0 General

0.1 Abbreviations

KB Keyboard
SAA System Application Architecture

1 Overview

1.1 Summary

The keyboard controllers of TA85P and TA58P have a Flash EPROM and their firmware has a feature, that allows loading of keyboard tables. The controller firmware is not only able to produce the codes currently used, but also can produce other codes specified by software developers.

The historical evolution of the PC and the compatibility requirements are the reason, that understanding keyboard codes fully has become more and more complex, which often is underestimated. Have in mind, that compatibility to the keyboard drivers of PC operating systems is a requirement. So there is no freedom in a sense: "let's just assign this code to that key"!

Two different situations regarding support of programmable keyboards have to be distinguished:

- Development of the proper tables by software developers
- Loading the keyboard tables by technicians responsible for system installation or maintenance

The developer requires comfortable handling and much flexibility for preparation, maintenance, and documentation of the tables.

People involved in installation or maintenance want to load the keyboards in short time with the means they have in the rsp. configuration. It should be possible to switch back the keyboard into the (currently used) Wincor Nixdorf default behaviour. This leads to the requirement for different utilities adapted to the needs of that two situations.

2 Dialog Utility

2.1 Configuration

To prepare the keyboard tables the following configuration can be used:

- BEETLE system with screen or PC, operating system DOS, Windows 95, Windows 98, or Windows NT
- Target keyboard with wedge (target keyboard connected to primary interface of the wedge)
- Standard keyboard connected to secondary interface of the wedge

Note, however, that loading a table into a programmable keyboard only can be done under DOS, in DOS Mode or in a DOS window of Windows 9x!

We (i.e. Wincor Nixdorf Retail Systems) have the convention, that unsolicited commands to the keyboard only go to the primary interface. So the target keyboard has to be connected to the primary interface to allow to load it. Using the standard PC type keyboard connected to the secondary interface to operate the utility doesn't give problems.

External keyboard wedges may behave different. Some wedges direct (unsolicited) commands to that keyboard, where the last key has been pressed.

Important: ========== It is strongly recommended to use the internal keyboard wedge of the TA85P, TA58P keyboard rsp. SNIkey rather than an external wedge!!!

This is a general rule referring to the programming of a TA85P, TA58P or a SNIkey. However, many rules have an exception: For good reasons the TA59 and TA60 don't have a wedge. Therefore these two keyboards can only be programmed using an external wedge. Note, that TA59 or TA60 must not be connected to a TA85P or TA58P (or even a SNIkey) for programming! The tables rsp. firmware would be sent to the first keyboard, which would accidentally overwrite those of the TA85P or TA58P (or even the SNIkey). Use an external wedge instead, preferably one with a manual switch to ensure the data goes to the right target.

Important: ========== For programming a TA59 or TA60, it must not be connected to a TA85P, a TA58P or a SNIkey. Use an external wedge instead!!!

Take extreme care with configurations like the following:
- an external wedge
- a TA60 to be programmed and
- a TA59 for alphanumeric input

connected to the wedge, as e.g. the firmware could be erroneously sent to the TA59 rather than the TA60!!!

Ensure you understand fully the behaviour of your wedge!!!

In some DOS configurations keyboard tables are updated at every startup of the system. Observe the restriction above in such configurations!

2.2 User Interface

For good acceptance the dialog utility has a SAA user interface. A mouse is strongly recommended, but only required for editing.

The main menu can be selected by F10. Items in the main menu can be selected by Alt+letter, like e.g. Alt+F for the 'File' menu. Also navigating is possible with the cursor keys and then selecting by pressing CR (Carriage Return).

Context sensitive help can be obtained by pressing F1.

The language used in menus, dialogs, help texts, etc. is English.

When KBUTI.EXE is started, first the program version is displayed. Click on OK or press Carriage Return, which is equivalent.

2.3 Menus

Now the menus of the dialog utility KBUTI.EXE are described.

2.3.1 File Menu

It has the functions:

- 'New'
- 'Open'
- 'Save'
- 'Save as ...'
- 'Send KB table'
- 'Receive KB table'
- 'Reset KB to default'
- 'Check for KB default'
- 'Convert file to table
- 'Change dir...'
- 'DOS shell'
- 'Exit'

If you select the 'File' menu after KBUTI.EXE was started, you will see the functions 'Save', 'Save as ...', 'Send KB table' in gray color.

Generally menu items displayed in gray color can not be selected currently.

Use the function 'New', when new tables are to be built. When New was selected, first the three menu items of the 'Config' menu have to be chosen and the proper selections made, before editing can be performed. See the 'Config' menu below.

The function 'Open' allows to open an existing file containing a table for a TA85P or TA58P. The extension .KBT is used by convention for such type of files.

The function 'Save' can be used, if a filename is already known, otherwise 'Save as ...' should be used.

'Send KB table' sends the actual table to the keyboard. Information about intermediate steps is displayed and acknowledged by clicking on OK with the mouse or pressing Carriage Return.

