KeyWarrior100

High key count USB Keyboard/Mouse Combo Controller with advanced programming features KeyWarrior II family

Code Mercenaries

1. Features

- USB full speed interface
- USB HID 1.1 compliant
- Up to 384 keys in 24x16 matrix
- Two function shift keys (FN) to switch to a second and third key table
- Keyboard layout programmable via USB
- Each key can be a modifier plus a typing key
- Mouse function with cursor control via keys
- Mouse function with quadrature encoder input for trackball etc.
- Mouse function with analog joysick (potentiometer and hall supported)
- Mouse function with four channel analog sensor
- Scroll wheel function via analog axis or optical encoder
- 32 media control and application keys supported
- Up to 240 macros with up to 31 keys each
- 64 dynamically reprogrammable macros
- Macros can be static, typing, or cell phone like
- Security features prevent overwriting keyboard layout by users
- Unique serial number for definite identification
- 16 bytes of programmable customer data for version tracking and other purposes
- 8 LED outputs programmable to indicate Caps lock, Num lock, Scroll lock, Compose, Kana, FN1, FN2, USB active, locking and sticky modifiers
- FN function can be activated by the lock LEDs
- Three PWM outputs i.e. for background lighting
- Two DAC outputs i.e. for background lighting
- 16 generic I/O lines, similar to IO-Warrior
- Fast I2C master interface to access peripheral circuits
- Support for HD44780 compatible display modules (LCD, OLED etc.)
- Support for 8x64 = 512 LED matrix
- Support for 500 WS-type digital RGB LEDs
- Single 3.3 V power supply
- Low power consumption: 30 mA max. (chip)
- Available in LQFP100 package

1.1 Variants

KeyWarrior100 is currently available in a single variant.

An evaluation board simplifies initial design steps and can be used as a component for small volume designs.

2. Functional overview

KeyWarrior100 is a very flexible keyboard/mouse combo controller with additional I/O capabilites.

It is designed to cover most needs for any kind of keyboard and mouse input devices for desktop, industrial, or any kind of special purpose application.

Several options are available for mouse sensors, including trackball, analog joystick and force sensors.

Three PWM and two DAC outputs for driving background lighting can be directly controlled by keys or by commands via USB.

16 generic I/O lines offer peripheral functions similar to our IO-Warrior controllers. Among the functions on the I/Os are a fast I2C master, driving LCD modules, driving a 8x64 LED matrix, and driving up to 500 digital RGB LEDs.

Adding an auxiliary display, sensors, actors, or a large indicator panel are easy with these I/Os.

Programmability of the key functions is very flexible. Any key can be assigned to have any function. Two FN keys allow to switch to a second and a third key table, which enables complex functions on smaller keyboard layouts.

Extensive macro capabilities allow any of the 240 macros to be programmed on any of the 384 keys in any FN level. Macros can generate either a simultaneous activation of multiple keycodes, a typing like press and release of a string of keycodes, or a cell phone like stepping through multiple keycodes.

The eight lock-LED outputs have no fixed function, instead each can be assigned to indicate any locking state, including the internal lock states for the FN keys and the modifiers.

32 media control and application control keycodes can be assigned to any of the keys. This allows functions like launching the browser, or controlling the sound volume.

KeyWarrior100 requires few external components. It runs at 3.3 V so a low drop voltage regulator is required. In addition it needs capacitors for the power supply and pull up resistors for the X-lines of the keyboard matrix. Additional protection and filtering may be required depending on the application.

2.1 Differences to old KeyWarrior variants

KeyWarrior100 is the successor of the KeyWarrior8LED, KeyWarrior16, KeyWarrior20, and KeyWarriorCombo families.

KeyWarrior100 combines all programming options of the old KeyWarrior family, that formerly have been available in separate chips (Flex, Operator, Commander, Cell), plus several features that formerly have been available only in custom designs and a number of completely new features.

Since KeyWarrior100 is highly configurable there is only a single version and not multiple versions as with the old KeyWarrior family.

There is only a limited backward compatibility.

KeyWarrior100 works with 3.3 V. External connections must observe this. Though some of the pins are 5 V tolerant.

The total cost for a design with KeyWarrior100 is comparable with the old KeyWarrior family but it offers vastly more functionality.

2.2 Configuration

KeyWarrior100 has internal FLASH memory that holds the keyboard table and configuration data. A tool for setting the configuration and downloading it into the KeyWarrior100 is available.

To prevent end users from reconfiguring a KeyWarrior100 based keyboard into an unusable state advanced programming and protection features are implemented to prevent end user tampering with the keyboard setup. The configuration can be protected against reading and writing. Additionally a custom 64 bit PIN code can even prevent erasing the controller by the end user.

End user programmable macros are supported. The access to these macros is separated from the configuration functions of the KeyWarrior100 to make sure that the end user can not change the vital settings.

16 bytes of customer data storage allow the keyboard manufacturer to store version and/or model information in the chip to better track the life cycle of the product. A factory programmed unique serial number does allow to definitely identify the individual KeyWarrior100.

Tampering can also be detected via a 16 bit erase counter, that counts how many times the configuration of the KeyWarrior100 has been erased.

Serial number and erase counter can not be changed by external manipulation.

2.3 Evaluation kit

The KeyWarrior100 evaluation kit is intended for easy and fast access to the full functionality of the KeyWarrior100.

It comes equipped with all signals grouped logically with the option to solder connectors to the positions. Eight LEDs for the Lock LEDs and a RGB LED for the PWM outputs are also on the board.

Disconnecting the on board LEDs can be done by cutting the jumper patterns next to them.

The board is ready to use. The keyboard configuration has to be uploaded first, otherweise the KeyWarrior100 will not generate any input and not indicate any lock status.

