL.TDD, please set install device Parameter P26 to 1 (alternative method)

If ANALOG1.TDD or ANALOG2.TDD is used in your application, the analog driver must be installed before TOUCHPANEL.TDD

### Installation of the driver (Standard settings)

INSTALL DEVICE #*D*, "TOUCHPANEL.TDD" [, *P1*]

- D** is a constant, variable or an expression of data type BYTE, WORD, LONG in range of 0...63 and is the device number of the driver.
- P1** Touchpanel connection type (see table). If this parameter is not given, the default value 0 is set (TP-1000).

| No | Type     | Description   |
|----|----------|---|
| 0  | TP_TYP_0 | TP-1000 (use this mode, if you use ANOLOG1 or ANALOG2 parallel to the TOUCHPANEL)         |
| 1  | TP_TYP_1 | TP-1000 (stand alone, without other analog device drivers)                                |
| 2  | TP_TYP_2 | Graphic Toolkit (use this mode, if you use ANOLOG1 or ANALOG2 parallel to the TOUCHPANEL) |
| 3  | TP_TYP_3 | Graphic Toolkit (stand alone, without other analog device drivers)                        |
| 4  | TP_TYP_4 | TEC-1000 (use this mode, if you use ANOLOG1 or ANALOG2 parallel to the TOUCHPANEL)        |
| 5  | TP_TYP_5 | TEC-1000 (stand alone, without other analog device drivers)                               |

Installation of the driver (Individual settings)

INSTALL DEVICE #*D*, "TOUCHPANEL.TDD" [, *P1*, ..., *P27*]

**D** is a constant, variable or an expression of data type BYTE, WORD, LONG in range of 0...63 and is the device number of the driver.

**P1...P29** are more parameters, which changes the settings of the TOUCHPANEL.TDD driver.

|      | Default | Description of parameters   |
|------|---------|---|
| P1   | -       | Port  |
| P2   | -       | XPORT<br>0: YES, use XPORT<br>255: NO, use IPORT  |
| P3   | -       | Default Pin settings (XPORT)  |
| P4   | -       | Pin-number X-left   |
| P5   | -       | Level X-left<br>0: High active<br>255: Low active   |
| P6   | -       | Pin-number X-right  |
| P7   | -       | Level X-right<br>0: High active<br>255: Low active  |
| P8   | -       | Pin-number Y-top  |
| P9   | -       | Level Y-top<br>0: High active<br>255: Low active  |
| P10  | -       | Pin-number Y-bottom   |
| P11  | -       | Level Y-bottom<br>0: High active<br>255: Low active   |
| P12a | OEEH    | - No Multiplexer  |
| P12b | -       | - Pin-number Multiplexer 0...7: Pin number<br>- Level Multiplexer<br>0: X High active<br>255: Y High active |
| P13  | -       | ADC Input Y   |
| P14  | -       | ADC Input X   |
| P15a | -       | minimum ADC Value X (low Byte)  |
| P15b | -       | minimum ADC Value X (High Byte)   |
| P16a | -       | maximum ADC Value X (low Byte)  |
| P16b | -       | maximum ADC Value X (High Byte)   |
| P17a | -       | minimum ADC Value Y (low Byte)  |
| P17b | -       | minimum ADC Value Y (High Byte)   |

|      |   |   |
|------|---|---|
| P18a | - | maximum ADC Value Y (low Byte)  |
| P18b | - | maximum ADC Value Y (High Byte)   |
| P19a | - | Scan ADC Value (low Byte)   |
| P19b | - | Scan ADC Value Y (High Byte)  |
| P20a | - | minimum X coordinate (low Byte)   |
| P20b | - | minimum X coordinate (High Byte)  |
| P21a | - | maximum X coordinate (low Byte)   |
| P21b | - | maximum X coordinate (High Byte)  |
| P22a | - | minimum Y coordinate (low Byte)   |
| P22b | - | minimum Y coordinate (High Byte)  |
| P23a | - | maximum Y coordinate (low Byte)   |
| P23b | - | maximum Y coordinate (High Byte)  |
| P24  | - | Integration (2 to the power of n)<br>1: 2<br>2: 4<br>3: 8<br>4: 16<br>5: 32<br>6: 64<br>7: 128                                  |
| P25  | - | sample Method<br>0: normal Mode<br>1: alternative Mode, use this mode, if you use ANOLOG1 or ANALOG2 parallel to the TOUCHPANEL |
| P26  | - | Delay in ms from setting the ctrl-pins and sampling the analog values (1...255)   |
| P27  |   | Port of Beeper<br>0: use XPORT (same as in P1)  |
| P28  |   | Pin No. of Beeper   |
| P29a |   | Beeper length in ms (low Byte)  |
| P29b |   | Beeper length in ms(high Byte)<br>Default value: 50ms   |

## Secondary addresses

Write an option to the driver is possible to different secondary addresses:

| Secondary address | Function                       | Instruction |
|-------------------|--------------------------------|-------------|
| 0                 | Definition of Keys (DACC)      | PUT         |
| 1                 | Definition of graphic area     | PUT         |
| 2                 | Definition of key codes (DACC) | PUT         |

Read out the result is possible from different secondary addresses:

| Secondary address | Function  | Instruction |
|-------------------|---|-------------|
| 0                 | Read out Keyboard from Touch panel                            | GET         |
| 1                 | Read out graphic  | GET         |
| 3                 | Read out incremental slider                                   | GET         |
| 4                 | Read out second (alternative) Keyboard buffer                 | GET         |
| 7                 | Read out current ADC Values (OFFFH: Touch panel not pressed)  | GET         |
| 8                 | Read out current coordinates (OFFFH: Touch panel not pressed) | GET         |

## User Function Codes

User-Function-Codes of TOUCHPANEL.TDD to read out parameters (Instruction GET, secondary address 0):

| No.  | Symbol<br>Präfix UFCI_ | Description                         |
|------|------------------------|-------------------------------------|
| 01H  | UFCI_IBU_FILL          | No. of Bytes in input buffer (Byte) |
| 02H  | UFCI_IBU_FREE          | Free space in input buffer (Byte)   |
| 03H  | UFCI_IBU_VOL           | Size of input buffer (Byte)         |
| 063H | UFCI_DEV_VERS          | Driver version                      |
| 093H | TP_SCAN                | Current scan ADC Value              |

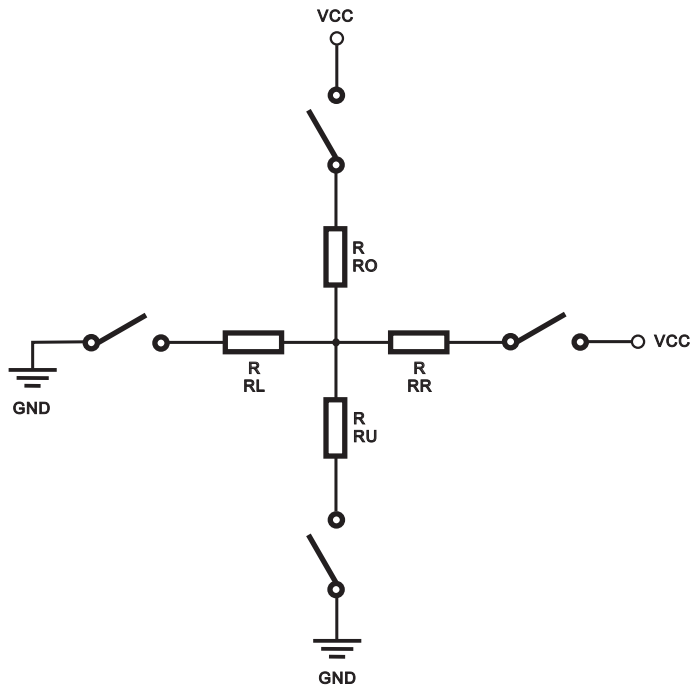
User-Function-Codes of TOUCHPANEL.TDD to set parameters (Instruction PUT, secondary address 0):

| No. | Symbol<br>Präfix: UFCO_ | Description  |
|-----|-------------------------|--|
| 1   | UFCO_IBU_ERASE          | Delete input buffer  |
| 80H | TP_START                | starts the device driver   |
| 81H | TP_STOP                 | stops the device driver  |
| 82H | TP_AUTOREPEAT_DELAY     | Delay (time in ms between keystroke and first repetition of code) (0=NO Auto Repeat)                               |
| 83H | TP_AUTOREPEAT_RATE      | Repeat rate in ms (a new code is generated by the touch panel keyboard every x ms)                                 |
| 84H | TP_GRA_MS               | Minimum time in ms between two coordinates in graphic mode   |
| 85H | TP_GRA_MIN_D            | Minimum distance between two coordinates in graphic mode   |
| 86H | TP_KEY_ATTRIBUTES       | assign attributes of keys ( <b>PUT to secondary address 2 ONLY</b> )   |
| 87H | TP_CALIBRATE            | recalibrates the Touchpanel ( <b>PUT to secondary address 2 ONLY</b> )   |
| 88H | TP_PRESSED_KEY          | defines String with pressed key information ( <b>PUT to secondary address 2 ONLY</b> )                             |
| 89H | TP_KEY_GRA              | Graphic start points of keys ( <b>PUT to secondary address 2 ONLY</b> )  |
| 8AH | TP_DEFINE_SLIDER        | Defines the slider areas of the Touchpanel ( <b>PUT to secondary address 2 ONLY</b> )                              |
| 8BH | TP_SLIDER_OUT           | Defines slider output string ( <b>PUT to secondary address 2 ONLY</b> )  |
| 8CH | TP_MAX_VARIATION        | Defines the maximum variation  |
| 8DH | TP_VARIATION_BUF_LEN    | Defines the number of required recurrences   |
| 8EH | TP_MAX_OUTLIER          | Defines the maximum number of outliers in variation buffer   |
| 8FH | TP_BEEP                 | Activates / Deactivates the Beeper   |
| COH | TP_BEEP_LEN             | Defines the length of the Beep in ms   |
| C1H | TP_SLIDER_FLAG          | Defines variable (Flag), which signals, if any slider value has changed ( <b>PUT to secondary address 2 ONLY</b> ) |

