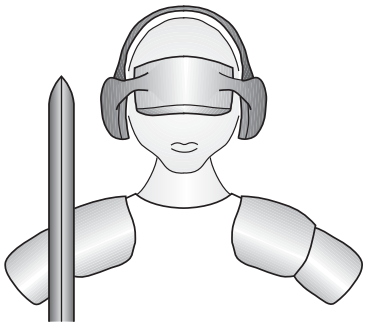
KeyWarrior II

Key Tables



Code Mercenaries

1. Introduction

Keyboards are usually organized as matrixes of switches. The placement of the keys in the matrix is mostly arbitrary with just a few constraints to keep keys fron interfering with each other.

The keyboard controller or encoder does read the matrix and translates the matrix coordinates into the proper usage codes. Usage codes are not ASCII characters but merely a standardized code for each key, telling the system that a certain key was pressed. It is then up to the keyboard driver to convert this information into actual characters.

KeyWarrior allows a physical key to have any function. Each individual key can be programmed to be a simple key, a macro, a mouse function or other things.

2. Using a KeyWarrior II chip

KeyWarrior II chips internally hold the table to convert the physical keys into keycodes and other functions. While the KeyWarrior II chips come with a default keytable factory programmed usually the manfacturer of the keyboard will replace this standard layout by a custom layout to fits the specific keyboard design.

A factory standard layout is primarily for testing to assure that the KeyWarrior II is generating something (if it supplied at all, not all variants do have a standard layout preloaded). It can be erased to write a custom layout into the chip. There is no option to restore the factory standard layout from within the chip. If you want to restore the default layout it must be done with the standard configuration files provided on our website.

Building the keytable can be done using our standard tools or may be build manually or with custom tools using the information contained in this document.

There is an editor tool for building the configuration data and a production tool that uses the files generated with the editor tool to mass program KeyWarrior II based devices via USB.

3. Building a Keyboard Table

The keyboard table consists of three tables with a 16 bit entry for each key and a number of 32 byte blocks for macros. The three tables are used depending on the FN-key status. The first table is used in the basic mode, the second if FN1 is active and the third when FN2 is active. Both FN keys active at the same time result in the third table to be used (FN2 takes priority).

For KeyWarrior28 each key table is 128 bytes long (2 bytes for each key) and there are 19 macros.

KeyWarrior100 has a much bigger key matrix, resulting in 786 bytes for each of the three key tables plus 176 macros. The 32 user macros are stored in a separate data structure.

Functions for keys are grouped into function pages. The 16 bit entries for each key define the function page with the high byte and the individual function with the low byte (low byte comes first in memory).

The following function pages are currently defined (any other codes will be ignored):

$00 - Standard keys, modifiers and FN keys

$01 - Media and application controls

$02 - Macros

$03 - Mouse

$04 - LED output control (KW100 only)

$05 - User Macros (KW100 only)

$E0…$E7 - Modfier plus standard key combo

4. Page $00 - Standard keys, modifiers and FN keys

Following is the table of the codes for normal keys, modifiers and FN keys, which are all on page $00.

Code	Function	AT-101 Ref.
$0000	Reserved - KeyWarrior sends no code	N/A
$0001	Reserved	N/A
$0002	Reserved	N/A
$0003	Reserved	N/A
$0004	Keyboard a and A	31
$0005	Keyboard b and B	50
$0006	Keyboard c and C	48
$0007	Keyboard d and D	33
$0008	Keyboard e and E	19
$0009	Keyboard f and F	34
$000A	Keyboard g and G	35
$000B	Keyboard h and H	36
$000C	Keyboard i and I	24
$000D	Keyboard j and J	37
$000E	Keyboard k and K	38
$000F	Keyboard l and L	39
$0010	Keyboard m and M	52
$0011	Keyboard n and N	51
$0012	Keyboard o and O	25
$0013	Keyboard p and P	26
$0014	Keyboard q and Q	17
$0015	Keyboard r and R	20
$0016	Keyboard s and S	32
$0017	Keyboard t and T	21
$0018	Keyboard u and U	23
$0019	Keyboard v and V	49
$001A	Keyboard w and W	18
$001B	Keyboard x and X	47
$001C	Keyboard y and Y German: z and Z	22
$001D	Keyboard z and Z German: y and Y	46
$001E	Keyboard 1 and !	2
$001F	Keyboard 2 and @ German: 2 and "	3
$0020	Keyboard 3 and # German: 3 and §	4
$0021	Keyboard 4 and $	5
$0022	Keyboard 5 and %	6
$0023	Keyboard 6 and ^ German: 6 and &	7
$0024	Keyboard 7 and & German: 7 and /	8
$0025	Keyboard 8 and * German: 8 and (	9
$0026	Keyboard 9 and (German: 9 and)	10
$0027	Keyboard 0 and) German: 0 and =	11
$0028	Keyboard Return	43
$0029	Keyboard ESCAPE	110
$002A	Keyboard Backspace	15
$002B	Keyboard Tab	16
$002C	Keyboard SPACE	61
$002D	Keyboard - and _ German: ß and ?	12
$002E	Keyboard = and + German: ´ and `	13
$002F	Keyboard [and { German: ü and Ü	27
$0030	Keyboard] and } German: + and *	28
$0031	Keyboard \ and | German: # and '	29
$0032	Keyboard (non US) # and ~	42