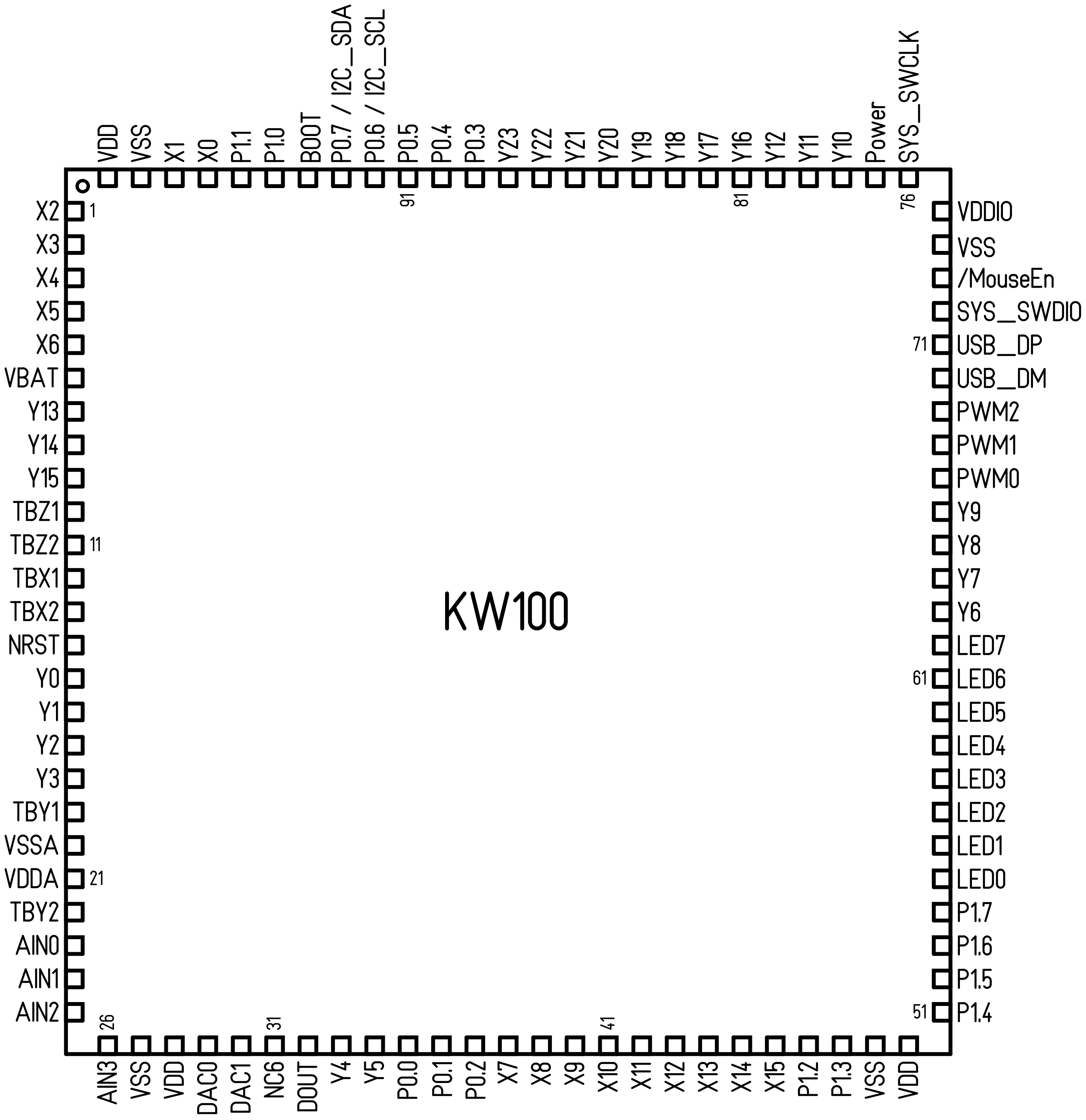
The Dout output for digital RGB LEDs is labeled "PA7" on the evalboard

KeyWarrior100

Content

3. Pin configuration

KeyWarrior100-LF100
LQFP100



TOP VIEW!

4. Pin descriptions KeyWarrior100-LQFP100

Name	I/O	Type	Pins	Description
USBP, USBM	I/O	special	71, 70	USB differential data lines
Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8. Y9. Y10, Y11, Y12, Y13, Y14, Y15, Y16, Y17, Y18, Y19, Y20, Y21, Y22, Y23	O	open drain outputs, no pullups	15, 16, 17, 18, 33, 34, 63, 64, 65, 66, 78, 79, 80, 7, 8, 9, 81, 82, 83, 84, 85, 86, 87, 88	Y lines for key matrix. These lines are periodically pulled low, between matrix scan they are high impedance.
X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15	I	inputs with weak internal pull ups	97, 98, 1, 2, 3, 4, 5, 6, 38, 39, 40, 41, 42, 43, 44, 45, 46	X lines for key matrix. Weak internal pull up. Add 1 kΩ pull up to 3.3 V externally for all used pins
TestR, TestD, TestC		special	14, 72, 76	Used during manufacturing, do not connect.
Vss, Vssa		power supply	20, 27, 49, 74, 99	Ground
Vdd, Vddio, Vdda		power supply	21, 28, 50, 75, 100	Supply voltage, connect to 3.3 V
Power	I	input, internal pull up	77	Sets the power requirement, high = 500 mA
TBX1, TBX2, TBY1, TBY2	I	input, 5 V tolerant	12, 13, 19, 22	Quadrature inputs for trackball or similar device
TBZ1, TBZ2	I	input, 5 V tolerant	10, 11	Quadrature inputs for optical scroll wheel
/MouseEN	O	open drain output, 5 V tolerant	73	Enable output for mouse electronics, can sink current for encoder LEDs up to 20 mA
AIN0, AIN1, AIN2, AIN3	I	analog input	23, 24, 25, 26	Analog inputs for mouse sensor
PWM0, PWM1, PWM2	O	open drain outputs	67, 68, 69	PWM outputs i.e. for driving background lighting
DAC0, DAC1	O	analog output	29, 30	Analog outputs, i.e. for driving background lighting
Dout (also NC7 or PA7)	O	push/pull fast output	32	Serial output for intelligent LEDs, controls up to 500 RGB LEDs
P0.0…P0.7 P1.0…P1.7	I/O		35, 36, 37, 89, 90, 91, 92, 93, 95, 96, 47, 48, 51, 52, 53, 54	Generic I/O
LED0. LED1. LED2. LED3. LED4. LED5. LED6. LED7	O	open drain outputs, 5 V tolerant	55. 56. 57, 58, 59, 60, 61, 62	Outputs for lock LEDs

4.1 Pin Functions KeyWarrior100-LQ100

USBP, USBM
Differential data lines of USB. Connect these signals direct to the USB cable or type B plug.

X[0:15]
Matrix horizontal inputs. These lines are read by KeyWarrior to detect pressed keys.
Internal pull up resistors are activated on device reset. The addition of external pull up resistors in the range of 1 kΩ to 4.7 kΩ is recommended for all lines that are used.

Y[0:23]
Vertical matrix outputs. These open drain outputs are periodically pulled low to detect pressed keys. No internal or external pull up resistors.

TestR, TestD, TestC
These pins are used during production of the KeyWarrior chips, do not connect.

Power
Allows to select if KeyWarrior100 should request 100 mA or 500 mA from the host.
A high on power up on this pin selects 500 mA, a low 100 mA.
Input, internal pull up.

TBX1, TBX2, TBY1, TBY2
Inputs for optical quadrature encoded trackball or similar mouse mechanism. TBX1 falling edge leads TBX2 falling edge for right movement. TBY1 falling edge leads TBY2 falling edge for up movement.
Input, no internal pull up/down resistors, 5 V tolerant.

TBZ1, TBZ2
Inputs for optical encoder scroll wheel.
Input, no internal pull up/down resistors, 5 V tolerant.

/MouseEN
Enable output for external mouse electronics. Active low, signals to the mouse electronics when it should work and when to enter sleep mode for minimum power use.
Output, open drain, no internal pull up, 5 V tolerant.

Ain0, Ain1, Ain2, Ain3
Analog inputs for mouse sensors.
Input range from Vss to Vdda. Input impedance max. 50 kΩ.
The function of the signals is determined by the selected mouse sensor configuration.