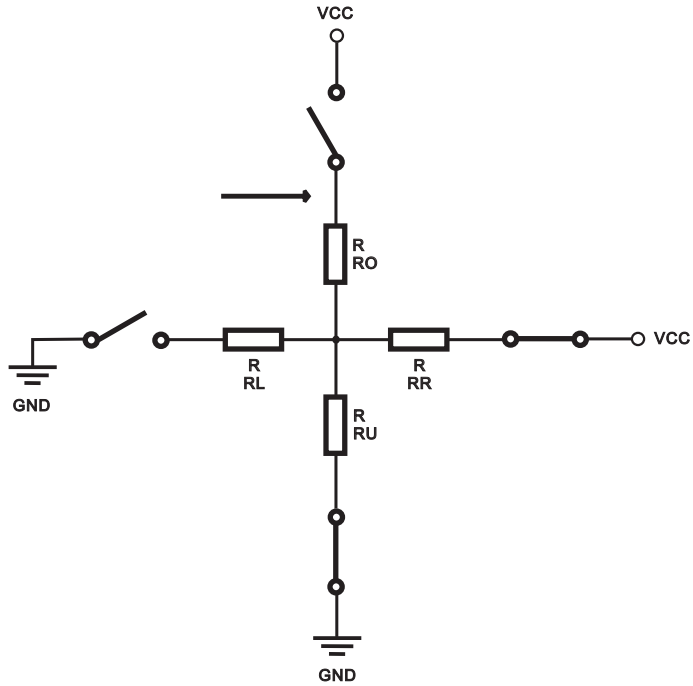
| No. | Symbol<br>Präfix: UFCO_ | Description   |
|-----|-------------------------|---|
| C2H | TP_PRESSED_NUM          | Specifies the number of samples, until the touchpanel is detected as pressed. |

## Touchpanel

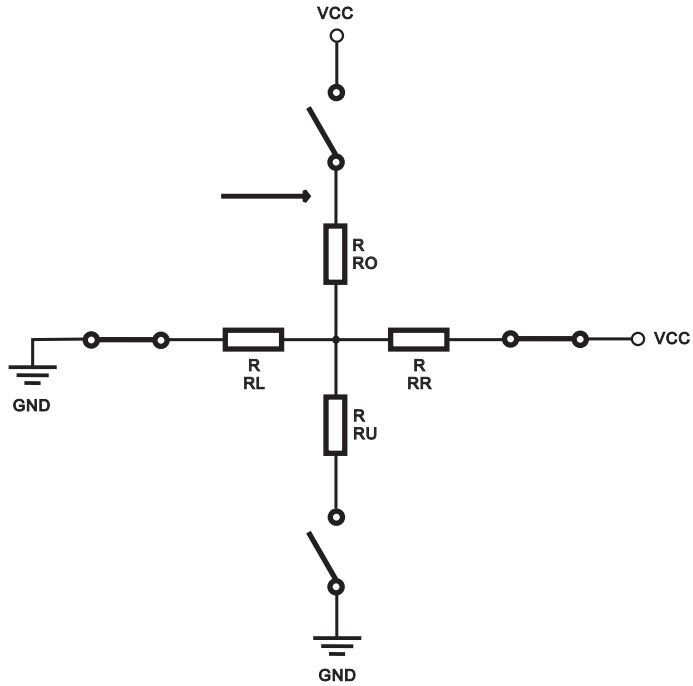
The analog Touch panel consists of two superimposed resistance films on which a voltage can be switched in a horizontal or vertical direction via a transistor. If you now press the Touch panel at one point an electrical contact is made and a voltage divider in a horizontal and vertical direction arises. The position of the pressure is determined by applying successive voltages in a horizontal and vertical direction and measuring the values of the voltage distribution with an analog input on the BASIC-Tiger®.



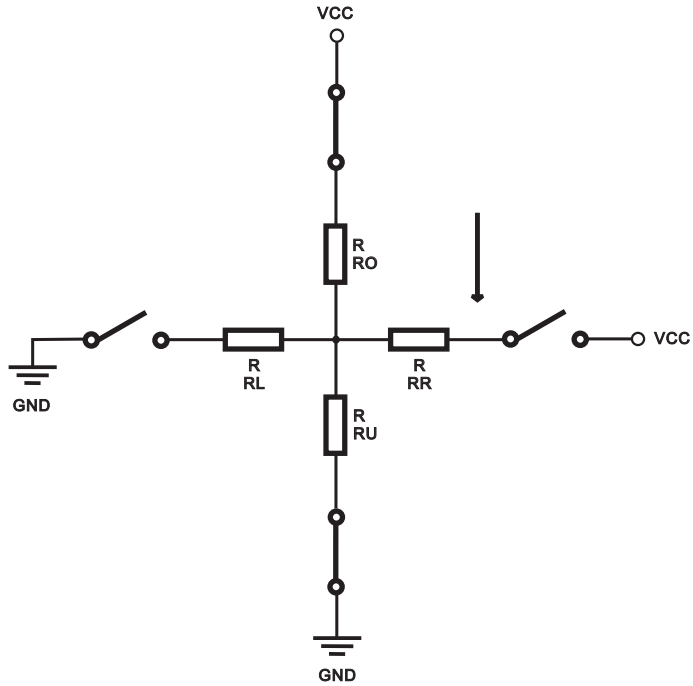
To determine that touch panel was pressed, X right and Y bottom is active and the multiplexer pin is switched to inactive. The analog Input ADC Input X is read.



To determine the x coordinates, X right and X left is active and the multiplexer pin is switched to inactive. The analog Input ADC Input X is read.



To determine the y coordinates, Y top and Y bottom is active and the multiplexer pin is also switched to active. The analog Input ADC Input Y is read.



**Read out current measurement**

With secondary addresses 7 and 8 you can read out the current state of the X and Y coordinates or the ADC value. The User Function Code TP\_SCAN is for checking, if the touch panel is pressed or not. With TP\_SCAN you get the analog value of course

**GET #D, #7, Number, Variable**

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**Number** is a constant, a variable or expression of the data type BYTE, WORD, LONG and specifies the length of output. The correct number is 0 or 4!!

**Variable** is a variable of the data type LONG or STRING which contains the analog values (2 Byte X coordinate and 2 Bytes Y coordinate)

The result of secondary address 7 is of 4 Bytes. The first 2 Bytes are the ADC Value of the x-axis and the last 2 Bytes the ADC Value of the y-axis.

If the touch panel is not pressed, the result is always 0FFFFFFH!!

## TOUCHPANEL.TDD

### GET #D, #8, Number, Variable

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**Number** is a constant, a variable or expression of the data type BYTE, WORD, LONG and specifies the length of output. The correct number is 0 or 4!!

**Variable** is a variable of the data type LONG or STRING which contains the absolute coordinates (2 Byte X coordinate and 2 Byte Y coordinate)

The result of secondary address 8 is also of 4 Bytes. The first 2 Bytes here are the absolute x-coordinates, and the last 2 Bytes are absolute y-coordinates. The driver calculates the coordinates with the minimum and maximum parameters from the install device line.

If the touch panel is not pressed, the result is always OFFFFFFFFH!!

## TOUCHPANEL.TDD

### GET #D, #0, #TP\_SCAN, Number, Variable

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**Number** is a constant, a variable or expression of the data type BYTE, WORD, LONG and specifies the length of output. The result should be at least 2 Byte.

**Variable** is a variable of the data type WORD, LONG or STRING which contains the result of the scan process (Touchpanel pressed?)

Program example:

```
PUT      #TP, #0, #TP_START, 0           ' Start Device Treiber

GET      #TP, #0, #TP_SCAN, 0, AD_PRESSED
GET      #TP, #7, 4, AD_KOORDS$
GET      #TP, #8, 4, KOORDS$
PRINT    #LCD, "<1>";"SCAN : "; AD_PRESSED

PRINT    #LCD,      "X-ADC: "; NFROMS (AD_KOORDS$, 0, 2)
PRINT    #LCD,      "Y-ADC: "; NFROMS (AD_KOORDS$, 2, 2)
PRINT    #LCD,      "X_COR: "; NFROMS (KOORDS$, 0, 2)
PRINT    #LCD,      "Y_COR: "; NFROMS (KOORDS$, 2, 2)
```

## Touch panel Keyboard

### PUT #D, #0, boxes\$

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**boxes\$** is a global or task global variable of the data type STRING and specifies the keys in a table of the spread areas in pairs of X-Y coordinates. (DACC)

It's easy to use the touch panel as a keyboard with up to 256 keys. This function assigns index numbers to several areas of the touch panel. If they are pressed (touched), the index is written to the input buffer.

To define the key areas, you have to generate a **global or task global variable of the type STRING**, which contains a table of the spread areas in pairs of X-Y coordinates. The first X-Y coordinate specifies the top left corner of the area and the second specifies the bottom right corner as WORD coordinates. 4 WORD values representing the two X-/Y coordinates. The first area in the string gets the index 0, the second gets the index 1 and so on.

If areas are overlapping, the returned index is always the lowest.

By default, there is no auto repeat function. To activate the auto repeat, please set TP\_AUTOREPEAT\_DELAY and TP\_AUTOREPEAT\_RATE. These commands adjust the delay and repeat rate of the touch panel keyboard. Delay adjusts the time in ms between keystroke and first repetition of code. To deactivate this function, set this parameter to 0. The repeat rate adjusts the time in ms for every new code generation after the first delay.

TOUCHPANEL.TDD

GET #D, #0, Number, Variable

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**Number** is a constant, a variable or expression of the data type BYTE, WORD, LONG and specifies the length of output.

**Variable** is a variable of the data type BYTE, WORD, LONG or STRING to read out input buffer from Touchpanel keyboard.