'Receive KB table' reads a table from the keyboard. Accordingly information about intermediate steps is displayed and acknowledged by clicking on OK with the mouse or pressing Carriage Return.

'Reset KB to default' resets the keyboard to the default behaviour, i.e. the (standard) state when delivered by the factory, which is the behaviour of a (nonprogrammable) TA58 or TA58.

'Check for KB default' queries the keyboard, whether it is in default state or programmed, i.e. controlled by a table loaded earlier.

Important: During transmission the operator must not hit any key, change the
========== key position, swipe a card etc., since this would lead to a fault!

The 4 functions last mentioned are also available by batch utilities (see below). However, having those functions also in the dialog utility, provides for a more comfortable development of tables.

'Update KB firmware' allows an update of the keyboard firmware. This is introduced with TA59 and TA60 keyboards and therefore not available for TA85P and TA58P. Keyboard firmware files usually have a .FRM extension.

'Show KB firmware info' serves to get information about the actual firmware and loader version and their checksum. The function is introduced with TA59 and TA60 keyboards and is therefore not available for TA85P and TA58P.

'Convert file to table' is intended to translate from a list file into internal table format, which then can be saved in .KBT format. It can be used to produce the internal table format from a Preh .MWX format. This has, of course, some limitations and therefore can only be used to make porting easier. You find more details in section 3.

'Change dir ...' allows to change the actual directory.

'DOS shell' allows you to shell out to the DOS prompt to issue DOS commands. Don't forget to get back into the utility by issueing the DOS Exit command in order not to lose any work you have done before!

'Exit' ends the program and may ask, whether the actual table should be saved.

Important: For all functions that communinicate with the Programmable Device,
========== i.e. a Programmable keyboard, a Programmable SNIkey, or a
Programmable MSR, the Programmable Device has to be connected
directly to the system!

For all functions that communicate with the Programmable Device
in the BIOS Setup Legacy USB Support must be disabled!

2.3.2 Edit Menu

The 'Edit' Menu has the functions:

- Keyboard components
- Keyboard keys

In the functions 'Keyboard components' and 'Keyboard keys' most of the work will be done.

The function 'Keyboard components' displays a window like this:

```
    WPR  Windows Protocol (SN Retail) for components:
         Magnetic card reader, Key lock
    ------------------------------------------------
         Header (xxH) / Trailer (xxT) Specification
    Mag card reader |
    M1H track 1 M1T |-----------------|
    M2H track 2 M2T | Key lock        |
    M3H track 3 M3T | KLH key no  KLT |
```

Click on WPR to select the so-called (by SN Retail) Windows Protocol for magnetic card reader, key lock, which is default for actual POS keyboards delivered by SN Retail. This also provides for availability of the so-called (by SN Retail) DOS-Protocol, which is enabled by special commands to the keyboard, that are not discussed here.

Information on Windows Protocol definition may be obtained from SN Retail.

If you want to explicitly assign your own codes to one of the keyboard components, you have to define the codes for all used by the application. To simplify the task, Headers and Trailers not specified are considered empty by default. Editing is initiated by clicking on:

```
    M1H / M1T   Header / Trailer codes of magnetic card reader track 1
    M2H / M2T   Header / Trailer codes of magnetic card reader track 2
    M3H / M3T   Header / Trailer codes of magnetic card reader track 3

    KLH / KLT   Header / Trailer codes of key lock
```

You then get into a window, which serves to enter the proper codes in one input line. The codes can be specified either

- in ASCII as a string of characters,
- by symbolic keynames or even key expressions included within {}
- by arbitrary combinations of both
- or as an alternative in the form of hexadecimal 8042 scan codes byte per byte, each
  separated by spaces.

In the upper border line of the window the actual item to edit is shown.

Important: To edit codes in hex, a fairly good understanding of the structure
========== and knowledge of the 8042 scan codes is required!

For more details about symbolic key names please refer to the section Syntax of Key Codes.

Regarding the magnetic card reader, note that neither a start character nor an end character of a track is transmitted by default. So you may (and presumably have to) chose your own. A good choice for the end character could be "?", as this often is used as the translation of the end sentinel.

Example:

- tracks 1, 2, 3 shall be identified by preceding rsp. "1:", "2:", "3:"
- end of a track shall be identified by "?"
- after track 3 an additional Carriage Return shall be transmitted

You would specify then for


```
M1H         "1:"
M1T         "?"
M2H         "2:"
M2T         "?"
M3H         "3:"
M3T         "?{return}"
```

Magnetic card data itself is delivered in ASCII. For track 1 separators are delivered as "^" and for tracks 2 and 3 as "=". There is no special indicator for failures that may occur during read for a specific track. Instead empty data is delivered in such a case for that track. So in the example above:

```
"1:?2:..."
```

would be delivered, if track 1 was unreadable. Note, however, that if reading resulted in faults at all 3 tracks, it could well occur, that the application would not get any data at all. This behaves the same like in a case where e.g. a card has no magnetic stripe or was swiped such, that the stripe didn't touch the read head, etc. .