PWM0, PWM1, PWM2
PWM outputs i.e. for background lighting. May be controlled with dedicated keys, or via USB.
Active low, frequency ~730 Hz
Open drain output, weak internal pull up.

DAC0, DAC1
Analog outputs i.e. for background lighting. May be controlled with dedicated keys, or via USB.
Analog output, ranging from Vss to Vdda.

Dout
(may be labeled as NC7 or PA7)
Output to drive up to 500 serial RGB LEDs.
Fast push-pull output. Idles low.

P0.0…0.7, P1.0…P1.7
Generic I/O. Similar to IO-Warrior.
Higher functions like I2C master and simple I/O pins.
Inputs and open drain outputs with internal pull up resistor.

LED0…7
Outputs to drive the lock LEDs. The function of each individual pin is progammable.
Open drain, can directly drive LEDs with up to 25 mA (a 80 mA limit to the total combined current into all pins does apply!),

Vss
Power supply ground.

Vdd, Vddi, Vdda
Supply voltages, connect to 3.3 V
Vddi supplies the peripheral pins of KW100.
Vdda supplies the analog functions and serves as the high reference for the analog inputs and DAC outputs. Filtering the voltage for Vdda increases stability of the analog signals.

4.2 Special mode pin functions

KeyWarrior100 supports driving I2C compatible chips, HD44780 compatible display modules and some other display modules direct. Handling I2C via the normal generic I/O would be very slow as each edge of data and clock would have to be transmitted separately. At a rate of 1000 such transactions per second (which is the maximum KeyWarrior100 is allowed by USB specifications) the maximum bit rate would be around 500 bits/sec.
To make I2C and other devices usable, KeyWarrior100 implements the special mode functions. By handling the I2C inside KeyWarrior100 the actual data rate is increased by some orders of magnitude.
When any of the special mode functions is activated a couple pins will no longer respond as generic I/O pins but are under control of the activated special mode function.

4.2.1 I2C mode pins

KeyWarrior100 can act as an I2C master. It offers speeds from 10 kBit/s to 1000 kBit/s and has been optimized for maximum performance, allowing transfers of > 60 kByte/s. This allows to get close to the theoretical maximum of I2C data transfer except for the fastest speed at 1 MBit/s, where about 60% of the theoretical throughput is possible.
The following pins get reassigned when the I2C function is enabled:

Function	Port pin
SCL	P0.6
SDA	P0.7

These pins will no longer be affected by the data sent via the normal port setting command. Both pins have internal pull up resistors and can be connected direct to I2C compatible chips.
External 1 kΩ pull-up resistors to 3.3 V are recommended. Without them the I2C performance may be significantly reduced.
KeyWarrior100 monitors its I2C lines and can adapt to relatively high parasitic capacitance on the bus, but this will reduce the effective data rate by slowing down SCL.

4.2.2 LCD mode pins

KeyWarrior100 has support for controlling alphanumeric display modules based on or compatible with HD44780 as well as some graphic displays which use the same interface.
For simplicity the function is refered to as the LCD function, but there are also OLED and vacuum fluorescence displays that are compatible with this interface.
The following pins get reassigned when the LCD function is enabled:

Function	Port pin
/On	P0.0
RS	P0.1
R/W	P0.2
E	P0.3
Data0	P1.0
Data1	P1.1
Data2	P1.2
Data3	P1.3
Data4	P1.4
Data5	P1.5
Data6	P1.6
Data7	P1.7

When the LCD function is enabled these pins will no longer be affected by the normal port setting command.
/On should be used to enable power supply to LCD modules that have high current demand or backlighting. The /On signal is low when the LCD function is enabled, it does go high when KeyWarrior100 enters suspend mode or when the LCD function is disabled.
The LCD function requires displays that work with 3.3 V power supply at least for the logic part.

4.2.3 LED matrix mode pins

KeyWarrior100 supports driving a LED matrix with up to 8x64 LEDs.

Function	Port pin
Strb	P0.2
Clk	P0.3
/OE	P0.4
Data	P0.5

When the LED matrix function is enabled these pins will no longer be affected by the normal port setting command.

/OE is driven high when KeyWarrior100 enters the suspend mode. The external driver should then disable the LEDs to stay within the USB power limits for suspend mode.

For more details on how to control a LED matrix please refer to the separate application note.

4.2.4 Digital LED mode pins

KeyWarrior100 supports driving up to 500 digital LEDs with a serial communication like WS2812/WS2812B.

The timing is programmable to accomodate many different variants of the digital LEDs

The intelligent LED function has its own dedicated pin (Dout, on the starterkit it is labeled "PA7") on KeyWarrior100. Activating the digital LED function does not take any pins away from the general purpose I/O.

This function was added with V1.0.1.4

5. Device Operation

KeyWarrior100 registers as a standard HID keyboard and mouse and supports boot protocol. It does not need any special drivers to be installed, standard system drivers are sufficient.

The country code is 0 for not localized hardware, which allows to use a single version of the chip for all international keyboard layouts. Usage codes are defined for 0 to 221, which include the power key and the = sign in the keypad, as well as the compose keys for Asian languages and several special keys that may or may not be supported by individual operating systems.

In addition the media control keys Mute, Play/ Pause, Eject, Fast Forward, Fast Backward, and several application control keys are supported via a separate function. The interface for programming the key table registers as a fourth function.

The 64 end user programmable macros are loaded from persistent memory on reset of the KeyWarrior100. This allows to set a default configuration for the macros, which can be overwritten at any time.

Two more USB interfaces serve the generic I/O. Details are described in chapter 7.

5.1 Power Up

Every time the supply voltage is applied KeyWarrior100 executes an internal reset sequence. All internal pull up resistors are disabled upon power up and will be activated during the internal reset sequence.

5.2 Keyboard Scanning

KeyWarrior100 scans the keyboard matrix every t_{scan} by sequentially pulling one of the Y lines low and then reading the status at the X lines. When the scan matrix changes status and then remains stable for $t_{debounce}$ KeyWarrior100 decodes the changes and generates scancodes.

5.3 Key Rollover

KeyWarrior supports n-key rollover. All keys in the matrix may be pressed at the same time without KeyWarrior missing a code. However due to the phantom key effect it can not be guaranteed that combinations of many keys are properly reported (see 5.3.1).

USB has a limitation on how many keys can be reported at the same time. On USB any six keys plus all eight modifiers (GUI, Ctrl, Alt, Shift) may be pressed at the same time. If more than six standard keys are pressed an error state is reported. So USB has a 6-key plus modifiers rollover.

5.3.1 Phantom Keys

Phantom keys do occur when the keys on three corners of an imaginary rectangle in the matrix are pressed, the fourth corner then appears to be pressed too.

To avoid phantom keys diodes may be added to the keys. The diodes have to be put in series with the key switches. The kathodes have to be connected to the Y lines and anodes to X lines.

It is highly recommended to at least place all modifier keys on a single row or column and put diodes on all of them.