Program example:

```

#include LCD_4.INC
#include TP_CALIBRATE.INC

user_var_strict
'.....
' variables
'.....
string lcd$(9600)
task main

    string boxes$(20)
    string erg$(1)
    lcd$ = fill$("<0>", max_len(lcd$))
    boxes$ = ""

'.....
' reset LCD
'.....
DIR_PIN 8, 5, 0           ' L85 is Reset pin of the LCD
OUT 8, 00100000b, 0      ' Reset the LCD
OUT 8, 00100000b, 255    ' Be sure there is no reset more
WAIT_DURATION 100       ' wait that the LCD is not busy

'.....
' LCD backlight on
'.....
dir_pin 8, 2, 0
out 8, mask(2), 0

'.....
' install device drivers
'.....
INSTALL_DEVICE #TP, "TOUCHPANEL.TD2", TYP_TP_1000_ADMOD0

INSTALL_DEVICE #LCD, "LCD-S1D13700.TD2",0,0,0EEH, 1, 250, 2, 0

CALL vcalibrate_touchpanel( lcd$, TP, LCD)
'           ! ! !
'           ! ! !----- Device No. LCD-S1D13700
'           ! !----- Device No. TOUCHPANEL
'           !----- Screen String Lcd

'.....
' create buttons
'.....
call create_new_key(lcd$, boxes$, 32, 16, 96, 48) ' create key
call create_new_key(lcd$, boxes$, 160, 16, 224, 48) ' create key

PUT #LCD, #1, lcd$           ' print to lcd
PUT #TP, #0, boxes$         ' put keys to driver

PRINT #LCD, "<1BH>A"; CHR$(0); CHR$(10); "<0F0H>"; "Keys: ";

PUT #TP, #0, #TP_START, 0           ' Start touchpanel driver

```

```

long ibu_fill          ' input buffer len of touchpanel driver
byte idx

while 1=1              ' <===== LOOP =====>

    ibu_fill = 0
    while ibu_fill = 0 ' <==== LOOP =====>
        GET #TP, #0, #UFCI_IBU_FILL, 0, ibu_fill ' get buffer length
        release_task
    endwhile          ' <==== LOOP =====>

    GET #TP, #0, 1, idx ' get index from touchpanel keyboard

    PRINT #LCD, idx;   ' print to lcd

endwhile              ' <===== LOOP =====>

end

sub create_new_key(var string lcd$, boxes$; word xleft, ytop, xright,
ybottom)

' -----
' erstelle String fuer TP Treiber mit Tastenpositionen
' -----
boxes$ = ntos$(boxes$, len(boxes$), 2, xleft) '
boxes$ = ntos$(boxes$, len(boxes$), 2, ytop) '
boxes$ = ntos$(boxes$, len(boxes$), 2, xright) '
boxes$ = ntos$(boxes$, len(boxes$), 2, ybottom) '

' -----
' Zeichne jetzt die Grafik in den String
' -----

'
' <-Format-> Base Pt-1 Pt-2 scaling Rot Draw Pen
' Destin,wide high X Y X Y X Y X Y ang Mode No
DRAW_LINE
(lcd$,LCD_WIDTH,LCD_HEIGHT,0,0,xleft,ytop,xleft,ybottom,1000,1000, 0, 0,
0)
DRAW_NEXT_LINE(xright, ybottom, 0)
DRAW_NEXT_LINE(xright, ytop, 0)
CLOSE_LINE(0)
end

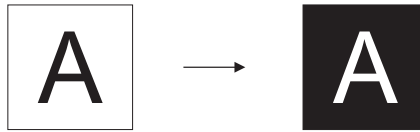
```

## Currently pressed button

PUT #D, #2, #TP\_PRESSED\_KEY, string\$

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**string\$** is a global or task global variable of the data type STRING (DACC) and is used to read out the current state of the buttons.



This function is used to detect the currently pressed button at the touchpanel. It's very easy to invert a pressed button at the touch field this way. You just have to watch this DACC String. The string is permanently written by the device driver.

To define the string, you have to generate a **global or task global variable of the type STRING with a length of min. 7**. The string contains the index and the key code of the button, a flag, if a button is pressed or not, and the coordinates of the button on the LCD.

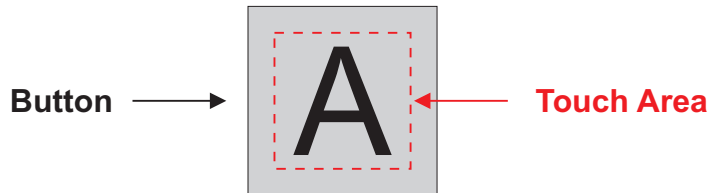
Structure of the string:

| Byte No. | Description  |
|----------|--|
| 0        | Index of pressed button                            |
| 1        | Key code of pressed button                         |
| 2        | Flag:<br>0: no button pressed<br>1: button pressed |
| 3-4      | X-position of top left corner                      |
| 5-6      | Y-position of top left corner                      |

PUT #D, #2, #TP\_PRESSED\_KEY, string\$

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**string\$** is a global or task global variable of the data type STRING (DACC) and is used to read out the current state of the buttons.



If the graphic coordinates differ from the touch coordinates, you can generate a string with graphic coordinates to put them in the DACC String for the currently pressed buttons.

To define the graphic areas, you have to generate a **global or task global variable of the type STRING**, which contains a table of the top left corners in pairs of X-Y coordinates. The first X-Y coordinate specifies the top left corner of the first button; the second specifies the top left corner of the second button. 2 WORD values representing the X-/Y coordinate of one button.

## Touch panel Keyboard key codes

### PUT #D, #2, key\_code\$

To define the key codes, you have to generate a **global or task global variable of the type STRING**. This command can be used to assign a character code up to 256 keys. The first Byte determines the assignment for the key with index 0, the second the assignment for the key with index 1. If index is greater than the key code table, this key will not be written to input buffer, so be sure the table is great enough. If the key code table is empty, no conversion will be done and the index will be put into input buffer.

This allows keyboard definition, i.e. which keys are assigned which code. Any number of keys can be assigned the same code. For example, if the code 97 is entered to bytes 15, 54 and 87, each of the associated keys is assigned the character "a".

Program example:

```

#INCLUDE LCD_4.INC
#include TP_CALIBRATE.INC

user_var_strict
' .....
' variables
' .....
string lcd$(9600)
task main

    string boxes$(20)
    string erg$(1)
    string attrib$(128)
    string charTab$(128)
    lcd$ = fill$("<0>", max_len(lcd$))
    boxes$ = ""

' .....
' reset LCD
' .....
DIR_PIN 8, 5, 0           ' L85 is Reset pin of the LCD
OUT 8, 00100000b, 0      ' Reset the LCD
OUT 8, 00100000b, 255    ' Be sure there is no reset more
WAIT_DURATION 100       ' wait that the LCD is not busy

' .....
' LCD backlight on
' .....
dir_pin 8, 2, 0
out 8, mask(2), 0

' .....
' install device drivers
' .....
INSTALL_DEVICE #TP, "TOUCHPANEL.TD2", TYP_TP_1000_ADMOD0

INSTALL_DEVICE #LCD, "LCD-S1D13700.TD2",0,0,0EEH, 1, 250, 2, 0

CALL vcalibrate_touchpanel( lcd$, TP, LCD)
'           !           !           !
'           !           !           !----- Device No. LCD-S1D13700
'           !           !           !----- Device No. TOUCHPANEL
'           !----- Screen String Lcd

' .....
' create buttons
' .....
call create_new_key(lcd$, boxes$, 32, 16, 96, 48)  ' create key
call create_new_key(lcd$, boxes$, 160, 16, 224, 48)  ' create key

PUT #LCD, #1, lcd$           ' print to lcd

' .....
' create convert tab & attributes
' .....
charTab$ = "AB"           ' (index -> character)

```

```

attrib$ = "<10H><00H>"          ' Attribute String

PUT #TP, #2, charTab$           ' activate index -> character
PUT #TP, #0, boxes$            ' put keys to driver
PUT #TP, #2, #TP_KEY_ATTRIBUTES, attrib$      ' attributes

' .....
' autorepeat options
' .....
PUT #TP, #0, #TP_AUTOREPEAT_DELAY, 1000 ' set autorepeat first delay
PUT #TP, #0, #TP_AUTOREPEAT_RATE, 500   ' repeat (every 0,5 seconds)

PRINT #LCD, "<1BH>A"; CHR$(0); CHR$(10); "<0F0H>";"Keys: ";

PUT #TP, #0, #TP_START, 0      ' Start touchpanel driver

long ibu_fill                  ' input buffer len of touchpanel driver

while 1=1                      ' <===== LOOP =====>

    ibu_fill = 0                ' init
    while ibu_fill = 0          ' <==== LOOP =====>
        GET #TP, #0, #UFCI_IBU_FILL, 0, ibu_fill ' get buffer length
        release_task            '
        endwhile               ' <==== LOOP =====>

        GET #TP, #0, 1, erg$    ' get key from touchpanel keyboard

        PRINT #LCD, erg$;      ' print to lcd

    endwhile                    ' <===== LOOP =====>

end

sub create_new_key(var string lcd$, boxes$; word xleft, ytop, xright,
ybottom)

' -----
' erstelle String fuer TP Treiber mit Tastenpositionen
' -----
boxes$ = ntos$(boxes$, len(boxes$), 2, xleft) '
boxes$ = ntos$(boxes$, len(boxes$), 2, ytop)  '
boxes$ = ntos$(boxes$, len(boxes$), 2, xright) '
boxes$ = ntos$(boxes$, len(boxes$), 2, ybottom) '

' -----
' Zeichne jetzt die Grafik in den String
' -----

'
' <-Format-> Base Pt-1 Pt-2 scaling Rot Draw Pen
' Destin,wide high X Y X Y X Y X Y ang Mode No

```

```
    DRAW_LINE  
(lcd$,LCD_WIDTH,LCD_HEIGHT,0,0,xleft,ytop,xleft,ybottom,1000,1000, 0, 0,  
0)  
    DRAW_NEXT_LINE(xright, ybottom, 0)  
    DRAW_NEXT_LINE(xright, ytop, 0)  
    CLOSE_LINE(0)  
end
```

## Keyboard-Auto-Repeat

PUT #D, #0, #TP\_AUTOREPEAT\_DELAY, n1

PUT #D, #0, #TP\_AUTOREPEAT\_RATE, n2

This command adjusts the delay and repeat rate of the Touch panel keyboard.

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**n1** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the delay (time between keystroke and generation of the 1st code) in ms (0=inactive)

**n2** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the repeat rate in ms (a new code is generated by the keyboard every n2 x ms)

If keys are to receive no auto-repeat function they have a special key attribute (See key attributes)

**Beep**

**PUT #D, #0, #TP\_BEEP, n1**

**PUT #D, #0, #TP\_BEEP\_LEN, n2**

This command switches the key click on or off and defines the length of the key click.

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**n1** is a constant, variable or expression of the data type BYTE, WORD, LONG and switches the key click on or off (0 = OFF, 1=ON)

**n2** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies length of the key click in ms.

If keys are to generate no key click, even if the Beep is enabled, they have a special key attribute (See key attributes)

You have to switch your beeper pin to OUTPUT in Basic before you can use the key click in TOUCHPANEL.TDD, if you use the beeper on an internal pin.