Code	Function	AT-101 Ref.
$0033	Keyboard ; and : German: ö and Ö	40
$0034	Keyboard ‘ and “ German: ä and Ä	41
$0035	Keyboard ` and ~ German: ^ and °	1
$0036	Keyboard , and < German: , and ;	53
$0037	Keyboard . and > German: . and :	54
$0038	Keyboard / and ? German: - and _	55
$0039	Keyboard Caps Lock (not physically locking)	30
$003A	Keyboard F1	112
$003B	Keyboard F2	113
$003C	Keyboard F3	114
$003D	Keyboard F4	115
$003E	Keyboard F5	116
$003F	Keyboard F6	117
$0040	Keyboard F7	118
$0041	Keyboard F8	119
$0042	Keyboard F9	120
$0043	Keyboard F10	121
$0044	Keyboard F11	122
$0045	Keyboard F12	123
$0046	Keyboard Print Screen (Mac: F13)	124
$0047	Keyboard Scroll Lock (Mac: F14)	125
$0048	Keyboard Pause (Mac: F15)	126
$0049	Keyboard Insert (Mac: Help)	75
$004A	Keyboard Home	80
$004B	Keyboard PageUp	85
$004C	Keyboard Delete Forward	76
$004D	Keyboard End	81
$004E	Keyboard PageDown	86
$004F	Keyboard RightArrow	89
$0050	Keyboard LeftArrow	79
$0051	Keyboard DownArrow	84
$0052	Keyboard UpArrow	83
$0053	Keypad NumLock and Clear	90
$0054	Keypad /	95
$0055	Keypad *	100
$0056	Keypad -	105
$0057	Keypad +	106
$0058	Keypad ENTER	108
$0059	Keypad 1 and End	93
$005A	Keypad 2 and DownArrow	98
$005B	Keypad 3 and PageDown	103
$005C	Keypad 4 and LeftArrow	92
$005D	Keypad 5	97
$005E	Keypad 6 and RightArrow	102
$005F	Keypad 7 and Home	91
$0060	Keypad 8 and UpArrow	96
$0061	Keypad 9 and PageUp	101
$0062	Keypad 0 and Insert	99
$0063	Keypad . and Delete	104
$0064	Keyboard non US \ and | German: < and >	45
$0065	Keyboard Application (Menu)	129