The large matrix size supported by KeyWarrior100 usually makes it easy to arrange keys in a way to prevent phantom keys.

5.4 Customizing the keyboard layout

KeyWarrior100 allows to load custom keyboard layouts into the controller. Tools for generating the tables and downloading them into the controller are provided for Windows.

The functions for downloading keyboard tables, configuring functions, and setting the protection features can also be accessed directly with your own software. All functions for the configuration of KeyWarrior100 are accessed via USB interface 3, which is like a virtual device in KeyWarrior100. The interface is named "ConfigInterface".

There are several commands on this interface to read and write data. Some of the commands can be locked, most require a PIN code.

5.5 Configuration protection scheme

To prevent end user tampering with the keyboard layout and general configuration KeyWarrior100 offers a couple protection options.

All commands that access the key layout data or changes settings have to provide a 64 bit PIN code. Factory setting for this PIN code is 0xFFFF.FFFF.FFFF.FFFF. If a command provides a wrong PIN code KeyWarrior100 will disable all commands except command 0. A USB reset, or unplugging is required to access the commands again.

The PIN can be changed to a custom PIN, making it basically impossible to access the commands by guessing the pin. Though losing the PIN means there is no way to even erase the KeyWarrior100 again.

It is also possible to lock the keytable. This will prevent any reading or writing access to the keytable, changing the PIN code, or writing the customer data. The only way to reset this mode is by erasing the configuration which can only be done if the PIN is known.

5.6 Reading the device information

Command 0 is always accessible, no matter what protection level has been activated.

The command is initiated by sending a 64 byte report to interface 3 with a 0x00 in the first byte (all other bytes should be zero too for future compatibility). In return KeyWarrior100 sends a 64 byte report with the following content:

Byte#	Data
0	0x00 - answer to command 0
1…3	unused, 0
4…19	Customer data
20…21	Lock status
22…23	Erase count
24…27	Serial number
28…63	unused, 0

Customer data can be programmed with command 5 and may contain whatever the customer wants to store here. For example it may be used to keep version info for the keyboard layout, a model number, or other product information.

Lock status is 0xFFFF for the unlocked status of the data, 0x0000 designates read and write protection.

Erase count shows the number of times the keytable data has been erased. This count can not be manipulated and will saturate at 0xFFFF.

Serial number is a factory programmed serial number that uniquely identifies the KeyWarrior100 chip. It can not be manipulated.

5.7 Reading the keytable

The keytable data can be read with command 1. This command may be disabled by the Lock function. The valid PIN code is required, supplying the wrong code does lock the commands.

Keytable data is read in blocks of 32 bytes each. There are 248 blocks.

The read command has the following format:

Byte#	Data
0	0x01 - Read data
1	Block number (0 to 247)
2…3	unused, write 0
4…11	PIN code
12…63	unused, write 0

It generates a return report with 64 bytes with the following content:

Byte#	Data
0	0x01 - answer to Read data
1	Error code, 0 = no error
2…3	unused, 0
4…35	Keytable data block
36…63	unused, 0

Error code is zero if no error occured, 1 means the block number was out of range, 4 is returned if the keytable is locked.

5.8 Writing the keytable

Command 2 is used to write the keytable in blocks of 32 bytes each.

Writing is possible only if the keytable has been erased. It is not possible to directly overwrite the keytable data.

This command may be disabled by the lock function. The valid PIN code is required, supplying the wrong code does lock the commands.

The write command has the following format:

Byte#	Data
0	0x02 - Write data
1	Block number (0 to 247)
2…3	unused, write 0
4…11	PIN code
12…43	Keytable data block
44…63	unused, write 0

The return report supplies just an error code:

Byte#	Data
0	0x02 - answer to Write data
1	Error code, 0 = no error
2…63	unused, 0

Error code is zero if no error occured, 1 means the block number was out of range, 3 means the write did fail, 4 is returned if the keytable is locked.

5.9 Erasing the configuration data

Command 3 erases the keytable data, configuration, and customer data, resets the PIN to default, removes the lock, and increments the erase count. Erasing puts 0xFF in all keytable positions.

This command is disabled if a wrong PIN code was supplied.

Byte#	Data
0	0x03 - Erase data
1…3	unused, write 0
4…11	PIN code
12…63	unused, write 0

The return report supplies just an error code:

Byte#	Data
0	0x03 - answer to Erase data
1	Error code, 0 = no error
2…63	unused, 0

Error code is zero if no error occured, 2 means the erase count failed to write, 3 means the erase did fail.

5.10 Set new PIN

The default PIN can be replaced by a custom PIN. This is possible only when the default PIN is active, so either in factory status, or after erasing.

Command 4 sets the new PIN code.

Byte#	Data
0	0x04 - Set PIN Code
1…3	unused, write 0
4…11	0xFFFF.FFFF.FFFF.FFFF
12…19	new PIN code
20…63	unused, write 0

This works only if the default PIN code is active.

The return report supplies just an error code:

Byte#	Data
0	0x04 - answer to Set PIN Code
1	Error code, 0 = no error
2…63	unused, 0

Error code is zero if no error occured, 3 means the write did fail, 4 is returned if the keytable is locked.

5.11 Set customer data

Command 5 allows to write 16 bytes of customer data. This is only possible when no customer data has been written previously. To overwrite customer data it is neccessary to first erase the keytable.

Byte#	Data
0	0x05 - Write custom data
1…3	unused, write 0
4…11	PIN code
12…27	Customer data
28…63	unused, write 0

The return report supplies just an error code:

Byte#	Data
0	0x05 - answer to Set customer data
1	Error code, 0 = no error
2…63	unused, 0

Error code is zero if no error occured, 3 means the write did fail, 4 is returned if the keytable is locked.

5.12 Lock configuration

Access to the configuration data can be denied. When setting the lock condition the only commands available are the erase and the get device information commands. This prevents an end user from overwriting any configuration data or setting a new PIN code. The lock condition can only be reset by erasing the configuration data, which requires to know the PIN code.

Byte#	Data
0	0x06 - Lock configuration
1…3	unused, write 0
4…11	PIN code
12…63	unused, write 0

The return report supplies just an error code:

Byte#	Data
0	0x06 - answer to Lock configuration
1	Error code, 0 = no error
2…63	unused, 0

Error code is zero if no error occured, 3 means writing the lock condition did fail.

5.13 LED configuration

There are three LED related functions that can be configured on KeyWarrior100:

- Each of the eight LED outputs can be configured to indicate one of several states of the keyboard
- The five standard lock LEDs (Caps, Num, Scroll, Compose, Kana) can be used to activate the FN1 or FN2 key functions.
- DAC and PWM outputs can be configured for driving LEDs.