Program example:

```

user_var_strict
#include TP_CALIBRATE.INC

'.....
' variables
'.....
string lcd$(9600)
task main

    string boxes$(20)
    string erg$(1)
    string attrib$(128)
    string charTab$(128)
    lcd$ = fill$("<0>", max_len(lcd$))
    boxes$ = ""

'.....
' reset LCD
'.....
DIR_PIN 8, 5, 0           ' L85 is Reset pin of the LCD
OUT 8, 00100000b, 0      ' Reset the LCD
OUT 8, 00100000b, 255    ' Be sure there is no reset more
WAIT_DURATION 100       ' wait that the LCD is not busy

'.....
' LCD backlight on
'.....
dir_pin 8, 2, 0
dir_pin 4, 2, 0          ' Beeper Pin
out 4, 00000100B, 0     ' beeper aus
out 8, mask(2), 0

'.....
' install device drivers
'.....
INSTALL_DEVICE #TP, "TOUCHPANEL.TD2", TYP_TP_1000_ADMOD0

INSTALL_DEVICE #LCD, "LCD-S1D13700.TD2",0,0,0EEH, 1, 250, 2, 0

'CALL vcalibrate_touchpanel( lcd$, TP, LCD)
'           !           !           !
'           !           !           !----- Device No. LCD-S1D13700
'           !           !           !----- Device No. TOUCHPANEL
'           !           !           !----- Screen String Lcd
call vCalibrateTouchpanelFlash(lcd$, TP, LCD, 7, 0, 1, 0)
' calibrate from EEPROM if available

'.....
' create buttons
'.....
call create_new_key(lcd$, boxes$, 32, 16, 96, 48) ' create key
call create_new_key(lcd$, boxes$, 160, 16, 224, 48) ' create key

PUT #LCD, #1, lcd$           ' print to lcd

```

```

'.....
' create convert tab & attributes
'.....
charTab$ = "AB" ' (index -> character)

attrib$ = "<30H><00H>" ' Attribute String
' ! !
' ! !----- BEEP ON / Autorepeat on
' !----- BEEP OFF / Autorepeat off

PUT #TP, #2, charTab$ ' activate index -> character
PUT #TP, #0, boxes$ ' put keys to driver
PUT #TP, #2, #TP_KEY_ATTRIBUTES, attrib$ ' attributes
PUT #TP, #0, #TP_BEEP, TRUE ' BEEP ON

'.....
' autorepeat options
'.....
PUT #TP, #0, #TP_AUTOREPEAT_DELAY, 1000 ' set autorepeat first delay
PUT #TP, #0, #TP_AUTOREPEAT_RATE, 500 ' repeat (every 0,5 seconds)

PRINT #LCD, "<1BH>A"; CHR$(0); CHR$(10); "<0F0H>";"Keys: ";

PUT #TP, #0, #TP_START, 0 ' Start touchpanel driver

long ibu_fill ' input buffer len of touchpanel driver

while 1=1 ' <===== LOOP =====>

    ibu_fill = 0 ' init
    while ibu_fill = 0 ' <==== LOOP =====>
        GET #TP, #0, #UFCI_IBU_FILL, 0, ibu_fill ' get buffer length
        release_task '
    endwhile ' <==== LOOP =====>

    GET #TP, #0, 1, erg$ ' get key from touchpanel keyboard

    PRINT #LCD, erg$; ' print to lcd

endwhile ' <===== LOOP =====>

end

sub create_new_key(var string lcd$, boxes$; word xleft, ytop, xright,
ybottom)

'-----
' erstelle String fuer TP Treiber mit Tastenpositionen
'-----

boxes$ = ntos$(boxes$, len(boxes$), 2, xleft) '
boxes$ = ntos$(boxes$, len(boxes$), 2, ytop) '
boxes$ = ntos$(boxes$, len(boxes$), 2, xright) '

```

```
boxes$ = ntos$(boxes$, len(boxes$), 2, ybottom)
'-----
' Zeichne jetzt die Grafik in den String
'-----
'
'          <-Format-> Base Pt-1 Pt-2  scaling Rot Draw  Pen
'          Destin,wide high X  Y X   Y  X Y   X   Y ang Mode No
DRAW_LINE
(lcd$,LCD_WIDTH,LCD_HEIGHT,0,0,xleft,ytop,xleft,ybottom,1000,1000, 0,  0,
0)
DRAW_NEXT_LINE(xright, ybottom, 0)
DRAW_NEXT_LINE(xright, ytop, 0)
CLOSE_LINE(0)
end
```

## Key attributes

PUT #D, #2, #TP\_KEY\_ATTRIBUTES, attributes\$

This command is used to assign attributes of all keys. **PUT the attributes to secondary address 2 ONLY!**

**D** is a constant, variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**attributes\$** is a **global** or **task global** variable of data type STRING up to 256 Bytes that determine the attribute of the respective keys (DACC). The first byte determines the attribute for the key with index 0, the second the attribute for the key with index 1 etc.

These are the attributes for every key:

|   |  |
|---|--|
| 0 | no function                                  |
| 1 | no function                                  |
| 2 | no function                                  |
| 3 | no function                                  |
| 4 | Pressed key without auto-repeat              |
| 5 | Pressed key without beep                     |
| 6 | Use second (alternative) buffer for this key |
| 7 | no function                                  |

**4** Attribute 4 (no Auto-Repeat) has the effect that even with an active REPEAT function, no automatic character repeat is carried out for this key. This can be practical for keys such as Esc, Shift, Return or the Function keys.

**5** Attribute 5 (no Beep) has the effect that even with an active BEEP, no beep will be generated for this key, if it is pressed.

**6** Attribute 6 (alternative buffer) has the effect that the key is written into a separate alternative buffer. You can read out this buffer via secondary address 4.

## Read out alternative Keyboard buffer

### GET #D, #4, Number, Variable

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**Number** is a constant, a variable or expression of the data type BYTE, WORD, LONG and specifies the length of output.

**Variable** is a variable of the data type BYTE, WORD, LONG or STRING to read out input buffer from Touchpanel keyboard.

This command reads out the second (alternative) keyboard buffer. To use this buffer, you have to set attribute 6 of the button.

Program example:

```

user_var_strict
#include TP_CALIBRATE.INC

'.....
' variables
'.....
string lcd$(9600)
task main

    string boxes$(20)
    string erg$(1)
    string attrib$(128)
    string charTab$(128)
    lcd$ = fill$("<0>", max_len(lcd$))
    boxes$ = ""
    byte blAttribute

'.....
' reset LCD
'.....
DIR_PIN 8, 5, 0                ' L85 is Reset pin of the LCD
OUT 8, 00100000b, 0           ' Reset the LCD
OUT 8, 00100000b, 255         ' Be sure there is no reset more
WAIT_DURATION 100            ' wait that the LCD is not busy

'.....
' LCD backlight on
'.....
dir_pin 8, 2, 0
dir_pin 4, 2, 0                ' Beeper Pin
out 4, 00000100B, 0           ' beeper aus
out 8, mask(2), 0

'.....
' install device drivers
'.....
INSTALL_DEVICE #TP, "TOUCHPANEL.TD2", TYP_TP_1000_ADMOD0

INSTALL_DEVICE #LCD, "LCD-S1D13700.TD2",0,0,0EEH, 1, 250, 2, 0

' calibrate from EEPROM if available
call vCalibrateTouchpanelFlash(lcd$, TP, LCD, 7, 0, 1, 0)

'.....
' create buttons
'.....
call create_new_key(lcd$, boxes$, 32, 16, 96, 48) ' create key
call create_new_key(lcd$, boxes$, 160, 16, 224, 48) ' create key

PUT #LCD, #1, lcd$            ' print to lcd

'.....
' create convert tab & attributes
'.....
charTab$ = "AB"                ' (index -> character)

blAttribute = TP_KEY_ALTERNATIVE_BUFFER

```

```

attrib$ = chr$(blAttribute) & ' use alternative buffer for first key
          + "<00>"              ' use standard buffer for second key

PUT #TP, #2, charTab$          ' activate index -> character
PUT #TP, #0, boxes$           ' put keys to driver
PUT #TP, #2, #TP_KEY_ATTRIBUTES, attrib$ ' attributes

PRINT #LCD, "<1BH>A"; CHR$(0); CHR$(10); "<0F0H>";"Keys: ";

PUT #TP, #0, #TP_START, 0      ' Start touchpanel driver

long ibu_fill                  ' input buffer len of touchpanel driver

while l=1                      ' <===== LOOP =====>

    GET #TP, #0, #UFCI_IBU_FILL, 0, ibu_fill ' get buffer length
    if ibu_fill > 0 then
        GET #TP, #0, 1, erg$           ' get key from touchpanel keyboard
        PRINT #LCD, erg$;              ' print to lcd
    endif

    GET #TP, #4, #UFCI_IBU_FILL, 0, ibu_fill ' get buffer length
    if ibu_fill > 0 then
        GET #TP, #4, 1, erg$           ' get key from touchpanel keyboard
        PRINT #LCD, erg$;              ' print to lcd
    endif
endwhile                        ' <===== LOOP =====>
end

sub create_new_key(var string lcd$, boxes$;word xleft,ytop,xright, ybottom)
'-----
' create string for touchpanel.tdd with key positions
'-----
boxes$ = ntos$(boxes$, len(boxes$), 2, xleft) '
boxes$ = ntos$(boxes$, len(boxes$), 2, ytop) '
boxes$ = ntos$(boxes$, len(boxes$), 2, xright) '
boxes$ = ntos$(boxes$, len(boxes$), 2, ybottom) '

'-----
' draw button
'-----
DRAW_LINE (lcd$,LCD_WIDTH,LCD_HEIGHT,0,0,xleft,ytop,xleft,ybottom, &
          1000,1000, 0, 0, 0)
DRAW_NEXT_LINE(xright, ybottom, 0)
DRAW_NEXT_LINE(xright, ytop, 0)
CLOSE_LINE(0)
end

```

## Slider definition

PUT #D, #2, #TP\_DEFINE\_SLIDER, string\$

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**string\$** is a global or task global variable of the data type STRING (DACC) and is used to define the slider area.



This function is used to create slider areas such like a mixer. You can quickly generate values in a specified area.

To define the string, you have to generate a **global or task global variable of the type STRING**. The string contains the spread area in pairs of X-Y coordinates. The first X-Y coordinate specifies the top left corner of the area and the second specifies the bottom right corner as WORD coordinates. 4 WORD values representing the two X-/Y coordinates. Byte 8 specifies the type of the slider. There are 2 different sliders, the X- and the Y-slider. The Y-slider is a top-bottom and the X-Slider is a left-right slider. The next values representing the minimum and maximum values. If the minimum value is greater than the maximum value, the lower coordinate represents the higher result. The last parameter sets the offset for the coordinate. The current touched coordinate is written to the output string added with this offset.

To create more than one slider, just append them to the string. Every slider consists of 33 Bytes, so 2 sliders will need 66 Bytes in the definition string.

Structure of the string:

| Byte No. | Description                       |
|----------|-----------------------------------|
| 0-1      | X-Coordinate top left             |
| 2-3      | Y-Coordinate top left             |
| 4-5      | X-Coordinate bottom right         |
| 6-7      | Y-Coordinate bottom right         |
| 8        | “X” or “Y” for the type of slider |
| 9-12     | Minimum value                     |
| 13-16    | Maximum value                     |
| 17-20    | Coordinate offset                 |
| 21-24    | Dummy                             |
| 25-28    | Dummy                             |
| 29-32    | Dummy                             |

## Slider output

PUT #D, #2, # TP\_SLIDER\_OUT, string\$

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**string\$** is a global or task global variable of the data type STRING (DACC) and is used to read out slider output from the device driver.