The key lock data consists of one ASCII digit "0",..,"5" for positions 0 to 5 or "t" for T ("technician's key") position.

To be able in an application to distinguish between a change of the key lock and entering the same codes at the keyboard, it is a must to assign at least Header codes.

The maximum number of codes for Header strings is 30 and for Trailer strings is 15. At the interface between keyboard and system make codes consist of 1 byte and break codes of 2 bytes. So a Header string may consist of up to 10 simple key presses and a trailer string of 5 simple key presses. Note, however, that there exist so-called extended keys, with 2 bytes make code and 3 bytes break code. Also, if large letters or special characters occor in the string, additional shift key presses are inserted. Note also, if specification is done like this:

```
{alt+#240},
```

the result is 4 key presses, i.e. 4 * 3 = 12 codes.

The function 'Keyboard codes' displays a window containing the key matrix according to the layout of the keyboard, i.e. for the TA85P:

```
G01  G02  G03  G04  G05  G06  G07  G08  G09  G10  G11  G12
F01  F02  F03  F04  F05  F06  F07  F08  F09  F10  F11  F12
E01  E02  E03  E04  E05  E06  E07  E08  E09  E10  E11  E12
D01  D02  D03  D04  D05  D06  D07  D08  D09  D10  D11  D12
C01  C02  C03  C04  C05  C06  C07  C08  C09  C10  C11  C12
B01  B02  B03  B04  B05  B06  B07  B08  B09  B10  B11  B12
A01  A02  A03  A04  A05  A06  A07  A08  A09  A10  A11  A12
```

It allows the selection of a key by a mouse click and the assignment of the rsp. keyboard codes.

There are (newer) versions of the SNIkey, that also are programmable. First versions are delivered in 2002. To allow for programming, at least Version 1.10 of the keyboard utilities is required. Note, that this applies not only to the dialog utlity KBUTI.EXE, but also especially to SENDKBT.EXE, RCVKBT.EXE.

The SNIkey has a 8 * 4 matrix keyboard and optional 8 softkeys:

```
S01       H01  H02  H03  H04
S02       G01  G02  G03  G04
S03       F01  F02  F03  F04
S04       E01  E02  E03  E04
S05       D01  D02  D03  D04
S06       C01  C02  C03  C04
S07       B01  B02  B03  B04
S08       A01  A02  A03  A04
```

The keys of the matrix keyboard are named A01...H04 and the softkeys S01...S08.

The TA60 is also a matrix keyboard. Its layout:

```
F01 F02 F03 F04 F05 F06 F07 F08 F09 F10
E01 E02 E03 E04 E05 E06 E07 E08 E09 E10
```



```
    D01 D02 D03 D04 D05 D06 D07 D08 D09 D10
    C01 C02 C03 C04 C05 C06 C07 C08 C09 C10
    B01 B02 B03 B04 B05 B06 B07 B08 B09 B10
    A01 A02 A03 A04 A05 A06 A07 A08 A09 A10
```

For the TA58 its layout is displayed accordingly. But as a matrix makes no sense, just key numbers are used instead:

```
    P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12
    P13 P14 P15 P16 P17 P18 P19 P20 P21 P22 P23 P24
    P25 P26 P27 P28 P29 P30 P31 P32 P33 P34 P35 P36

    Q01 Q02 Q03 Q04 Q05 Q06 Q07 Q08 Q09 Q10 Q11
    Q12 Q13 Q14 Q15 Q16 Q17 Q18 Q19 Q20 Q21 Q22     R01 R02 R03 R04
    Q23 Q24 Q25 Q26 Q27 Q28 Q29 Q30 Q31 Q32 Q33     R05 R06 R07 R08
      Q34 Q35 Q36 Q37 Q38 Q39 Q40 Q41 Q42 Q43       R09 R10 R11 R12
    Q44 Q45 Q46 Q47 Q48 Q49 Q50 Q51 Q52 Q53  Q54    R13 R14 R15 R16
    Q55 Q56 Q57 Q58 Q59 Q60 Q61 Q62 Q63 Q64 Q65     R17 R18 R19 R20
```

The TA59 is also an alphanumeric keyboard. Its layout (nonprogrammable keys omitted):

```
    P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15  R01 R02 R03 R04
    P16 P17 P18 P19 P20 P21 P22 P23 P24 P25 P26 P27 P28 P29 P30  R05 R06 R07 R08

                                                                 R09 R10 R11 R12 R13
R14 R15 R16
                                                                 R17 R18 R19 R20 R21
R22 R23 R24
                                                                 R25 R26 R27 R28 R29
R30 R31 R32
                                                                 R33 R34 R35 R36 R37
R38 R39 R40
    Q01  Q02  Q03                          Q04  Q05  Q06  Q07   R41 R42 R43 R44 R45
R46 R47 R48
```