Command 7 allows to read the LED configuration:

Byte#	Data
0	0x07 - Read LED configuration
1…3	unused, write 0
4…11	PIN code
12…63	unused, write 0

The return report supplies an error code and the configuration byte:

Byte#	Data
0	0x07 - answer to Read LED config.
1	Error code, 0 = no error
2…3	unused, 0
4	LEDFN configuration byte
5	reserved, should read 0xFF
6…13	LEDx function
14…28	DimLED function
29	reserved, should read 0xFF
30…63	unused, 0

See chapters 5.13.1, 5.13.2, 5.13.3 for the detailed description of the LED functions.

The LED configuration can be set with Command 8:

Byte#	Data
0	0x08 - Write LED configuration
1…3	unused, write 0
4…11	PIN code
12	LEDFN configuration byte
13	reserved, write 0xFF
14…21	LEDx function
22…36	DimLED function
37	reserved, write 0xFF
38…63	unused, 0

The return report supplies just an error code:

Byte#	Data
0	0x08 - answer to Write LED config.
1	Error code, 0 = no error
2…63	unused, 0

Error code is zero if no error occured, 3 means the write did fail, 4 is returned if the keytable is locked.

5.13.1 Using Lock LEDs as FN function

If the LEDFN function is activated an active lock LED will result in the same behaviour as if the corresponding FN key is pressed.

The LEDFN configuration byte contains the number of the lock LED the FN1 and FN2 functions react to. The lower nibble contains the lock LED for FN1, the upper nibble the one for FN2:

0 - none
1 - Num
2 - Caps
3 - Scroll
4 - Compose
5 - Kana
6…E - reserved, do not use
F - none

The lock LEDs status may be used for the LEDFN function even if there is no LED output configured to indicate that status.

Setting the LEDFN configuration to i.e. 0x12 will result in FN1 being active if the Caps lock LED is on and FN2 when the Num lock LED is on.

5.13.2 Configuring lock LED outputs

KeyWarrior100 has no fixed assignment of the LED outputs to a lock status. Each of the eight LED outputs can individually set to indicate one of several states of the keyboard.

There is a byte for each of the LEDx outputs which has the following values:

0x00 - Constant on (indicating non sleeping state)
0x01 - Num Lock
0x02 - Caps Lock
0x03 - Scroll Lock
0x04 - Compose
0x05 - Kana
0x10 - FN1 (standard, sticky, locking, counting)
0x11 - FN2 (standard, sticky, locking, counting)
0x2x - Locking modfiers (bitmask in lower nibble)
0x3x - Sticky modifiers (bitmask in lower nibble)
0xFF - Off, LED not used

All other values are reserved, do not use.

The indication for the modifiers allows to display the locking or sticky status of multiple modifiers. The lower nibble encodes the modifiers as a bitmask and does not distinguish between left and right:

Bit #
0 - Ctrl
1 - Shift
2 - Alt
3 - GUI

So setting 0x23 for a LED will have it indicating the locked status for the shift and control keys.

5.13.3 Dimming outputs for backlight

The DimLED function configures the two DAC and three PWM outputs for control via keys.

There are three parameters for each channel: Initial value, step, and lin/log mapping flag. Initial value is an 8 bit value to which the channel is set after reset. Step is used for incrementing or decrementing the value on key press, or as a delay in ms between dimming up or down one step, lin/log flag selects wether the mapping of the 8 bit brightness value to the 12 bits of the DACs or 16 bits of the PWMs is linear or logarithmic.

The data format is as follows:

DAC0 initial value
DAC1 initial value
PWM0 initial value
PWM1 initial value
PWM2 initial value
DAC0 step
DAC1 step
PWM0 step
PWM1 step
PWM2 step
DAC0 flags
DAC1 flags
PWM0 flags
PWM1 flags
PWM2 flags

flags currently has only the lowest bit defined, 0 means linear, 1 logarithmic.

5.14 Mouse configuration

KeyWarrior100 offers several options to provide a mouse function. There is no special configuration necessary to use the key based mouse function.

For the trackball, joystick, and force sensor options a number of parameters may be set to customize the mouse function.

Command 9 allows to read the mouse configuration:

Byte#	Data
0	0x09 - Read mouse configuration
1…3	unused, write 0
4…11	PIN code
12…63	unused, write 0

The return report supplies an error code and the configuration byte:

Byte#	Data
0	0x09 - answer to Read mouse config.
1	Error code, 0 = no error
2…3	unused, 0
4	MouseFunction
5	MouseOption
6	MouseAxisMin
7	MouseAxisMax
8	DeadZone
9	DeadZoneZ
10	Ballistics flags
11…18	Ballistics table
19…63	unused, 0

The mouse configuration can be set with Command 10:

Byte#	Data
0	0x0A - Write mouse configuration
1…3	unused, write 0
4…11	PIN code
12	MouseFunction
13	MouseOption
14	MouseAxisMin
15	MouseAxisMax
16	DeadZone
17	DeadZoneZ
18	Ballistics flags
19…26	Ballistics table
27	reserved, write 0xFF
28…63	unused, write 0

The return report supplies just an error code:

Byte#	Data
0	0x0A - answer to Write mouse config.
1	Error code, 0 = no error
2…63	unused, 0

There are five MouseFunction values defined. The default value 0xFF means the sensor based mouse function is off, though the key based mouse function may be used.

0x00 - Trackball, TBXx/TBYx inputs will be used
0x01 - Joystick, AIN0 and AIN1 are used for X/Y
0x02 - 4 quadrant sensor, east/west, south/north
0x03 - 4 quadrant mode, slow mode

When the trackball mode is in use the quadrature signals on TBX1, TBX2, TBY1. TBY2 control the movement of the mouse cursor.

In joystick mode the voltage values on AIN0 and AIN1 control the X and Y axes. There are several configureable parameters that fine tune the behaviour.

The 4 quadrant sensor mode allows the use of force sensors that have a sensor element for each direction. AIN0 and AIN1 control east/west, AIN2 and AIN3 south/north movement in this mode.

MouseOption contains a number of flags for the mouse operation:

0 - Analog wheel
1 - Optical wheel
2 - Auto center
3 - Auto range
4 - Reduced range
5 - Invert X
6 - Invert Y
7 - Invert Z

Setting any of the bits to 1 activates the option.

Analog wheel converts AIN2 into scroll wheel input. This function is available only in joystick mode.

Optical wheel uses the quadrature signal on TBZ1, TBZ2 to generate scroll wheel input. This option is available for all mouse modes.

Auto center does recalculate the center position if X or Y are off center and do not change their value for more than 4 sec. This automatically brings the mouse cursor to a halt if the mouse sensor is stuck off center, or has drifted. This works only if Auto range is also active.

Auto range compensates for variations in the sensor range. It initially assumes a maximum travel of the sensor less than 100% and it uses the initial axis position as the center position. If the axis does exceed the inital assumed travel range the calibration is recalculated on the fly. MouseAxisMin and MouseAxisMax are used as the initial min and max values for the joystick travel range.

Each axis uses 8 bits with 127 as the center position. The initial min/max values are absolute. For an initial assumed range of ±50% min would be 63 and max 190.

Reduced range works in combination with auto range. It cuts the re-ranging of the axes to 50%. So even if the axis goes beyond the 50% it will not re-range to more than ±50%. This can be used for example for sensors which do not reach their maximum values in the diagonal direction.