This function is used to get the output information of the slider field from the Touchpanel. This variable has to be a **global or task global variable of the type STRING**.

The first 4 Bytes of the string is the result of the slider. This is the number you can work with. The next 2 Bytes shows the coordinate on the LCD. You can put your graphic button directly to this position. This coordinate is the real pressed coordinate added with the given offset of this touch area. The last Byte is a flag. If this flag is 0, the slider area is not pressed at the moment, otherwise this area is touched. It's possible to visualize this to LCD.

The first 13 Byte represents the first slider. If you have more than one slider, Byte 13-25 is output of the next slider and so on.

Structure of the string:

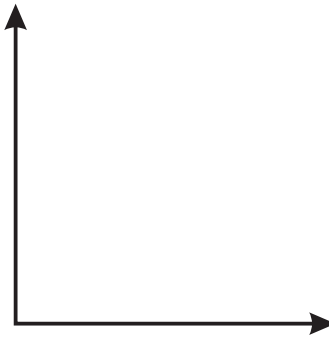
| Byte No. | Description  |
|----------|--------------|
| 0-3      | Result       |
| 4-5      | Coordinate   |
| 6        | Contact flag |
| 7-10     | Dummy        |
| 11-12    | Dummy        |

## XY-Slider definition

PUT #D, #2, #TP\_DEFINE\_SLIDER, string\$

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**string\$** is a global or task global variable of the data type STRING (DACC) and is used to define the slider area.



This function is used to create slider areas such like a coordinate system. You can quickly generate values in a specified area.

To define the string, you have to generate a **global or task global variable of the type STRING**. The string contains the spread area in pairs of X-Y coordinates. The first X-Y coordinate specifies the top left corner of the area and the second specifies the bottom right corner as WORD coordinates. 4 WORD values representing the two X-/Y coordinates. Byte 8 specifies the type of the slider This must be “D”. The next values representing the minimum and maximum values of the x-coordinate. If the minimum value is greater than the maximum value, the lower coordinate represents the higher result. The next parameter sets the offset for the x-coordinate. The current touched coordinate is written to the output string added with this offset. This is followed by the minimum, the maximum and the offset of the y-coordinate in this slider.

To create more than one slider, just append them to the string. Every slider consists of 33 Bytes, so 2 sliders will need 66 Bytes in the definition string.

Structure of the string:

| Byte No. | Description                    |
|----------|--------------------------------|
| 0-1      | X-Coordinate top left          |
| 2-3      | Y-Coordinate top left          |
| 4-5      | X-Coordinate bottom right      |
| 6-7      | Y-Coordinate bottom right      |
| 8        | “D” (Flag for the type)        |
| 9-12     | Minimum value X-coordinate     |
| 13-16    | Maximum value X-coordinate     |
| 17-20    | Coordinate offset X-coordinate |
| 21-24    | Minimum value Y-coordinate     |
| 25-28    | Maximum value Y-coordinate     |
| 29-32    | Coordinate offset Y-coordinate |

## XY-Slider output

PUT #D, #2, # TP\_SLIDER\_OUT, string\$

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**string\$** is a global or task global variable of the data type STRING (DACC) and is used to read out slider output from the device driver.

This function is used to get the output information of the slider field from the Touchpanel. This variable has to be a **global or task global variable of the type STRING**.

The first 4 Bytes of the string is the result of the sliders x-coordinate. This is the number you can work with. The next 2 Bytes shows the x-coordinate on the LCD. This coordinate is the real pressed coordinate added with the given offset of this touch area. The next Byte is a flag. If this flag is 0, the slider area is not pressed at the moment, otherwise this area is touched. It's possible to visualize this to LCD. The next values representing the result of the y-coordinate and the y-coordinate added with the offset.

The first 13 Byte represents the first slider. If you have more than one slider, Byte 13-25 is output of the next slider and so on.

Structure of the string:

| Byte No. | Description             |
|----------|-------------------------|
| 0-3      | Result X-coordinate     |
| 4-5      | Coordinate X-coordinate |
| 6        | Contact flag            |
| 7-10     | Result Y-coordinate     |
| 11-12    | Coordinate Y-coordinate |

## Incremental Touch area

PUT #D, #2, #TP\_DEFINE\_SLIDER, string\$

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**string\$** is a global or task global variable of the data type STRING (DACC) and is used to define the slider area.

This function is used to create slider areas such like a touch-pad mouse from laptops. You can quickly generate values in a specified area.

To define the string, you have to generate a **global or task global variable of the type STRING**. The string contains the spread area in pairs of X-Y coordinates. The first X-Y coordinate specifies the top left corner of the area and the second specifies the bottom right corner as WORD coordinates. 4 WORD values representing the two X-/Y coordinates. Byte 8 specifies the type of the slider. For incremental sliders you have to choose between “x” and “y”, or “d” for a XY-area. The next values representing the sensibility of the increments. If you choose “100”, 1 pixel is 1 increment. If you want to slow down the speed, choose a lower value, to increase the speed, choose a higher value than 100. The next parameter sets the offset for the coordinate(case “d” the x-coordinate). The current touched coordinate is written to the output string added with this offset. If you use type “d”, this is followed by the minimum, the maximum and the offset of the y-coordinate in this slider. Otherwise the rest are dummies.

You can read out and reset the number of increments with secondary address 3.

To create more than one slider, just append them to the string. Every slider consists of 33 Bytes, so 2 sliders will need 66 Bytes in the definition string.

Structure of the string for “x” and “y”:

| Byte No. | Description                       |
|----------|-----------------------------------|
| 0-1      | X-Coordinate top left             |
| 2-3      | Y-Coordinate top left             |
| 4-5      | X-Coordinate bottom right         |
| 6-7      | Y-Coordinate bottom right         |
| 8        | “x” or “y” for the type of slider |
| 9-10     | Sensibility                       |
| 11-16    | Dummy                             |
| 17-20    | Coordinate offset                 |
| 21-24    | Dummy                             |
| 25-28    | Dummy                             |
| 29-32    | Dummy                             |

Structure of the string for “d”:

| Byte No. | Description                    |
|----------|--------------------------------|
| 0-1      | X-Coordinate top left          |
| 2-3      | Y-Coordinate top left          |
| 4-5      | X-Coordinate bottom right      |
| 6-7      | Y-Coordinate bottom right      |
| 8        | “d” for the type of slider     |
| 9-10     | Sensibility X-coordinate       |
| 11-12    | Dummy                          |
| 13-14    | Sensibility Y-coordinate       |
| 15-16    | Dummy                          |
| 17-20    | Coordinate offset X-coordinate |
| 21-24    | Dummy                          |
| 25-28    | Dummy                          |
| 29-32    | Coordinate offset Y-coordinate |

## XY-Slider output

PUT #D, #2, # TP\_SLIDER\_OUT, string\$

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**string\$** is a global or task global variable of the data type STRING (DACC) and is used to read out slider output from the device driver.

This function is used to get output information of the slider field from the Touchpanel. This variable has to be a **global or task global variable of the type STRING**.

The first 4 Bytes of the string is a dummy in this case. The next 2 Bytes shows the first coordinate on the LCD. This coordinate is the real pressed coordinate added with the given offset of this touch area. The next Byte is a flag. If this flag is 0, the slider area is not pressed at the moment, otherwise this area is touched. It's possible to visualize this to LCD. The next values representing the result of the second coordinate added with the offset.

The first 13 Byte represents the first slider. If you have more than one slider, Byte 13-25 is output of the next slider and so on.

Structure of the string for type “x” and y”:

| Byte No. | Description  |
|----------|--------------|
| 0-3      | Dummy        |
| 4-5      | Coordinate   |
| 6        | Contact flag |
| 7-10     | Dummy        |
| 11-12    | Dummy        |

Structure of the string for type “x” and y”:

| Byte No. | Description             |
|----------|-------------------------|
| 0-3      | Dummy                   |
| 4-5      | Coordinate X-coordinate |
| 6        | Contact flag            |
| 7-10     | Dummy                   |
| 11-12    | Coordinate Y-coordinate |

## **Read out and reset increments**

With secondary address 3 you can read out and reset the increments of all incremental sliders defined with TP\_DEFINE\_SLIDER.

### **GET #D, #3, Number, Variable**

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**Number** is a constant, a variable or expression of the data type BYTE, WORD, LONG and specifies the length of output.

**Variable** is a variable of the data type STRING which contains the increments (4 Byte for each increment)

The result of secondary address 3 is of 4 Bytes per increment. The XY-Slider returns 8 Byte. The first 4 Bytes are the X-increments and the second the Y-increments. The counter is set to 0 after reading the value.

## Read out slider activity

With the User Function Code TP\_SLIDER\_FLAG, you can set an activity flag at secondary address 2. This flag signals, if a slider value has changed.

### PUT #D, #2, #TP\_SLIDER\_FLAG, Variable

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**Variable** is a global or task global variable of the data type BYTE, WORD or LONG.

If any slider value has changed, this Flag will be set to a 1. If Variable is  $\neq 0$ , then set Variable to 0 in Basic, before reading out all slider values. It is important to reset this flag in Basic, the device driver never resets this flag. TOUCHPANEL.TDD only sets this flag, if a slider value has changed. If no slider value has changed, this request is very fast-. Not every slider must be analyzed separately.

Activate flag:

```
bgTglSliderFlag = 0
PUT #TP, #2, #TP_SLIDER_FLAG, bgTglSliderFlag
```

Read and reset flag:

```
if bgTglSliderFlag <> 0 then ' any slider value changed ???
  bgTglSliderFlag = 0      ' YES, reset Flag and check all sliders
...
endif
```

## Graphic with Touch panel

Define the graphic area with the instruction:

**PUT #D, #1, GRAPHIC\_AREA\$**

To define the graphic areas, you have to generate a variable, a constant or expression of the type STRING, which contains the spread area in pair of X-Y coordinates. The first X-Y coordinate specifies the top left corner of the area and the second specifies the bottom right corner as WORD coordinates. 4 WORD values representing the two X-/Y coordinates.

## TOUCHPANEL.TDD

### GET #D, #0, Number, Variable

**D** is a constant, a variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**Number** is a constant, a variable or expression of the data type BYTE, WORD, LONG and specifies the length of output.

**Variable** is a variable of the data type BYTE, WORD, LONG or STRING to read out input buffer from Touchpanel Graphic Area..

You should read out the graphic buffer in 2 Byte steps. The control character has 2 Bytes length and every coordinate consists of 2 byte x coordinate and 2 byte y coordinate.