For space restrictions the right part of the keyboard layout of the TA59 is displayed below the left part. Note, that not all keys are programmable. The programmable keys are:

- function keys P01,...,P30
- special keys and numeric part R01,...,R48
- Left Ctrl key Q01, Left Windows key Q02, Left Alt key Q03, Right Alt (rsp. AltGr key) Q04, Right Windows key Q05, Windows Context Menu key Q06, Right Ctrl key Q07

After a key has been clicked, the Edit Key dialog is entered. In the upper border line of the window the rsp. name/number of the actual key to edit is shown. This dialog offers input fields for up to 4 levels to specify the codes either as

- an ASCII string of lower/upper case letters, digits, or special characters
- by symbolic key names or even expressions of symbolic key names included in {}
- arbitrary combinations of both ASCII strings and expressions enclosed in {}
- alternatively in the form of hexadecimal 8042 scan codes specified byte per byte, each separated by spaces.

Use {Space} to specify a space as the only code or at the end of a key string.

Important: To edit codes in hex, a fairly good understanding of the structure
========== and detailed information of the 8042 scan codes is required!

For technical reasons, input fields for all 4 levels are displayed. However, input fields of levels not available are labelled "Not available!"

You may want to override default Edit mode and do your editing in Hex rather than in ASCII or vice versa. So you can change it for a level just by clicking on the proper "Radio button".

Also the default Repeat mode behaviour for a level of a key may be overridden. By clicking on the "Check box" labelled Autorepeat you may switch it on or off. Note, however, that the keyboard distinguishes between several types of keys. Keys can be:

- strings of codes, e.g. {ctrl+f5}, {alt+#240}, "A" (Autorepeat not possible)
- double keys, e.g. {ctrl+alt} (Autorepeat not possible)
- single keys, e.g. "a", {f10}, {alt} (Autorepeat possible)
- special level keys, e.g. POS Shift level x (Autorepeat makes no sense)

You may select Autorepeat for all of them. However, you will find that KBUTI.EXE automatically detects single keys and only allows Autorepeat for them.

You should know, that specification of key codes in the form of {alt+#nnn} requires an operating system like MS-DOS, Windows 3.x, Windows 9x or Windows NT! Keyboard drivers of other operating systems, let's say UNIX or LINUX, to our knowledge don't do such processing! This is important to know especially for the magnetic card reader, if keyboard languages other than US are used. The language tables used in the keyboard for non-US countries use the {alt+#nnn} mechanism at least for some codes.

Important: ========== For Non-Microsoft operating systems don't use the {alt+#nnn} mechanism!!! In such a case it is strongly recommendable to use the US keyboard driver on the target system, or at least the Windows Mode for MSR, Key Lock!!!

Check the codes generated by using one of the functions in the 'Print' menu!!!

You may have configured the keyboard for:

- Normal + POS Shift levels
- Normal + Ctrl / Alt / AltGr levels

in the 'Config' menu (see below). In such a case you may want to define the actual key as a means to switch to another level different from the Normal. Look at the 4 Radio buttons labelled Std, L1, L2, L3. By default, Std will be selected. Simply click on the level you want the key to switch to (L1, L2, or L3). That's all for this key. If a code has to be assigned to this key, let this be {Ctrl}, {Alt}, or {AltGr}, it will be done automatically. You may convince you by clicking on OK and then select the same key again.

The maximum number of codes for a key is 60. At the interface between keyboard and system make codes consist of 1 byte and break codes of 2 bytes. So a string may contain of up to 20 simple key presses. Note, however, that there exist so-called extended keys, with 2 bytes make code and 3 bytes break code. If large letters or special characters occur in the string, additional shift key presses are inserted. Note also, if specification is done like:

```
{alt+#240},
```

the result is 4 key presses, i.e. 4 * 3 = 12 codes.

When you have finished editing of codes for a key, click on OK. If you were not able to finish all the work for this key now, you can proceed with that later. Don't forget, however, to save your work before leaving the program.

Also you may wish to look at the key assignments without doing any changes. In such cases you would click on Cancel or on the square in the top left corner of the window, which is equivalent.

It may occur, that you specified some language specific character, which has a code above 7F (hex), like e.g. a German A-Umlaut, which has a code of 8E (hex). When you again look at the key code, you will see it in the form:

```
{alt+#xyz},
```

which in case of code 8E (hex) would be actually:

```
{alt+#142}.
```

It is done this way to avoid dependency of specific code pages. Note, that for a country there may be more than one possible code page. Also pitfalls occurring with dead keys (sometimes also known under the term diacritical characters) are avoided. As a positive effect, it also reduces complexity in building country specific tables.

On the bottom left side of the Key Edit window you find a checkbox labelled 'Next key'. If you mark this checkbox and leave the dialog window, you will automatically get the Key Edit window for the next key (in lexical order). This happens as long as that checkbox is marked. You may use this feature also for viewing the key codes assigned key by key, leaving the window by pressing Cancel.