Invert X/Y/Z allows to invert each of the axes to adapt to the sensor in use. This applies only to joystick and 4 quadrant mode.

DeadZone and DeadZoneZ define the zone around the center position in which no cursor movement or scroll wheel action take place.

Ballistics flags allows to enable a ballistic acceleration curve for each of the axes. They are enabled by the following bits:

0 - X axis
1 - Y axis
2 - Z axis
3 - Z axis slow mode

Setting any of the bits to 1 enables ballistic acceleration for this individual axis.

Setting the bit for the Z axis slow mode reduces the resolution for the optical scroll wheel.

Enabling ballistic acceleration introduces an extra latency of 8 ms for the mouse data. This is not noticeable for the user.

The 8 entries of the ballistics table define trigger levels for the acceleration. If the actual number of mickeys on an axis passes the first trigger value it is doubled, if it passes the second it is tripled and so on. The resulting value will be saturated at 127.

5.15 Matrix status mode

To simplify programming the key matrix KeyWarrior100 can be switched into a mode where it does not generate key codes but rather sends the status of the matrix. This makes it easier to locate a key for programming, the user can press the key to be programmed and the software gets the info which matrix coordinate so use.

Enabling and disabling the mode is done with command 11:

Byte#	Data
0	0x0B - Set Matrix Status Mode
1	0 = disable, 1 = enable
2…3	unused, write 0
4…11	PIN code
12…63	unused, write 0

There is no reply report to acknowledge the command.

While the mode is active no keyboard reports are generated. Instead on each change of the matrix status a report is returned on the config interface containing the raw matrix data:

Byte#	Data
0	0x0B - Matrix Status
1…3	unused, 0
4…51	matrix data, key down = 1
52…63	unused, 0

Do not issue other commands to the config interface while running in the matrix status mode. Reply reports may interfere with the matrix data and vice versa.

5.16 Loading user programmable macros

KeyWarrior100 supports 64 macros that can be changed by the end user. Downloading these macros to the KeyWarrior100 can be done even when the key table is locked or PIN protected. But only the content of the macros is accessible in this way, it is not possible to reassign macros to keys, or change any other configurations.

This allows the keyboard manufacturer to offer their customers an option to do a limited reconfiguration of the keyboard without the risk of changing vital settings that can render the keyboard non functional.

The 64 user programmable macros are held in RAM and can be overwritten at any time. It is possible to store the current configuration in persistent memory, so that it will be available on power up of the keyboard.

End user programmable macros are assigned to keys with codes from the key code page 0x05. Each of the macros can have the same functions as the permanently programmed macros.

There are three commands to handle the end user programmable macros.

Reading a macro from KeyWarrior100 is done by command 0x80:

Byte#	Data
0	0x80 - Read Macro
1	Macro number (0…63)
2…63	unused, write 0

This produces a return report of the following format:

Byte#	Data
0	0x80 - answer to Read Macro
1	Error code, 0 = no error
2…3	unused, 0
4…35	Macro Data
36…63	unused, 0

An error code = 1 means the macro number was out of range

Writing a macro to KeyWarrior100 is done by command 0x81. The change does take effect immediately.

Byte#	Data
0	0x81 - Write Macro
1	Macro number (0…63)
2, 3	unused, write 0
4…35	Macro Data
36…63	unused, 0

This produces a return report of the following format:

Byte#	Data
0	0x81 - answer to Write Macro
1	Error code, 0 = no error
2……63	unused, 0

An error code = 1 means the macro number was out of range.

To copy the current macro configuration to persistent memory, so it is available on the next power up, is done with command 0x82. Bytes 1 and 2 have to have a fixed value in them to reduce the chance that this command is executed in error. This command writes the data to FLASH-ROM, which has a limited number of write cycles.

Byte#	Data
0	0x82 - Store in persistent memory
1	0x08
2	0x15
3…63	unused, 0

This produces a return report of the following format:

Byte#	Data
0	0x82 - answer to Store persistent
1	Error code, 0 = no error
2……63	unused, 0

An error code = 3 means the write to FLASH has failed.

5.17 Setting the dimmable LED outputs

The two DAC and three PWM outputs can be set directly by command 0x83:

Byte#	Data
0	0x83 - Set LEDs
1	Channel
2	Value
3	Lin/Log Flag
4…63	unused, 0

This produces a return report of the following format:

Byte#	Data
0	0x83 - answer to Set LEDs
1	Error code, 0 = no error
2……63	unused, 0

An error code = 1 means the channel number was out of range.

6. Factory default key tables

KeyWarrior100 ships in an empty state, no key table loaded and no configurations are preset. So when connected to a USB port it will show up as a keyboard and mouse but generates no input until it has been configured. It does not even react to the lock LED status, as the lock LEDs are not configured.

6.1 KeyTables

KeyWarrior100 uses a key table with 16 bits for each physical key to turn matrix coordinates into functions.
There are three alternative tables between which can be switched by using the FN-keys. Any of the physical keys can be assigned to perform any of the available functions.
The functions are grouped into pages, identified by the high byte (the one on the higher address in the table) of the 16 bits for each physical key.
KeyWarrior100 supports the following pages:

0x00 - Standard keys, modifiers and FN keys
0x01 - Media and application controls
0x02 - Manufacturer macros
0x03 - Mouse
0x04 - Dimmable LED control
0x05 - End user macros
0xE0…0xE7 - Modifier plus standard key combo

A more detailed description including extensive tables for all codes can be found in the "KeyWarrior II Scancode Tables" document.

6.2 Standard keys

KeyWarrior100 allows the use of all standard keys as defined by HID usage codes in the range of 0x04 to 0xDD. The usage codes are just entered into the table as they are, i.e. 0x0004 is the "a" key on ISO keyboards.
Which actual character is generated from a usage code depends on the operating system and its keyboard settings.

6.3 Modifier keys

Modifier keys are the keys that change the meaning of another key, they are Shift, Control, Alt and Command (GUI).
The normal modifier keys are coded with their standard usage code, i.e. 0x00E2 is the left Alt key.
In additon the codes 0x00E8 to 0x00EF produce a locking version of the modifier keys. So if 0x00E8 is programmed on a key, pressing this key once will result in the left Control key becoming active and staying active until the key is pressed again or another key with the normal left Control function is pressed.
To allow single finger operation a "sticky" version of the modifiers is available also. 0x00F0 to 0x00F7 do activate the modifier when the key is pressed and hold it active until any standard key has been pressed. Essentially this is a single shot locking modifier.

6.4 FN keys

To switch between the three alternative keyboard tables two FN levels are available which can be accessed in a number of ways.
0x00FF and 0x00FE are the normal FN1 and FN2 keys. While FN1 is active the second key table is used and the third table while FN2 is active. Pressing both FN keys at the same time will result in the third table being used.
Like with the modifiers there are also locking and sticky versions of the FN keys.
0x00FD is locking FN1 and 0x00FC locking FN2. They stay active until the locking key is pressed again or the respective non locking FN key is pressed.
0x00FB is sticky FN1 and 0x00FA is sticky FN2. They stay active until the next standard key is pressed.
0x00F9 is a counting FN key. Every time it is pressed it changes to the next key table, after the third it wraps around to the first.
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.