Code	Function	AT-101 Ref.
$0066	Keyboard Power	N/A
$0067	Keypad =	N/A
$0068	Keyboard F13	N/A
$0069	Keyboard F14	N/A
$006A	Keyboard F15	N/A
$006B	Keyboard F16	N/A
$006C	Keyboard F17	N/A
$006D	Keyboard F18	N/A
$006E	Keyboard F19	N/A
$006F	Keyboard F20	N/A
$0070	Keyboard F21	N/A
$0071	Keyboard F22	N/A
$0072	Keyboard F23	N/A
$0073	Keyboard F24	N/A
$0074	Keyboard Execute	N/A
$0075	Keyboard Help	N/A
$0076	Keyboard Menu	N/A
$0077	Keyboard Select	N/A
$0078	Keyboard Stop	N/A
$0079	Keyboard Again	N/A
$007A	Keyboard Undo	N/A
$007B	Keyboard Cut	N/A
$007C	Keyboard Copy	N/A
$007D	Keyboard Paste	N/A
$007E	Keyboard Find	N/A
$007F	Keyboard Mute	N/A
$0080	Keyboard Volume Up	N/A
$0081	Keyboard Volume Down	N/A
$0082	Keyboard Locking Caps Lock	N/A
$0083	Keyboard Locking Num Lock	N/A
$0084	Keyboard Locking Scroll Lock	N/A
$0085	Keypad Comma	N/A
$0086	Keypad Equal Sign	N/A
$0087	Keyboard Kanji1	N/A
$0088	Keyboard Kanji2	N/A
$0089	Keyboard Kanji3	N/A
$008A	Keyboard Kanji4	N/A
$008B	Keyboard Kanji5	N/A
$008C	Keyboard Kanji6	N/A
$008D	Keyboard Kanji7	N/A
$008E	Keyboard Kanji8	N/A
$008F	Keyboard Kanji9	N/A
$0090	Keyboard LANG1	N/A
$0091	Keyboard LANG2	N/A
$0092	Keyboard LANG3	N/A
$0093	Keyboard LANG4	N/A
$0094	Keyboard LANG5	N/A
$0095	Keyboard LANG6	N/A
$0096	Keyboard LANG7	N/A
$0097	Keyboard LANG8	N/A
$0098	Keyboard LANG9	N/A

Code	Function	AT-101 Ref.
$0099	Keyboard Alternate Erase	N/A
$009A	Keyboard SysReq/Attention	N/A
$009B	Keyboard Cancel	N/A
$009C	Keyboard Clear	N/A
$009D	Keyboard Prior	N/A
$009E	Keyboard Return	N/A
$009F	Keyboard Separator	N/A
$00A0	Keyboard Out	N/A
$00A1	Keyboard Oper	N/A
$00A2	Keyboard Clear/Again	N/A
$00A3	Keyboard CrSel/Props	N/A
$00A4	Keyboard ExSel	N/A
$00A5-00AF	Reserved	N/A
$00B0	Keypad 00	N/A
$00B1	Keypad 000	N/A
$00B2	Thousands Separator	N/A
$00B3	Decimal Separator	N/A
$00B4	Currency Unit	N/A
$00B5	Curreny Sub-unit	N/A
$00B6	Keypad (	N/A
$00B7	Keypad)	N/A
$00B8	Keypad {	N/A
$00B9	Keypad }	N/A
$00BA	Keypad Tab	N/A
$00BB	Keypad Backspace	N/A
$00BC	Keypad A	N/A
$00BD	Keypad B	N/A
$00BE	Keypad C	N/A
$00BF	Keypad D	N/A
$00C0	Keypad E	N/A
$00C1	Keypad F	N/A
$00C2	Keypad XOR	N/A
$00C3	Keypad ^	N/A
$00C4	Keypad %	N/A
$00C5	Keypad <	N/A
$00C6	Keypad >	N/A
$00C7	Keypad &	N/A
$00C8	Keypad &&	N/A
$00C9	Keypad |	N/A
$00CA	Keypad ||	N/A
$00CB	Keypad :	N/A
$00CC	Keypad #	N/A
$00CD	Keypad Space	N/A
$00CE	Keypad @	N/A
$00CF	Keypad !	N/A
$00D0	Keypad Memory Store	N/A
$00D1	Keypad Memory Recall	N/A
$00D2	Keypad Memory Clear	N/A
$00D3	Keypad Memory Add	N/A
$00D4	Keypad Memory Subtract	N/A
$00D5	Keypad Memory Multiply	N/A