Special characters:

**OFFFEH: Touch panel was pressed**

**OFFFDH: Touch panel released**

If Touch panel is pressed, the code OFFFEH is written to input buffer and afterwards all sampled coordinates, unless the touch panel is released, then OFFFDH is written to input buffer. All coordinates consists of 2 WORDs, 1 WORD x-coordinate and 1 WORD y-coordinate. You can combine these coordinates e.g. with DRAW\_LINE and DRAW\_NEXT\_LINE.

## Graphic Delay

PUT #D, #0, #TP\_GRA\_MS, n

This command is used to assign the minimum delay between 2 coordinates in graphic mode

**D** is a constant, variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**n** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the minimum time in ms until the next coordinate is written to input buffer. This prevents overflow of data.

## Distance between 2 coordinates

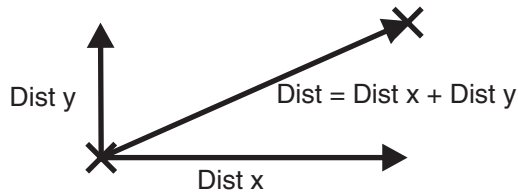
PUT #D, #0, #TP\_GRA\_MIN\_D, n

This command is used to assign the minimum distance between 2 coordinates in graphic mode

**D** is a constant, variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**n** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the minimum distance between 2 coordinates in input buffer. Coordinates with less distance are not written to input buffer.

This command specifies the minimum distance between two coordinates in input buffer. The distance is the sum of the absolute value of the difference of the two X and Y coordinates. Example: First coordinate is 10/20 and second coordinate is 11/18. The difference of 10 and 11 is 1, and the difference of 20 and 18 is 2, so the distance is  $1+2 = 3$ .



```

PUT #TP, #0, #TP_GRA_MS, 10          ' every 10ms new value minimum
PUT #TP, #0, #TP_GRA_MIN_D, 5       ' min. 5 Points distance

-----
' define graphic area
-----
PUT #TP, #1, "<0><0><0><0><240><0><120><0>"
'
'      ! ! ! ! ! ! ! !
'      ! ! ! ! ! ! ! !---- bottom right Y-coordinate
'      ! ! ! ! ! ! ! !---- bottom right Y-coordinate
'      ! ! ! ! ! ! ! !---- bottom right X-coordinate
'      ! ! ! ! ! ! ! !---- bottom right X-coordinate
'      ! ! ! ! ! ! ! !---- top left Y-coordinate
'      ! ! ! ! ! ! ! !---- top left Y-coordinate
'      ! ! ! ! ! ! ! !---- top left X-coordinate
'      ! ! ! ! ! ! ! !---- top left X-coordinate
-----

PUT #TP, #0, #TP_START, 0           ' start touchpanel device driver

long ibu_fill                       ' input buffer length

while 1=1                            ' <===== LOOP =====>

    ibu_fill = 0                     ' init
    while ibu_fill < 2               ' <==== LOOP =====>
        GET #TP, #1, #UFCI_IBU_FILL, 0, ibu_fill ' get buffer length
    endwhile                         ' <==== LOOP =====>

    GET #TP, #1, 2, xPos              ' get x-coordinate

    if xPos = 0FFFEH then            ' test: begin of line

        GET #TP, #1, #UFCI_IBU_FILL, 0, ibu_fill '
        while ibu_fill < 4           ' <==== LOOP =====>
            GET #TP, #1, #UFCI_IBU_FILL, 0, ibu_fill ' get buffer length
        endwhile                     ' <==== LOOP =====>

        GET #TP, #1, 2, xPos          ' first X-coordinate
        GET #TP, #1, 2, yPos          ' first Y-coordinate

'
'      <-Format-> Base Pt-1 Pt-2 scaling Rot Draw Pen
'      Destin,wide high X Y X Y X Y X Y ang Mode No
DRAW_LINE (lcd$,240,128,0,0,xPos,yPos,xPos,yPos,1000,1000, 0, 0, 0)
PUT #LCD, #1, lcd$                  ' show to lcd

    GET #TP, #1, #UFCI_IBU_FILL, 0, ibu_fill '
    while ibu_fill < 2               ' <==== LOOP =====>

```

```

        GET #TP, #1, #UFCI_IBU_FILL, 0, ibu_fill      ' get buffer length
    endwhile                                         ' <==== LOOP ====>

    GET #TP, #1, 2, xPos                             ' get next x-coordinate
    while xPos <> 0FFFDH                             ' is this end of line??

        GET #TP, #1, #UFCI_IBU_FILL, 0, ibu_fill      '
        while ibu_fill < 2                            ' <==== LOOP ====>
            GET #TP, #1, #UFCI_IBU_FILL, 0, ibu_fill  ' get buffer length
            endwhile                                  ' <==== LOOP ====>

            GET #TP, #1, 2, yPos                       ' get Y coordinate
            DRAW_NEXT_LINE(xPos, yPos, 0)              ' draw next point of line
            PUT _LCD, #1, lcd$                          ' to lcd

            GET #TP, #1, #UFCI_IBU_FILL, 0, ibu_fill  '
            while ibu_fill < 2                          ' <==== LOOP ====>
                GET #TP, #1, #UFCI_IBU_FILL, 0, ibu_fill  ' get buffer length
                endwhile                                  ' <==== LOOP ====>

                GET #TP, #1, 2, xPos                    ' get x-coordinate for next loop

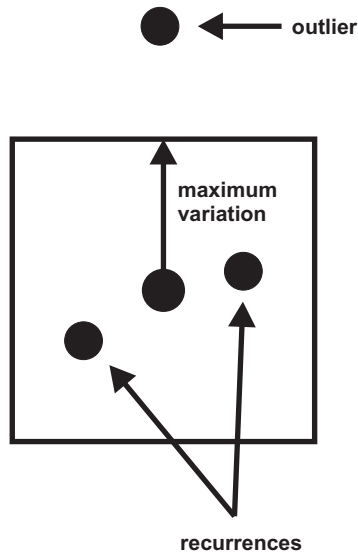
            endwhile                                     '
        endwhile
    endif

endwhile                                         ' <===== LOOP =====>

```

## Maximum variation

One important issue is to filter out bad data from the touchpanel unit. One possibility is to check the distance between the last points and detect outliers. A number of following samples must be located in a valid radius/box. This box can be specified with the User Function Code TP\_MAX\_VARIATION. The number of required recurrences is defined with TP\_VARIATION\_BUF\_LEN. The box must be filled with specified number of recurrences in the allowed radius, before a valid value is given out. The User Function Code TP\_MAX\_OUTLIER defines the number of allowed outliers, before the last sampled values are rejected.



## TOUCHPANEL.TDD

### PUT #D, #S, #TP\_MAX\_VARIATION, n

This command is used to assign the maximum variation/radius of the box.

**D** is a constant, variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**S** is a constant, variable or expression of the data type BYTE, WORD, LONG. To choose the graphic mode, select 1, otherwise use 0.

**n** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the maximum distance between the coordinates in pixels.

### PUT #D, #S, #TP\_VARIATION\_BUF\_LEN, n

This command is used to assign the number of required recurrences in a box.

**D** is a constant, variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**S** is a constant, variable or expression of the data type BYTE, WORD, LONG. To choose the graphic mode, select 1, otherwise use 0.

**n** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the number of required recurrences.

TOUCHPANEL.TDD

PUT #D, #S, #TP\_MAX\_OUTLIER, n

This command is used to define the maximum number of allowed outliers in variation buffer.

**D** is a constant, variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**S** is a constant, variable or expression of the data type BYTE, WORD, LONG. To choose the graphic mode, select 1, otherwise use 0.

**n** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the number of allowed outliers.

## Pressed length

This is also an important issue to filter out bad data. You can define the number of following correct samples, which detect that the touchpanel was pressed. This filters out erroneous data at the beginning of a touch process. One sample takes 2ms. If you chose 2 samples here, the touchpanel must be pressed 4ms before keys, sliders etc. are checked. The standard setting is 3.

**PUT #D, #0, #TP\_PRESSED\_NUM, n**

**D** is a constant, variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.

**n** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the number of samples until the touchpanel is detected as pressed. (1 sample = 2ms)

## Calibration of Touchpanel

**PUT #D, #2, #TP\_CALIBRATE, X1, Y1, ADX1, ADY1, X2, Y2, ADX2, ADY2, ADP**

This command is used to calibrate the Touchpanel during run time.

- D** is a constant, variable or expression of the data type BYTE, WORD, LONG in the range from 0→63 and stands for the device number of the drivers.
- X1** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the x coordinate of the first point.
- Y1** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the y coordinate of the first point.
- ADX1** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the analog value from the x axis of the first point.
- ADY1** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the analog value from the y axis of the first point.
- X2** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the x coordinate of the second point.
- Y2** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the y coordinate of the second point.
- ADX2** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the analog value from the x axis of the second point.
- ADY2** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the analog value from the y axis of the second point.
- ADP** is a constant, variable or expression of the data type BYTE, WORD, LONG and specifies the scan value of the touchpanel. If the analog value is greater than ADP, the touchpanel was pressed.



TOUCHPANEL.TDD

TP\_CALIBRATE.INC

This is a Basic library with lots of defines and subroutines for the touchpanel device driver.

**CALL vcalibrate\_touchpanel ( sLcd\$, bIDevNrTp, bIDevNrLcd)**

This subroutine is used to calibrate the Touchpanel during run time.

**sLcd\$** is a variable of the data type STRING with the lcd graphic data.

**bIDevNrTp** is a constant or variable of the data type BYTE in the range from 0→63 and stands for the device number of the Touchpanel device driver.

**bIDevNrLcd** is a constant or variable of the data type BYTE in the range from 0→63 and stands for the device number of the LCD device driver.

## TOUCHPANEL.TDD

**CALL vCalibrateTouchpanelFlash (spvLcd\$, bpDevNrTp, bpDevNrLcd, bpl2cPort, bpl2cClk, bpl2cData, wpl2cAdr)**

This subroutine is used to calibrate the Touchpanel during run time. If there are saved values in the EEPROM at the specified address on the TP-1000, no recalibration is needed and the saved values will be send to the device driver. Otherwise a calibration is started and the result is stored into the EEPROM at specified address.

**sLcd\$** is a variable of the data type STRING with the lcd graphic data.

**blDevNrTp** is a constant or variable of the data type BYTE in the range from 0→63 and stands for the device number of the Touchpanel device driver.

**blDevNrLcd** is a constant or variable of the data type BYTE in the range from 0→63 and stands for the device number of the LCD device driver.

**bpl2cPort** is a constant or variable of the data type BYTE and specifies the port for the I2C Bus.

**bpl2cClk** is a constant or variable of the data type BYTE and specifies the pin number of the CLOCK line for the I2C Bus.