6.5 Media Control and Application Keys

KeyWarrior100 supports a total of 32 media control and application keys. If they actually work depends on the operating system of the target platform. Not all codes are available on all systems.

The following codes are used to generate these keys (second code is the USB usage code):

0x0100	0xB0	Play
0x0101	0xB1	Pause
0x0102	0xB2	Record
0x0103	0xB3	Fast Forward
0x0104	0xB4	Rewind
0x0105	0xB5	Scan Next Track
0x0106	0xB6	Scan Previous Track
0x0107	0xB7	Stop
0x0108	0xB8	Eject
0x0109	0xB9	Random Play
0x010A	0xCC	Stop/Eject
0x010B	0xCD	Play/Pause
0x010C	0xE2	Mute
0x010D	0xE9	Volume Increment
0x010E	0xEA	Volume Decrement
0x010F	0x0184	Launch Word Processor
0x0110	0x0185	Launch Text Editor
0x0111	0x0186	Launch Spreadsheet
0x0112	0x0187	Launch Graphics Editor
0x0113	0x0188	Launch Presentation App
0x0114	0x0189	Launch Database App
0x0115	0x018A	Launch Emailer
0x0116	0x018B	Launch Newsreader
0x0117	0x018C	Launch Voicemail
0x0118	0x018D	Launch Address Book
0x0119	0x018E	Launch Calendar
0x011A	0x0192	Launch Calculator
0x011B	0x0194	Launch Local Browser
0x011C	0x0196	Launch Internet Browser
0x011D	0x01A0	Command Line
0x011E	0x01A6	Help Center
0x011F	0x021F	Find

6.6 Macros

KeyWarrior100 has an advanced macro capability essentially combining all the macro features of the old KeyWarrior family variants.

There are two groups of macros supported by KeyWarrior100. One group is intended to be programmed by the keyboard manufacturer and as part of the keyboard configuration can be protected against end user tampering. The second group is intended for programming by the end user. It is separated from the general configuration of the functions.

A manufacturer macro is programmed onto a key by 0x02xx, an end user macro by 0x05xx."xx" is the number of the macro to be used. Manufacturer macros are numbered 0 to 175 and end user macros 0 to 63. Those are different sets of macros, so the total available macros are 240.

Each macro is 32 bytes long. The first byte of the macro data selects the mode of the macro.

There are gour modes for a macro: Static, typing, cell and cell dual mode. In static mode all key codes of the macro are activated and released together. Typing mode will make and break each of the key codes, to allow repeating characters. Cell mode is similar to the entry mode for characters on cell phones. Every time a key is pressed it generates the next key code in a sequence, overwriting the last one. This is a macro mode interesting for small keyboard layouts to generate multiple characters from a single key.

6.6.1 Static macros

A static macro is defined by a 0x00 in the first byte of the macro. The following 31 bytes contain a 8 bit representation of the page 0x00 key codes. So each of these 31 codes may contain any standard key, modifier, or FN key. A 0x00 code denotes the end of the macro, any further bytes behind a 0x00 until the end of the macro are ignored.

All valid codes in a static macro are processed and send in a single report. So there is a limit to the usable codes in a static macro. No more than all modifier keys plus six standard keys may be contained in a single static macro.

Be aware that the results of having FN keys or locking/sticky modifiers in a static macro can be very complex.

6.6.2 Typing macros

A typing macro starts with 0x01 in the first byte. The following 31 bytes contain a 8 bit representation of the page 0x00 key codes. But only standard keys and non-locking/non-sticky modifiers are allowed for typing macros. A 0x00 code denotes the end of the macro, any further bytes behind a 0x00 until the end of the macro are ignored.

Any standard key in a typing macro is send and then deactivated again immediately. Modifiers work a bit different. The codes 0xE0 to 0xE7 are the standard modifiers but the modifier will stay active until its break code is in the macro. Break codes for the modifiers are 0xE8 to 0xEF.

All modifiers will be released at the release of the macro key, so they do not need to be released by the end of the macro.

6.6.3 Cell macros

A cell macro is defined by a 0x02 in the first byte. The second byte holds a timeout value in 100 ms. If the timeout expires the last character generated will not be overwritten, i.e. allowing to type the same character again. A zero for the timeout will be handeled as immediate timeout.

The remaining 30 bytes can hold an 8 bit keycode each. When the macro key gets pressed the first keycode is send followed by a left arrow, so the cursor will be under the just generated character. If no key is pressed within the timeout then a right arrow will be send to move the cursor behind the character. If a different cell macro key is pressed before the timeout then also a right arrow will be send first.

Pressing the same macro key again before the timeout will produce a delete forward to be send to remove the last character and then the next keycode will be send followed by a left arrow. When the next keycode is 0x00 the macro wraps around to the first code again.

6.6.4 Cell dual macros

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 in each cell dual macro.

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.

6.7 Mouse function

Page 0x03 contains the codes for the mouse function.

It is possible to assign left, right, up, down, and up to 8 mouse buttons to any keys. The mouse cursor movement is controlled with an acceleration curve, making it possible to do precise small movements, as well as long movements across the screen.

The key codes for the mosue buttons are also used for the other mouse options.

The codes are as follows:

0x0300	Right move
0x0301	Left move
0x0302	Down move
0x0303	Up move
0x0305	Down/right move *
0x0306	Down/left move *
0x0309	Up/right move *
0x030A	Up/left move *
0x0310	Left Button
0x0311	Right Button
0x0312	Center Button
0x0313	Button 4
0x0314	Button 5
0x0315	Button 6
0x0316	Button 7
0x0317	Button 8

*) Function is available since firmware version V1.0.1.A earlier versions ignore this code

6.8 LED dimming keys

Page 0x04 is used for the LED dimming keys, that control the DAC and PWM outputs.
For KeyWarrior100 there are five sets of key codes, each controlling one of the outputs. The outputs are numbered as follows:

Channel0	DAC0
Channel1	DAC1
Channel2	PWM0
Channel3	PWM1
Channel4	PWM2

The following key codes are available:

0x04n0	Off
0x04n1	Max
0x04n2	Step up
0x04n3	Step down
0x04n4	Step up and wrap
0x04n5	Dim up
0x04n6	Dim down
0x04n7	Dim up/down
0x04n8	Toggle *

*) available on Firmware V1.0.1.B and later

Details can be found in the "KeyWarrior II Scancode Tables" document.

6.9 Modifier plus key

Pages 0xE0 to 0xE7 are a combination of a modifier plus one key. The high byte contains the standard code for the modifier, 0xE0 to 0xE7. In the low byte any of the 8 bit key codes may be used, including modifiers and FN keys.
The two keys are then send together.
This is a simple way to generate special characters from any key matrix position without having to use a complete macro.