Code	Function	AT-101 Ref.
$00D6	Keypad Memory Divide	N/A
$00D7	Keypad +/-	N/A
$00D8	Keypad Clear	N/A
$00D9	Keypad Clear Entry	N/A
$00DA	Keypad Binary	N/A
$00DB	Keypad Octal	N/A
$00DC	Keypad Decimal	N/A
$00DD	Keypad Hexadecimal	N/A
$00DE-00DF	Reserved	N/A
$00E0	Keyboard LeftControl	58
$00E1	Keyboard LeftShift	44
$00E2	Keyboard LeftAlt (Option)	60
$00E3	Keyboard LeftGUI (Windows/Command)	127
$00E4	Keyboard RightControl	64
$00E5	Keyboard RightShift	57
$00E6	Keyboard RightAlt (Option)	62
$00E7	Keyboard RighGUI (Windows/Command)	128
$00E8	Locking LeftControl	58
$00E9	Locking LeftShift *	44
$00EA	Locking LeftAlt (Option) *	60
$00EB	Locking LeftGUI (Windows/Command) *	127
$00EC	Locking RightControl *	64
$00ED	Locking RightShift *	57
$00EE	Locking RightAlt (Option) *	62
$00EF	Locking RighGUI (Windows/Command) *	128
$00F0	Sticky LeftControl	58
$00F1	Sticky LeftShift	44
$00F2	Sticky LeftAlt (Option)	60
$00F3	Sticky LeftGUI (Windows/Command)	127
$00F4	Sticky RightControl	64
$00F5	Sticky RightShift	57
$00F6	Sticky RightAlt (Option)	62
$00F7	Sticky RighGUI (Windows/Command)	128
$00F8	Reserved	N/A
$00F9	Counting FN-Key	N/A
$00FA	Sticky FN2	N/A
$00FB	Sticky FN1	N/A
$00FC	Locking FN2	N/A
$00FD	Locking FN1	N/A
$00FE	FN2	N/A
$00FF	FN1	N/A

*) These codes are used as "modifier break" codes within typing mode macros.

The codes for the normal keys and modifiers are identical to the standard USB usage codes of the keyboard usage page. Which codes are used to represent the keys on the computer depends on the operating system in use. Some operating systems map the USB usage codes to a different code set internally.

The format of the table is little endian. So the bytes of the key code $0004 are arranged in memory so that $04 comes first, followed by $00.

4.1 FN Keys

To switch between the three alternative keyboard tables two FN keys are available.

When FN1 is active the KeyWarrior uses the second key table, when FN2 is active the third table is used. If both FN keys are active at the same time the third table will be used.

There are four different ways the FN keys can work.

The normal FN keys are active as long as they are held down.

A locking FN key toggles between active and not active each time it is pressed. This can be used to switch between the tables without having to hold down the FN key. If the non locking FN key is pressed it will unlock the locking FN key.

Sticky FN keys are like a one shot locking FN key. When pressed they go active and stay active until another key is pressed.

The counting FN key is stepping through the FN levels. Each time it is pressed it will advance from normal to FN1, on second press to FN2, and on the third press back to normal.

When programming FN keys remember to put the same FN key code at the same position of all three keyboard tables. Otherwise it can result in unstable behaviour, like oscillation between key tables.

Every time the FN status changes KeyWarrior first sends a release for all the currently active keys (if any), then it sends the codes for the now active keytable of any keys that may still be pressed.

4.2 Locking modifiers

$00E8 to $00EF are locking versions of the modifier keys. That means every time this key is pressed it will toggle the status of the modifier. This allows single finger operation.

Pressing the same modifier in its non locking or the sticky variant will unlock the modifier status.

4.3 Sticky modifiers

$00F0 to $00F7 are the sticky version of the modifier keys. When pressed they go active and stay active until another key has been pressed. On release of that key the modifier will also go back to inactive.

This allows another option for single finger operation.

A sticky modifier is cleared if the non locking or locking version of the same modifier is operated.

5. Page $01 - Media Controls and application keys

Following is the table of the codes for media controls and application keys, which are on page $01.

Code	Function	Usage code
$0100	Play	$B0
$0101	Pause	$B1
$0102	Record	$B2
$0103	Fast Forward	$B3
$0104	Rewind	$B4
$0105	Scan Next Track	$B5
$0106	Scan Previous Track	$B6
$0107	Stop	$B7
$0108	Eject	$B8
$0109	Random Play	$B9
$010A	Stop/Eject	$CC
$010B	Play/Pause	$CD
$010C	Mute	$E2
$010D	Volume Increment	$E9
$010E	Volume Decrement	$EA
$010F	Launch Word Processor	$0184
$0110	Launch Text Editor	$0185
$0111	Launch Spreadsheet	$0186
$0112	Launch Graphics Editor	$0187
$0113	Launch Presentation App	$0188
$0114	Launch Database App	$0189
$0115	Launch Emailer	$018A
$0116	Launch Newsreader	$018B
$0117	Launch Voicemail	$018C
$0118	Launch Address Book	$018D
$0119	Launch Calendar	$018E
$011A	Launch Calculator	$0192
$011B	Launch Local Browser	$0194
$011C	Launch Internet Browser	$0196
$011D	Command Line	$01A0
$011E	Help Center	$01A6
$011F	Find	$021F

All media control and application keys are on the USB usage page $0C for Consumer Controls.