2.3.3 Config Menu

The 'Config' Menu has the functions:

- 'Levels'
- 'Keyboard language'
- 'Keyboard type'

By chosing 'Levels' you can either select

- Normal level (standard KB handling)
- Normal + POS Shift levels
- Normal + Ctrl / Alt / AltGr levels
- Normal + CapsLock / ScrollLock levels

By the first selection you have only one level, but on the other hand it is the easiest to implement for the programmer and to handle by the operator.

Normal level (standard KB handling) behaves the usual way, i.e. there are no keys that can be used to modify the codes issued by the keyboard. Note, that e.g. output of capital letters instead of small letters, when Shift together with an alpha key is pressed, is a task of the operating system's keyboard driver rather than the keyboard. Normal level in the following is implicitly regarded as level 0.

Selecting 'Normal + POS Shift levels' allows you to assign up to three shift keys (or in other words level keys). By just pressing one of these POS Shift keys no code is issued at all. Instead the appropriate level (i.e. 1, 2, or 3) is selected by the keyboard firmware, as long as one of them is pressed. So, if e.g. the POS Shift key for level 1 is hold and then, let's say, key G01 is pressed, the keyboard would issue the codes assigned to level 1 of key G01, etc. .

Selecting 'Normal + Ctrl / Alt / AltGr levels' allows you to use the Ctrl key, the Alt key, the AltGr key to select 3 additional levels in much the same way. So if Ctrl is hold and then, let's say, key G01 is pressed, the keyboard would issue the codes assigned to level 1 of key G01. Similarly, if Alt is hold and then, let's say, key G01 is pressed, the keyboard would issue the codes assigned to level 2 of key G01, and if AltGr is hold, then G01 pressed, those assigned to level 3.

Primarily, the 'Normal + Ctrl / Alt / AltGr level' is intended for an alphanumeric keyboard like TA58P, which by default has Ctrl, Alt, AltGr keys. However, it can be used with TA85P also, provided the Ctrl, Alt, AltGr keys or a proper subset of them according to the requirements are assigned (see the 'Edit' menu on how to assign those codes easily).

The 'Normal + CapsLock / ScrollLock levels' selection allows you to have 2 levels additionally. From the keyboard point of view these levels are selected by the state of the CapsLock LED and the ScrollLock LED respectively.

Note, however, that there are several restrictions! A means to influence the state of the LEDs from an application point of view normally only exists under DOS, in DOS Mode of Windows 95 or 98 or in a DOS Window of the latter two. For Win32 applications or in a DOS Window of Windows NT the state of the LEDs is impossible to change. What could be possible in such an environment just from a technical point of view, is to assign ScrollLock to one key and use it as a toggle key between Normal and ScrollLock keyboard levels. The ScrollLock LED then would indicate the toggle state. But not everything technically possible may also be good from an ergonomic point of view.

There is also another restriction concerning CapsLock level. In many cases this requires the use of either the US keyboard driver or the KEYBRD2.SYS driver. The reason is, that normally one would like to have the numeric keys in both the Normal and in the CapsLock level. KEYBOARD.SYS driver for some languages, like e.g. German, has ShiftLock rather than CapsLock behaviour, which would lead to special characters rather than decimal digits in CapsLock state! An idea to circumvent that problem could be to use the keys of the numeric block rather than those of the alphanumeric, however, there codes are dependent of the proper NumLock state!

Take the above as an indication, that you get much flexibility by programmable keyboards, but also there are a number of pitfalls, which may require a lot of time for testing or experience to circumvent!

The 'Keyboard language' selection of the 'Config' menu is important to have the proper mapping of ASCII characters to keyboard codes and vice versa and for the translation of magnetic card reader data as well.

The 'Keyboard type' selection of the 'Config' menu is required for the utility to know key names and to display the proper keyboard layout. It must not be

changed later!

2.3.4 Preferences Menu

The 'Preferences' menu allows you to select and switch settings during the editing process. They remain valid as defaults until they are changed. So you may change them at any time during editing codes of keys or keyboard components. In the process of editing you can override the preferences easily by making a different choice.

The selection made in function 'Edit mode' is used to switch the default for specifying codes as ASCII strings or a sequence of hexadecimal 8042 scan codes. The design of the utility has been done so that the latter is required rarely.

It may be useful for countries, which keyboard languages are not in the list, especially for Asian countries. Note, however, that specification of hexadecimal 8042 scan codes requires a deep knowledge! Note also, that codes of the magnetic card reader also are an issue, if your country is not in the list of keyboard languages supported!

In the 'Repeat mode' function you select as default whether you want to have Autorepeat for a key or not. In the process of editing you can override this selection easily by making a different choice.

2.3.5 Print Menu

In the 'Print' Menu you chose whether you want to print:

- 'to File'
- 'to LPT1'
- 'to display'

This functions are intended for documentation purposes. They write the utility internal representation in readable form into a file, to a printer or for viewing to the screen.

The format used is similar to that of Preh's utility to deal with .MWX files.