**bpl2cData** is a constant or variable of the data type BYTE and specifies the pin number of the DATA line for the I2C Bus.

**wpl2cAdr** is a constant or variable of the data type WORD and specifies the address in the EEPROM, where calibration data is saved.

## TOUCHPANEL.TDD

CALL vCalibrateTouchpanelFlashForce (spvLcd\$, bpDevNrTp, bpDevNrLcd, bpl2cPort, bpl2cClk, bpl2cData, wpl2cAdr)

This subroutine is used to calibrate the Touchpanel during run time. A calibration is started and the result is stored into the EEPROM at specified address.

|                   |  |
|-------------------|--|
| <b>sLcd\$</b>     | is a variable of the data type STRING with the lcd graphic data.   |
| <b>blDevNrTp</b>  | is a constant or variable of the data type BYTE in the range from 0→63 and stands for the device number of the Touchpanel device driver. |
| <b>blDevNrLcd</b> | is a constant or variable of the data type BYTE in the range from 0→63 and stands for the device number of the LCD device driver.        |
| <b>bpl2cPort</b>  | is a constant or variable of the data type BYTE and specifies the port for the I2C Bus.  |
| <b>bpl2cClk</b>   | is a constant or variable of the data type BYTE and specifies the pin number of the CLOCK line for the I2C Bus.                          |
| <b>bpl2cData</b>  | is a constant or variable of the data type BYTE and specifies the pin number of the DATA line for the I2C Bus.                           |
| <b>wpl2cAdr</b>   | is a constant or variable of the data type WORD and specifies the address in the EEPROM, where calibration data is saved.                |

CALL **sInitSlider** ( **slResult\$**, **wlxPos**, **wlyPos**, **wlWidth**, **wlHeight**, **llMin**, **llMax**, **llOffs**, **slTyp\$** )

This subroutine is used to define 1 Slider in a string. (X and Y slider)

**slResult\$** is a variable of the data type STRING and the slider definition is saved in this string.

**wlxPos** is a constant or variable of the data type WORD and specifies the absolute x-coordinate of the starting point.

**wlyPos** is a constant or variable of the data type WORD and specifies the absolute y-coordinate of the starting point.

**wlWidth** is a constant or variable of the data type WORD and specifies the width of the slider.

**wlHeight** is a constant or variable of the data type WORD and specifies the height of the slider.

**llMin** is a constant or variable of the data type LONG and defines the minimum result value for the slider.

**llMax** is a constant or variable of the data type LONG and defines the maximum result value for the slider.

**llOffs** is a constant or variable of the data type LONG and specifies the coordinate offset.

**slTyp\$** is a constant or variable of the data type STRING and specifies the type of slider.

## TOUCHPANEL.TDD

CALL `sCreate_new_button ( sIScreen$, sIBoxes$, sIButton$, wIXleft, wIYtop, wIHeight, wIWidth, llBmpWidth, llBmpHeight)`

This subroutine is used to create a button for the touchpanel device driver and print the button bitmap to the LCD.

|                    |   |
|--------------------|---|
| <b>sIScreen\$</b>  | is a variable of the data type STRING with the lcd graphic data.  |
| <b>sIBoxes\$,</b>  | is a variable of the data type STRING with the definitions of the buttons. The new button is appended to this string.         |
| <b>sIButton\$</b>  | is a variable of the data type STRING. This is the bitmap of the button.  |
| <b>wIXleft</b>     | is a constant or variable of the data type WORD and specifies the absolute x-coordinate of the top left corner of the button. |
| <b>wIYtop</b>      | is a constant or variable of the data type WORD and specifies the absolute y-coordinate of the top left corner of the button. |
| <b>wIHeight</b>    | is a constant or variable of the data type WORD and specifies the height of the button.                                       |
| <b>wIWidth</b>     | is a constant or variable of the data type WORD and specifies the width of the button.  |
| <b>llBmpWidth</b>  | is a constant or variable of the data type WORD and specifies the width of the bitmap.  |
| <b>llBmpHeight</b> | is a constant or variable of the data type WORD and specifies the height of the bitmap.                                       |

TOUCHPANEL.TDD

CALL sCreate\_new\_button2 ( sBoxes\$, wXleft, wYtop, wHeight, wWidth)

This subroutine is used to create a button for the touchpanel device driver and without any bitmap.

**sBoxes\$,** is a variable of the data type STRING with the definitions of the buttons. The new button is appended to this string.

**wXleft** is a constant or variable of the data type WORD and specifies the absolute x-coordinate of the top left corner of the button.

**wYtop** is a constant or variable of the data type WORD and specifies the absolute y-coordinate of the top left corner of the button.

**wHeight** is a constant or variable of the data type WORD and specifies the height of the button.

**wWidth** is a constant or variable of the data type WORD and specifies the width of the button.

TOUCHPANEL.TDD

CALL defineGraphicArea ( wXpos, wYpos, wWidth, wHeight)

This subroutine is used to define the graphic area

**wXpos** is a constant or variable of the data type WORD and specifies the absolute x-coordinate of the top left corner.

**wYpos** is a constant or variable of the data type WORD and specifies the absolute y-coordinate of the top left corner.

**wWidth** is a constant or variable of the data type WORD and specifies the width of the graphic area.

**wHeight** is a constant or variable of the data type WORD and specifies the height of the graphic area.

**CALL slnitXYSlider ( slStri\$, wXPos, wYPos, wWidth, wHeight, llMin, llMax, lOffs, llMin2, llMax2, lOffs2 )**

This subroutine is used to define 1 XY-Slider in a string.

**slStri\$** is a variable of the data type STRING and the slider definition is saved in this string.

**wXPos** is a constant or variable of the data type WORD and specifies the absolute x-coordinate of the starting point.

**wYPos** is a constant or variable of the data type WORD and specifies the absolute y-coordinate of the starting point.

**wWidth** is a constant or variable of the data type WORD and specifies the width of the slider.

**wHeight** is a constant or variable of the data type WORD and specifies the height of the slider.

**llMin** is a constant or variable of the data type LONG and defines the minimum result value for the slider (x-coordinate).

**llMax** is a constant or variable of the data type LONG and defines the maximum result value for the slider (x-coordinate).

**lOffs** is a constant or variable of the data type LONG and specifies the coordinate offset (x-coordinate).

**llMin2** is a constant or variable of the data type LONG and defines the minimum result value for the slider (y-coordinate).

**llMax2** is a constant or variable of the data type LONG and defines the maximum result value for the slider (y-coordinate).

**lOffs2** is a constant or variable of the data type LONG and specifies the coordinate offset (y-coordinate).

CALL `sinitSlider` ( `sStri$`, `wXPos`, `wYPos`, `wlWidth`, `wlHeight`, `wlSens`, `lOffs`, `sTyp$`)

This subroutine is used to define 1 incremental Slider in a string. (X and Y slider)

- sStri\$** is a variable of the data type STRING and the slider definition is saved in this string.
- wXPos** is a constant or variable of the data type WORD and specifies the absolute x-coordinate of the starting point.
- wYPos** is a constant or variable of the data type WORD and specifies the absolute y-coordinate of the starting point.
- wlWidth** is a constant or variable of the data type WORD and specifies the width of the slider.
- wlHeight** is a constant or variable of the data type WORD and specifies the height of the slider.
- wlSens** is a constant or variable of the data type WORD and defines the sensibility of the slider.
- lOffs** is a constant or variable of the data type LONG and specifies the coordinate offset.
- sTyp\$** is a constant or variable of the data type STRING and specifies the type of slider.

## TOUCHPANEL.TDD

CALL `slnitlXYSlider` ( `slStri$`, `wlXPos`, `wlYPos`, `wlWidth`, `wlHeight`, `wlSens`, `wlSens2`, `lloffs`, `lloffs2`)

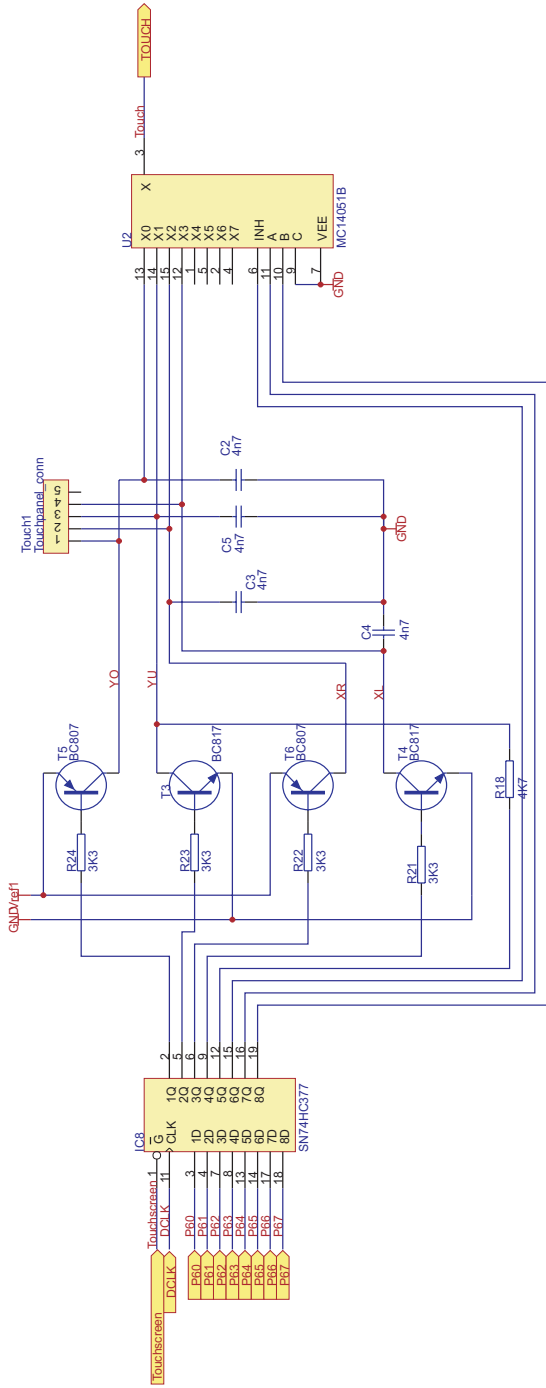
This subroutine is used to define 1 incremental XY-Slider in a string.