If these keys work depends on the operating system and in some cases on the proper configuration of the system.

6. Page $02 - Macros

KeyWarrior II has a advanced macro capability that combines all macro functions of the old KeyWarrior family into a single solution.

Macros can be assigned to any key. The same macro may be put onto multiple keys.

Each macro can contain multiple key codes of the page $00, only the lower byte of the 16 bit code is used in a macro (i.e. $04 is the key for "a"). So any normal keys and modifiers may be part of a macro. Using FN keys as part of a macro is possible in the static mode but this is not recommended as it can get very tricky.

In the keyboard table a macro takes up 32 bytes each. Macro $0200 is the first 32 byte block following the third key table.

There are three modes for a macro. The first byte of each macro defines which mode the macro uses.

6.1 Mode $00 - Static macro

$00 in the first byte defines a static macro. This means all keys in the macro act as if they are pressed at the same time and they stay active until the macro key is released.

Static macros can not produce multiple copies of the same character and modifier keys will be active for all keys, no matter where they are inserted in the macro.

The remaining 31 bytes of the static macro may contain any keycodes ranging from $04 to $FF. A $00 signals the end of the macro. Any codes behind a $00 will not be used.

The useful length of a static macro is limited by the standard report form of a USB keyboard report, which allows all eight modifiers plus up to 6 normal keys to be active at once.

6.2 Mode $01 - Typing macro

Typing macros have a $01 in the first byte. Any normal keys in a typing macro are pressed and immediately released again, as if someone was typing. This allows multiple copies of the same character to be produced by a typing macro.

Locking and sticky modifiers as well as FN keys are not allowed in a typing macro. The 31 bytes may contain the codes for any normal key or modifier in 8 bit form.

Modifiers behave a bit different in typing macros. The codes $E0-$E7 behave similar as if they are locking. So when any of the modifier codes is encountered in a typing macro that modifier goes active and stays active. To release the modifier within the typing macro the codes $E8-$EF are used, which act as "break" codes for the modifiers.

This allows to generate strings with special characters, of a capital first letter.

Codes $F0 to $FF are ignored. A $00 denotes the end of the macro (if it is shorther than 31 keys), any codes after a $00 will not be used.

Modifiers do not need to be released by the end of the macro, this will happen upon releasing the macro key in any case.

6.3 Mode $02 - Cell macro

Code $02 selects the cell mode for the macro. The second byte contains a timeout value in 100 ms steps and the remaining 30 bytes may hold 8 bit key codes.

Cell mode replicates the input method for alphanumeric characters on a numeric keypad, like on an old cell phone. Pressing the same key multiple times steps through a sequence of keys each replacing the one from the last key press.

When the macro key is pressed the first code from the sequence is send immediately followed by a "left arrow" code. So the cursor will be under the newly generated character. If the same macro key is pressed again before the timeout expires first a "forward delete" is send, followed by the next code of the sequence and a "left arrow key". So the last character will be replaced by the next one and the cursor is again under that character.

When the timeout expires or a different macro key is pressed, a "right arrow" is send so the next input happens behind the last character generated by the cell mode macro. After a timeout the next code to be send by the macro key will be the first of the sequence.

If the next code in a sequence is $00 or if the end of the macro is reached, the next key press will use the first code of the macro.

6.4 Mode $03 - CellDual macro (KW100 only)

CellDual macros are a variant of the Cell macros. They are not available on all KeyWarrior variants.

Basically the CellDual macros works identical to the Cell macro. The difference is that CellDual macros can have only up to 15 codes and they override the modifier keys.

When special characters need to be generated in a cell macro the code 0x03 in the first byte selects a macro format with two bytes for each key.

The first byte for each key is a bitmap, directly representing the modifiers to be used for this key, the second byte holds the standard 8 bit key code. There can be up to 15 pairs of keycode and modifier status.

The modifier status overrides any other modifiers pressed at the time the cell dual macro is used.