A comment line starting with '!' indicates the file path, if available.
A 'TYPE' line specifies keyboard type (TA85P or TA58P) and a 'LANGUAGE' line the keyboard language.

For a key or component you find the following information:

- key rsp. component id, e.g. D08
- level conditions, e.g. +C or -C
- attributes, e.g. /A
- code(s) enclosed in quotes

Level conditions:

+	active
-	inactive
C	Ctrl
A	Alt
AG	AltGr
CL	CapsLock
SL	ScrollLock
L1	PosShift level 1
L2	PosShift level 2
L3	PosShift level 3

Attributes:

/A	Autorepeat
/H	Editing in Hex
/Z	WKL Remove delivers '000'

Note, that a temporary file in the actual directory is written in the function 'Print to display'. Therefore writing to the actual directory has to be allowed.

2.3.6 About Menu

The 'About' Menu displays company, copyright and the version of the utility. This dialog is the same as the one you see at start. Have the version number at hand, when you require support.

2.3.7 Syntax Of Key Codes

For easy reference the key symbols used are the same as with Preh's MWX keyboard series (mostly de-facto standard anyway):

F1...F12	Function keys F1 ... F12
FS1...FS12	Shift + Function keys F1...F12
FC1...FC12	Ctrl + Function keys F1...F12
FA1...FA12	Alt + Function keys F1...F12
N0...N9	Numeric keys in numeric block
SPACE	Space key
BACKSPACE	Backspace
BS	Backspace alternatively
PAUSE	Pause
BREAK	Break
CAPS	CapsLock
INS	Insert
DEL	Delete
PGUP	Page Up
PGDN	Page Down
HOME	Home
END	End
SYS	SysReq
NUM	NumLock
SCROLL	ScrollLock
PRTSC	Print screen
ESC	Escape
TAB	Tab
LEFT	Cursor Left
RIGHT	Cursor Right
UP	Cursor Up
DOWN	Cursor Down
RETURN	Return
ENTER	Enter
PLUS	Plus (numeric block)
MINUS	Minus (numeric block)
STAR	* (numeric block)
DIV	/ (numeric block
SHIFT	Shift
CTRL	Ctrl
ALT	Alt
ALTGR	AltGr
LWIN	Left Windows key
RWIN	Right Windows key
APP	Windows Context Menu key

LWIN and RWIN can be used like shift keys, e.g. {LWIN+e} to open the Windows Explorer window.

You could have the idea to set the language to US and use hexadecimal scan codes to be more language independent under certain conditions. Note, that the use of key symbols is preferable in such a case, whereever available. The reason is that some keys as e.g. the cursor keys, INS, HOME, PGUP, PGDN, DEL, END, PGDN issue different scan codes depending on the NumLock state, which behaviour would be lost if programmed by hex scan codes.

Some of the key symbols should be used restrictively. Hints to such special key symbols:

SYS	would generate the 8042 scan codes 54(Make) D4(Break). However, on PC/AT type keyboards there is no such key. It may be useful in the conjunction {Alt+Sys}.
PRTSC	generates 8042 scan codes [E0 2A] E0 37 E0 B7 [E0 2A]. Can be used in conjunctions {Shift+Prtsc}, {Ctrl+Prtsc}. Do not(!) use it with ALT, use {Alt+Sys} instead!
BREAK	generates 8042 scan codes 1D E0 46 E0 C6 9D. So in fact it already has the same effect as {Ctrl+Break} would have! Do not(!) use {Ctrl+Pause}, use {Break} instead!

In addition to those well-known key names some more are implemented to provide for a clean and consistent syntax in the various keyboard languages. The intention is to allow a language independent specification for PC type keyboard keys that may deliver special characters . To refer to them independently of a specific keyboard language, the relevant alpha part of the US PC type keyboard layout is used. The lower case characters are also given for easier reference. Note, that the keys named Z42 and Z45 are only available with 102 keys MF2 keyboards, i.e. not available with the 101 keys MF2 of US type. This, however, has a key named Z29, which is not available with MF2 keyboards with 102 keys.


```
    `    1    2    3    4    5    6    7    8    9    0    -    =     BS
   Z1   xxx  xxx  xxx  xxx  xxx  xxx  xxx  xxx  xxx  xxx  Z12  Z13   xxx

          q    w    e    r    t    y    u    i    o    p    [    ]    \
         xxx  xxx  xxx  xxx  xxx  xxx  xxx  xxx  xxx  xxx  Z27  Z28  Z29

             a    s    d    f    g    h    j    k    l    ;    '        Return
            xxx  xxx  xxx  xxx  xxx  xxx  xxx  xxx  xxx  Z40  Z41  Z42   xxx

               z    x    c    v    b    n    m    ,    .    /
          Z45  xxx  xxx  xxx  xxx  xxx  xxx  Z52  Z53  Z54  Z55  xxx
```

So the key names provided additionally are:

- Z1, Z12, Z13
- Z27..Z29
- Z40..Z42
- Z45
- Z52..Z55

Key names have to be enclosed in {} (one level of bracketing only) to be recognized. The names of the shift keys SHIFT, CTRL, ALT, ALTGR, even may be combined with key names or a single decimal digit using the + operator to form key expressions. ALT may also be combined with a 3 digit string of the form #nnn.