|                 |  |
|-----------------|--|
| <b>slStri\$</b> | is a variable of the data type STRING and the slider definition is saved in this string.                       |
| <b>wlXPos</b>   | is a constant or variable of the data type WORD and specifies the absolute x-coordinate of the starting point. |
| <b>wlYPos</b>   | is a constant or variable of the data type WORD and specifies the absolute y-coordinate of the starting point. |
| <b>wlWidth</b>  | is a constant or variable of the data type WORD and specifies the width of the slider.                         |
| <b>wlHeight</b> | is a constant or variable of the data type WORD and specifies the height of the slider.                        |
| <b>wlSens</b>   | is a constant or variable of the data type WORD and defines the sensibility of the slider. (X-coordinate)      |
| <b>wlSens2</b>  | is a constant or variable of the data type WORD and defines the sensibility of the slider. (Y-coordinate)      |
| <b>lloffs</b>   | is a constant or variable of the data type LONG and specifies the coordinate offset. (X-coordinate)            |
| <b>lloffs2</b>  | is a constant or variable of the data type LONG and specifies the coordinate offset. (Y-coordinate)            |

### Example 1: Graphic Toolkit

This is the example of the Basic Tiger Graphic Toolkit. It uses the XPORT address 40H. The analog Input pin is AN2. The control pins are connected:

|              |   |                                      |
|--------------|---|--------------------------------------|
| X left :     | 3 | (High active)                        |
| X right:     | 2 | (Low active)                         |
| Y top:       | 0 | (Low active)                         |
| Y bottom:    | 1 | (High active)                        |
| Multiplexer: | 7 | (Multiplexer Pin 6 is NOT necessary) |



## TOUCHPANEL.TDD

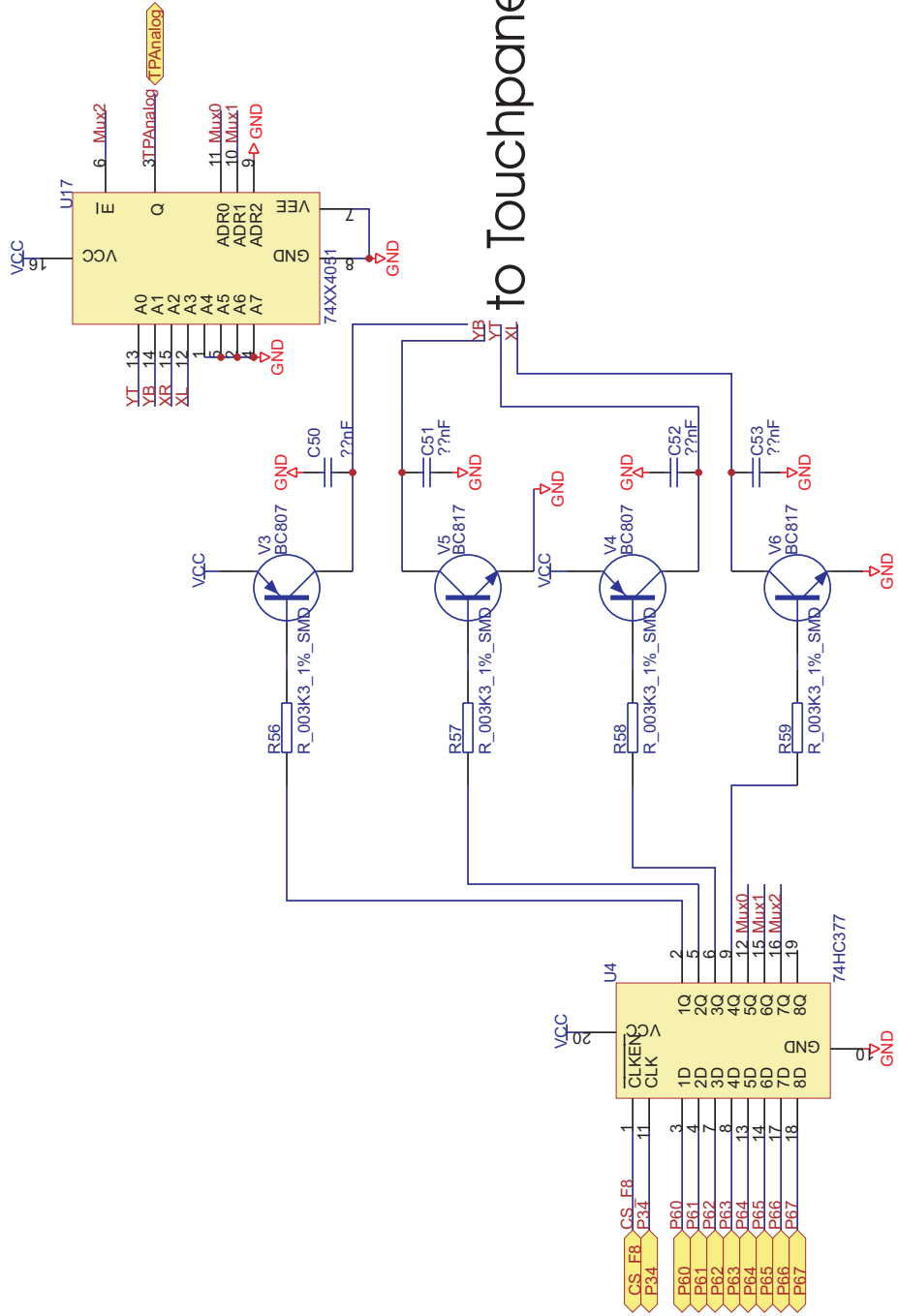
Init of TOUCHPANEL.TDD with this circuit:

```
INSTALL_DEVICE #TP, "TOUCHPANEL.TDD", &
040H, & ' Port address
0, & ' use XPORT here
00000000b, & ' Default Pins of Port
3, & ' Bitnumber XL (X-Left)
HIGH_ACTIVE, & ' High Active
2, & ' Bitnumber XR (X-Right)
LOW_ACTIVE, & ' Low active
0, & ' Bitnumber YT (Y-Top)
LOW_ACTIVE, & ' Low active
1, & ' Bitnumber YB (Y-Bottom)
HIGH_ACTIVE, & ' High active
7, & ' Pin Number Multiplexer
X_MUX_HIGH_ACTIVE, & ' 0: Xi = High Active
2, & ' Input Y (ADC-Ch.)
2, & ' Input X (ADC-Ch.)
57H, & ' AD-MIN X low Byte
0H, & ' AD-MIN X High Byte
035H, & ' AD-MAX X low Byte
3H, & ' AD-MAX X High Byte
88H, & ' AD-MIN Y low Byte
0H, & ' AD-MIN Y High Byte
50H, & ' AD-MAX Y low Byte
3H, & ' AD-MAX Y High Byte
39, & ' AD-PRESS low Byte
0H, & ' AD-PRESS High Byte
0H, & ' Coord X Min (Low Byte)
0H, & ' Coord X Min (High Byte)
0F0H, & ' Coord X Max (Low Byte)
0H, & ' Coord X Max (High Byte)
0H, & ' Coord Y Min (Low Byte)
0H, & ' Coord Y Min (High Byte)
080H, & ' Coord Y Max (Low Byte)
0H, & ' Coord Y Max (High Byte)
3, & ' Integration (3 = 8 Integration
' length)
0, & ' 0: normal mode, 1: combi-Modus
'with Analog-Driver
4 ' Delay in ms from setting the
'ctrl-pins and sampling the analog
'values
```

### Example 2: TEC-1000

This is the example of the Basic Tiger TEC-1000. It uses the XPORT address F8H. The analog Input pin is AN0. The control pins are connected:

|              |   |                                      |
|--------------|---|--------------------------------------|
| X left :     | 3 | (High active)                        |
| X right:     | 0 | (Low active)                         |
| Y top:       | 2 | (Low active)                         |
| Y bottom:    | 1 | (High active)                        |
| Multiplexer: | 5 | (Multiplexer Pin 4 is NOT necessary) |



to Touchpanel

Init of TOUCHPANEL.TDD with this circuit:

```

install_device #TP, "TOUCHPANEL.TDD", &
  0F8H, &           ' Port address
  0, &              ' use XPORT here
  00000000B, &     ' Default Pins of Port
  3, &              ' Bitnumber XL (X-Left)
  HIGH_ACTIVE, &   ' High Active
  0, &              ' Bitnumber XR (X-Right)
  LOW_ACTIVE, &    ' Low active
  2, &              ' Bitnumber YT (Y-Top)
  LOW_ACTIVE, &    ' Low active
  1, &              ' Bitnumber YB (Y-Bottom)
  HIGH_ACTIVE, &   ' High active
  5, &              ' Pin Nummer Multiplexer
  X_MUX_HIGH_ACTIVE, & ' 0: Xi = High Active
  0, &              ' Input Y (ADC)
  0, &              ' Input X (ADC)
  0A0H, &          ' AD-MIN X low Byte
  0H, &            ' AD-MIN X High Byte
  057H, &          ' AD-MAX X low Byte
  3H, &            ' AD-MAX X High Byte
  0FFH, &          ' AD-MIN Y low Byte
  0H, &            ' AD-MIN Y High Byte
  61H, &          ' AD-MAX Y low Byte
  3H, &            ' AD-MAX Y High Byte
  20, &            ' AD-PRESS low Byte
  0H, &            ' AD-PRESS High Byte
  0H, &            ' Coord X Min (Low Byte)
  0H, &            ' Coord X Min (High Byte)
  0F0H, &          ' Coord X Max (Low Byte)
  0H, &            ' Coord X Max (High Byte)
  0H, &            ' Coord Y Min (Low Byte)
  0H, &            ' Coord Y Min (High Byte)
  080H, &          ' Coord Y Max (Low Byte)
  0H, &            ' Coord Y Max (High Byte)
  3, &              ' Integration (3 = 8 Integration length)
  0, &              ' 0: normal mode, 1: combi-Modus with Analog-Driver
  4                 ' Delay in ms from setting the ctrl-pins and
                    ' sampling the analog values

```

**Example 3: TP-1000**

The standard settings of the device driver TOUCHPANEL.TDD runs with the TP-1000:



## Documentation History

| Version of Documentation | Version of driver | Description / Changes  |
|--------------------------|-------------------|--|
| 012                      | 1.00h             | New subroutines in TP_CALIBRATE.INC: <ul style="list-style-type: none"><li>- vCalibrateTouchpanelFlash</li><li>- vCalibrateTouchpanelFlashForce</li></ul>                        |
| 013                      | 1.00i             | <ul style="list-style-type: none"><li>- Look of documentation changed</li><li>- BEEP</li><li>- BEEP Key attribute</li><li>- Index added</li><li>- Slider activity flag</li></ul> |
| 014                      | 1.00j             | <ul style="list-style-type: none"><li>- Second alternative keyboard buffer</li></ul>   |
| 015                      | 1.00n             | <ul style="list-style-type: none"><li>- Pressed length</li></ul>   |
| 016                      | 1.00o             | <ul style="list-style-type: none"><li>- Examples</li></ul>   |
| 017                      | 1.01a             | <ul style="list-style-type: none"><li>- Beeper on XPORT possible</li></ul>   |