Assignment of the bits for the modifier status is as follows:

0 - Left control

1 - Left shift

2 - Left alt

3 - Left GUI

4 - Right control

5 - Right shift

6 - Right alt

7 - Right GUI

There can be combinations that will not generate a character, this should be avoided as it may lead to unintended deletion of characters.

Otherwise the behaviour is exactly as for the normal cell macro.

7. Page $03 - Mouse function

KeyWarrior II always has a mouse function in addition to the keyboard. Depending on the KeyWarrior model there may be support for various mouse sensor types. All KeyWarrior II variants do suport a digital switch mouse, using 4 keys to control the mouse cursor.

Starting with firmware version 1.0.1.A diagonal mouse move keys are supported too. Older firmware versions ignore the codes $0305, $0306, $0309, and $030A.

The buttons for the mouse function are always located in the keyboard matrix. Key codes $0310-$0317 allow to assign up to eight mouse buttons to any keys.

Code	Function
$0300	Move mouse cursor right
$0301	Move mouse cursor left
$0302	Move mouse cursor down
$0303	Move mouse cursor up
$0305 *	Move mouse cursor down/right
$0306 *	Move mouse cursor down/left
$0309 *	Move mouse cursor up/right
$030A *	Move mouse cursor up/left
$0310	Left mouse button
$0311	Right mouse button
$0312	Center mouse button
$0313	Mouse button 4
$0314	Mouse button 5
$0315	Mouse button 6
$0316	Mouse button 7
$0317	Mouse button 8

*) Function is available since firmware version V1.0.1.A

8. Page $04 - Dimmable LED controls

KeyWarrior100 has two DAC and three PWM outputs which are intended to drive LEDs i.e. for backlighting.

In addition to a direct control via commands on USB, the five outputs can be controlled with keys.

The "brightness" for each output is represented by an 8 bit value. 0 means off, 255 is maximum.

Following are the keycodes wich allow to control each individual channel:

Code	Function
$0400	DAC0 off
$0401	DAC0 max
$0402	DAC0 step up
$0403	DAC0 step down
$0404	DAC0 step und and wrap
$0405	DAC0 dim up
$0406	DAC0 dim down
$0407	DAC0 dim up/down
$0408	DAC0 toggle *
$0410	DAC1 off
$0411	DAC1 max
$0412	DAC1 step up
$0413	DAC1 step down
$0414	DAC1 step und and wrap
$0415	DAC1 dim up
$0416	DAC1 dim down
$0417	DAC1 dim up/down
$0418	DAC1 toggle *
$0420	PWM0 off
$0421	PWM0 max
$0422	PWM0 step up
$0423	PWM0 step down
$0424	PWM0 step und and wrap
$0425	PWM0 dim up
$0426	PWM0 dim down
$0427	PWM0 dim up/down
$0428	PWM0 toggle *
$0430	PWM1 off
$0431	PWM1 max
$0432	PWM1 step up
$0433	PWM1 step down
$0434	PWM1 step und and wrap
$0435	PWM1 dim up
$0436	PWM1 dim down
$0437	PWM1 dim up/down
$0438	PWM1 toggle *
$0440	PWM2 off
$0441	PWM2 max
$0442	PWM2 step up
$0443	PWM2 step down
$0444	PWM2 step und and wrap
$0445	PWM2 dim up
$0446	PWM2 dim down
$0447	PWM2 dim up/down
$0448	PWM2 toggle *

*) Function is available since firmware V1.0.1.B older versions ignore these codes

"off" switches the output immediately to its off state.

"Max" switches the output immediately to its maximum.

"step up" increases the brightness for the channel by one step defined when setting up the function. The brightness value saturates at 255, so if the step value is not a integer fraction of 255 it will still be possible to reach the maximum.

"step down" decreases the brightness for the channel by one step value. It stops at zero.

"step up and wrap" does increase the brightness by one step until it reaches 255, then on the next key press it will jump to zero for off.

"dim up" increases the brightness as long as the key is held down until it reaches maximum. The step value in this case defines the milliseconds between each increase.

"dim down" decreases the brightness as long as the key is held down.

"dim up/down" works similar, but it changes the direction of dimming each time it is pressed.

"toggle" switches between off and the last brightness. If the channel has not been dimmed since the last reset the default value will be used.