It makes no sense if letters of key names are specified in upper or lower case.

Examples of key expressions and allowed concatenations in key strings are:

```
{Ctrl}
{Ctrl+Alt}
{Shift+F4}
{Ctrl+x}{Ctrl+y}{Alt+#240}
Key{Alt+#123}
{Ctrl+u}
{Alt+Z12}
{Ctrl+4}
```

Note, that

```
{Alt+N1} is equivalent to {Alt+#1}
         and not (!!!) to {Alt+1}
```

Take care to distinguish clearly between digits of the alphanumeric and the numeric block, so:

```
{Alt+1} is different than {Alt+#1}

{Ctrl+#4} is equivalent to {Ctrl+N4}
{Shift+4} is equivalent to the '$' sign, if US keyboard language is used
```

The French and Belgian keyboards have the digits in the shifted, rather than in unshifted (Normal) level. However, for easy handling and a clean syntax, you also specify in this case:

```
{Alt+4}
{Ctrl+8}
...
```

It may be recommendable to enclose spaces in {}, although not required:

```
{Space}
```

3.2 Converting Codes from Existing Keyboard into .KBT format

'Convert file to table' serves to translate from a list file into internal table format, which then can be saved in .KBT format. Before 'Convert file to table' can be used, the function 'Levels' of the 'Config' menu first has to be used to select the levels intended. 'TYPE' and 'LANGUAGE' lines are handled by the utility. So the selections of the functions 'Keyboard type' and 'Keyboard language' are not relevant, instead they are overwritten.

The criteria and attributes supported are:

Level criteria:


```
 +   active
 -   inactive
 C   Ctrl
 A   Alt
 AG  AltGr
 CL  CapsLock
 SL  ScrollLock
 L1  PosShift level 1
 L2  PosShift level 2
 L3  PosShift level 3

Attributes:

 /A   Autorepeat
 /H   Editing in Hex
 /Z   WKL Remove delivers '000'
```

To use this function to convert from Preh's .LST format, the following has been found important to observe:

- Preh allows special characters like German ae (A-Umlaut) and some special characters to specify keys within {}. This is not allowed by KBUTI.EXE to avoid problems (e.g. in syntax checking, with deadkeys etc.) with the various keyboard languages. You can specify the equivalent by e.g.

  {Alt+Z41} rather than {Alt+ae} ae stands for a-Umlaut

- Pay attention to possible differences in the criteria for implementing levels, e.g. KBUTI.EXE doesn't allow {Shift} or {Num} for level selection.

- There is no {Make...} keyword.

- In the special case of CapsLock the Preh .LST file assigns codes to decimal digits in their shifted state, e.g. '%' for key '5' for the US keyboard language. KBUTI.EXE, however, expects and documents their true resulting code '5' also in the CapsLock level.

Most characters having codes of above 0x7F are implemented by Preh the same way, i.e. by a combination like {Alt+#nnn}, but are only displayed rsp. written to a .LST file differently.

The criteria for level specification and the key attributes ignored by the function 'Convert file to table' are:

```
Criteria:

 Shift  S
 Num    N

Attributes:

 Keyclick     K
 Slow output  L
 Restore      R
 No MakeBreak M (???)
```

Important: The function 'Convert file to table' is a kind of bridgeware. It can be used to make conversion of a Preh .MWX file easier. In simple cases it may do the whole task. However, it is strongly required to check the result and test it carefully!!!

4 Batch Utilities

4.1 Utilities For Use In The Field

There are utilities to:

- Send a keyboard table to the keyboard (SENDKBT.EXE)
- Receive a keyboard table from the keyboard (RCVKBT.EXE)
- Reset a keyboard to the appropriate default behaviour, i.e. state when delivered by factory (KB2DEF.EXE)
- Check whether keyboard is in default behaviour (KBCHKDEF.EXE)

Syntax and returned errorlevel:

```
        sendkbt <filepath>      errorlevel = 2  File open error or bad format
                                           = 1  Keyboard error
                                           = 0  OK

        rcvkbt  <filepath>      errorlevel = 2  File open error
                                           = 1  Keyboard error
                                           = 0  OK

        kbchkdef                errorlevel = 2  Keyboard error
                                           = 1  Keyboard programmed
                                           = 0  Keyboard in default state

        kb2def                  errorlevel = 1  Keyboad error
                                           = 0  OK
```

Example check for table activated:

```
    @echo off
    kbchkdef
    if errorlevel 2 goto :error
    if errorlevel 1 goto :programmed
    :default
    rem Keyboard has default behaviour
    echo Keyboard in default state
    goto :exit

    :programmed
    rem Keyboard has behaviour controlled by keyboard table
    echo Keyboard is programmed
    goto :exit

    :error
    rem Keyboard error occurred or keyboard is not programmable
    echo Keyboard error

    :exit
    echo on
```

Important: During transmission the operator must not hit any key, change the
========== key position, swipe a card etc., since this would lead to a fault!

To execute these utilities when using numeric keyboards, it makes sense to have proper .BAT files (TA85P).

It makes sense to deposit at the customers site a diskette with the utility SENDKBT.EXE and the file containing the customer specific keyboard table. This avoids the need for a sophisticated logistics, which wouldn't work 100% well anyway from our experience. A technician of a reseller or a software house could for security take with him a diskette with the utilities RCVKBT.EXE and SENDKBT.EXE, to possibly read out the keyboard table from a working keyboard installed at the customers site, in case the diskette with the keyboard table file was lost.

For Wincor Nixdorf personnel e.g. the following situation could occur: He took with him a keyboard for exchange at the customers site. Unfortunately that keyboard was loaded with a keyboard table different from default behaviour. In such a case a bootable diskette with the utility KB2DEF.EXE is useful, to reset the keyboard to the default behaviour. For DOS systems it is possible to query e.g. in AUTOEXEC.BAT, whether the keyboard has default behaviour or is controlled by keyboard table. So the returned ERRORLEVEL can be used to either reset it to default or load the table, whatever is needed.

It should be noted here, that the dialog utility also provides the functionality of the batch utilities (see Menu 'File').

5 Executability

All utilities have to communicate with the keyboard. For that purpose they have to circumvent to some extent the keyboard driver of the operating system, since the standard driver doesn't know the rsp. commands. The utilities must not confuse the standard keyboard driver in doing his task.

For technical reasons this can only be done under DOS or in either DOS mode or a DOS window of Windows 9x. Windows NT, 2000, XP don't allow access to the keyboard interface at all!

Important: For installations with Windows NT, 2000, XP an installation or a
========== replacement of keyboards can only take place with keyboards, that

were loaded previously with the proper tables!

For all functions that communinicate with the Programmable Device, i.e. a Programmable keyboard, a Programmable SNIkey, or a Programmable MSR, the Programmable Device has to be connected directly to the system!

For all functions that communicate with the Programmable Device in the BIOS Setup Legacy USB Support must be disabled!

6 Storage

The utilities are stored on one diskette. The diskette is delivered with a programmable keyboard. For the TA58P on the diskette a file is delivered, that implements the default behaviour of the keyboard by proper tables. This makes sense e.g. when compared to the default behaviour only a few codes are to be modified.

The diskette then contains the following files:

KBUTI.EXE	Dialog utility
KBUTIHLP.HLP	Help file for the dialog utility
SENDKBT.EXE	Send a keyboard table
RCVKBT.EXE	Receive a keyboard table
KB2DEF.EXE	Reset keyboard to default behaviour
KBCHKDEF.EXE	Check whether keyboard has default behaviour
TA58DEF.KBT	Table with default codes for TA58
TA59DEF.KBT	Table with default codes for TA59
TA60DEF.KBT	Table with default codes for TA60
TA85DEF.KBT	Table with default codes for TA85
SNIKDEF.KBT	Table with default codes for SNIkey
README.TXT	(This) Readme file

7 Changes And Bug Fixes

Rel. 1.24
- Some handshake introduced to provide for sufficient waiting time for newer firmware, i.e. the boards developed for RoHS compliance (with backward compatibility)

Rel. 1.23
- Waiting time made longer after sending country table (for Programmable MSR)

Rel. 1.22
- Support for MSR only added

Rel. 1.21:
- Waiting time introduced before sending last record (for Programmable MSR)

Rel. 1.20:
- Support for TA59 and TA60 added.
- Support for Firmware Upgrade and Firmware Module Info added (applies to TA59, TA60).
- Support for LWIN, RWIN, APP keys added.

Rel. 1.11:

- Scenario: A matrix keyboard (TA85P, SNIkey) was programmed for 'Normal level' and and PC type keyboard was connected as secondary keyboard to the matrix keyboard. When the PC type keyboard was in CapsLock state, pressing a key on the matrix keyboard had no effect. Also change of key switch or reading a card could switch from CapsLock ON to CapsLock OFF.
  Important: To fix this, KBUTI.EXE and(!) SENDKBT.EXE of Rel. 1.11 have to
  ========== be used!

- TA85DEF.KBT changed to KB language US instead of GR

Rel. 1.10:

- Support for programmable SNIkey added, i.e. matrix KB and softkeys. This feature requires a special SNIkey KB controller. That controller is delivered with the SNIkey 12.1 Inch A, i.e. 3 conditions must be fulfilled:
  . SNIkey
  . 12.1 Inch
  . PanelLink interface

8 Known Bugs

- If input to a dialog field is entered extremely fast, e.g. by randomly keying on the keyboard, a crash may be the result. This, however, doesn't occur in normal operation