9. Page $05 - User macros

In addition to the 176 macros which are intended to be programmed by the keyboard manufacturer KeyWarrior100 supports 64 user macros.

User Macros are programmed onto a key with the page number $05 and the number of the macro in the lower byte. $0500 puts the first of the user macros onto a key.

User macros are separate from the manufacturer macros, so user macro 0 is a different macro than manufacturer macro 0.

The data format and function of user macros is identical to manufacturer macros. Please refer to chapter 6 for a detailed description.

8. Page $E0 - $E7 - Dual codes

The pages $E0 to $E7 basically use the high byte of the keycode to add a modifier key to a normal key, another modifier, or FN key.

This allows a simple combination of a modifier plus any page $00 key on every physical key. So such small key combinations do not require a complete macro, which is of interest if the number of macros on the KeyWarrior model would otherwise be insufficient.

The lower byte holds an 8 bit keycode, the upper byte the 8 bit code of a modifier.

9. Keyboard table memory layout

The exact layout of the keyboard table depends on the KeyWarrior model and its matrix size. But the basic layout is always the same.

The primary table contains two bytes for each of the physical keys. The lower byte of the 16 bit keycode is on the lower address. So the bytes for the code $0310 (left mouse button) is stored in the sequence: $10, $03.

There are three instances of the primary table. The first is used if no FN key is active, the second if FN1 is active and the third if FN2 or both FN keys are active. This allows to switch between different modes for a keyboard for various tasks, or to get full functionality from a small keyboard.

The three primary tables are consecutive in memory and are followed by a number of 32 byte blocks each containing a macro. The number of macros depends on the KeyWarrior model.

9.1 KeyWarrior28 Keyboard table

Following is the mapping of the keyboard table for a KeyWarrior28 which has a 8x8 matrix. The first entries are detailed to show how the bytes are arranged.

The total size of the keyboard table for a KeyWarrior28 is 992 bytes arranged in 31 data blocks of 32 bytes each

Memory position	Content
$0000	Lower byte of keycode for key on X0, Y0 with no FN key active
$0001	Upper byte of keycode for key on X0, Y0 with no FN key active
$0002-$0003	Keycode for key on X1, Y0 with no FN key active
$0004-$000F	Keycodes for keys on X2, Y0 to X7, Y0 with no FN key active
$0010-$001F	Keycodes for keys on X0, Y1 to X7, Y1 with no FN key active
$0020-$007F	Keycodes for keys on X0, Y2 to X7, Y7 with no FN key active
$0080-$00FF	Keycodes for keys on X0, Y0 to X7, Y7 with FN1 key active
$0100-$017F	Keycodes for keys on X0, Y0 to X7, Y7 with FN2 key active
$0180-$01BF	Macro 0
$01C0-$01FF	Macro 1
$0200-$03BF	Macros 2 to 18

9.2 KeyWarrior100 Keyboard table

Following is the mapping of the keyboard table for a KeyWarrior100 which has a 24x16 matrix. The first entries are detailed to show how the bytes are arranged.

The total size of the keyboard table for a KeyWarrior100 is 7936 bytes arranged in 248 data blocks of 32 bytes each

Memory position	Content
$0000	Lower byte of keycode for key on X0, Y0 with no FN key active
$0001	Upper byte of keycode for key on X0, Y0 with no FN key active
$0002-$0003	Keycode for key on X1, Y0 with no FN key active
$0004-$001F	Keycodes for keys on X2, Y0 to X15, Y0 with no FN key active
$0020-$003F	Keycodes for keys on X0, Y1 to X15, Y1 with no FN key active
$0040-$02FF	Keycodes for keys on X0, Y2 to X15, Y23 with no FN key active
$0300-$05FF	Keycodes for keys on X0, Y0 to X15, Y23 with FN1 key active
$0600-$08FF	Keycodes for keys on X0, Y0 to X15, Y23 with FN2 key active
$0900-$091F	Macro 0
$0920-$093F	Macro 1
$0940-$1EFF	Macros 2 to 175

Legal Stuff

Code Mercenaries

Hard- und Software GmbH

Karl-Marx-Str. 147a

12529 Schönefeld

Germany

Tel: +49-3379-20509-20

Mail: support@codemercs.com

Web: www.codemercs.com

HRB 9868 CB

Geschäftsführer: Guido Körber, Christian Lucht