7. Generic I/O

KeyWarrior100 contains a generic I/O function the works similar to the IO-Warrior. Basically a small IO-Warrior is integrated into the KeyWarrior100.
The two interfaces for the I/O function register as generic HID class. This makes it easy to access them on all operating systems. There are no high level drivers that grab these functions, so they can be accessed from application level.

7.1 Communication with generic I/O

KeyWarrior100 has 4 USB endpoints for the I/O communication. Endpoints are like virtual communication ports into or out of the device.
Two endpoints each are assigned to an interface. Interfaces are like virtual devices or subsystems within a device.
Interface 4 is used to talk to the I/O pins direct, interface 5 for the special mode functions.
(I2C had a separate interface until V1.0.1.6 but a bug in the Windows driver for the Intel Extensible Host Controller prevents devices with 7 interfaces to work behind a hub)

7.2 Input behaviour

KeyWarrior100 scans the status of all generic I/O pins once every millisecond. If it detects a change from the last status a new report via interface 4 is issued. Pins which are currently used by a special mode function are not checked and are always reported as being high.

7.3 Using pins as inputs or outputs

All I/O pins can be used as input or output pins.
Basically all pins act as inputs all of the time. When receiving an input report from KeyWarrior100 you always get the current status on the pins.
Writing a 0 as output value to any pin causes it to drive the pin low with an open drain driver. Usually this will result in this pin being read as a zero input as well, unless so much current has to be drained by the pin that the voltage at it gets above the threshold level.
Writing a 1 to a pin causes the open drain driver to be turned off. The pin will be pulled high by an internal pull up resistor. Now the pin acts either as an output with a high level, or can be used as an input.

7.4 Special mode I/O

To enable KeyWarrior100 to talk to devices that have more complex demands there is the set of special mode functions. When any of these functions is enabled a couple of the generic I/O pins of KeyWarrior100 turn into special function pins.
Talking to the special mode functions is handeled via the USB interface 5, which is also configured as generic HID.
To talk to these functions and to handle various requests ReportIDs are used which enable multiple functions to use the same endpoint. All reports to and from special mode interfaces are always 64 bytes long, including the ReportID.
The following chapters describe the individual special mode functions.

7.4.1 Fast IIC special mode function

The fast I2C function of KeyWarrior100 offers very high performance.
Fast mode and fast mode plus are supported, allowing data rates up to 1 MBit/s and a throughput of > 60.000 bytes/s.
The I2C function is enabled and disabled by sending a report with the following structure with ReportID=1 to interface 5:

ReportID	1	2	3	4	…	62	63
$01 out	enable	flags	timeout	$00	$00	$00	$00

enable=$01 enables the I2C function, $00 disables it. Other values are reserved for future use.
Upon enabling I2C the SDA and SCL pins are pulled high and are no longer under control of SetReport requests to interface 4. Disabling I2C does return the pins under control of interface 4 and pulls them high initially.
flags selects the speed of the I2C interface and sets the drivers on the SCL/SDA pins accordingly.
The values for flags are as follows:
0 - 100 kbit/s
1 - 400 kbit/s
2 - 50 kbit/s
3 - 10 kbit/s
4 - 1000 kbit/s

timeout is not yet implemented.

A write request to the I2C is sent with ReportID=2 and has the following format:

ReportID	1	2	3	4	…	62	63
$02 out	flags	data	data	data	data	data	data

flags contains the following bits:
7 - Generate Start
6 - Generate Stop
5 - data count MSB
4 - data count
3 - data count
2 - data count
1 - data count
0 - data count LSB

If bit 7 - "Generate Start" is set a start signal (SDA falling edge while SCL is high) is generated on the I2C prior to sending out the first data byte.
Bit 6 - "Generate Stop" causes a stop signal (SDA goes high while SCL is high) to be generated after sending the last valid data byte of this report.
"data count" gives the number of valid data bytes in the report. The number may range from one to 62, other values cause an error to be returned.
To do write transactions that are longer than 62 bytes, send the first report with just the "Generate Start" bit set, then send additional reports with neither bit 6 or 7 set until the report with the last bytes is send which has the "Generate Stop" bit set.

Any write transactions are acknowledged by a report via interrupt-in endpoint 7:

ReportID	1	2	3	4	…	62	63
$02 in	flags	code	$00	$00	$00	$00	$00

flags contains the following bits:
7 - Error bit, 1=error
6 - Arb Loss, 1 = lost arbitration
5 - data count MSB
4 - data count
3 - data count
2 - data count
1 - data count
0 - data count LSB

"data count" indicates the last byte that was successfully transfered and acknowledged by the slave (if any). An error is indicated when the slave does not acknowledge a transfer.

"code" contains an additional error code to indicate what happend. The following values are defined:
0 - no special code
1 - wrong number of bytes requested
2 - transaction without Start requested
3 - NACK received
4 - Bus Error

Reading data off the I2C is initiated with a ReportID=3. The initiating report has the following format:

ReportID	1	2	3	4	…	62	63
$03 out	count	addr	Mcount	$00	$00	$00	$00

"addr" holds the slave address to be send to the I2C.
"count" is the number of bytes that should be read off the I2C after sending the command byte, "Mcount" is the most significant byte for the count. The maximum number of bytes that may be requested at once is 65535.
A start signal is automatically generated before sending the command byte and a stop is generated after the last data byte is received.
Data is returned in input reports with ReportID=3 (which is different from the output ReportID=3 used to initiate the read transaction) via endpoint 3. If more than 62 bytes are requested the data is returned in multiple reports.

ReportID	1	2	3	4	…	62	63
$03 in	flags	data	data	data	data	data	data

flags contains the following bits:
7 - error, set if slave does not ack command byte
6 - unused, zero
5 - data count MSB
4 - data count
3 - data count
2 - data count
1 - data count
0 - data count LSB

Should the I2C slave fail to acknowledge the command byte the error flag will be set and the transaction aborted. I2C does not have an error condition during the actual reading of data after the command byte was sent.

7.4.2 LCD special mode function

The LCD special mode function supports display modules that are compatible with the HD44780 controller and several graphic display controllers that use a compatible interface. This controller is made by Hitachi and has set the de-facto standard for alphanumeric LCD modules. There are also OLED and vacuum fluorescence displays that use this type of interface.
The display modules come in various configurations with up to 80 characters total in any kind of arrangement from single line to four lines. Displays with more than 80 characters typically use more than one HD44780. KeyWarrior100 does not directly support modules with more than a single HD44780, some additional hardware may be required for this.
It is recommended to read the HD44780 manual for using the LCD function.

The LCD function is enabled by sending an output report with ID 4 to the USB interface 5:

ReportID	1	2	3	4	…	62	63
$04 out	enable	$00	$00	$00	$00	$00	$00

enable = $00 disables the LCD function. enable = $01 enables the LCD function, other values are reserved.
Upon enabling the LCD function the pins are put under control of the LCD function and can no longer be controlled by interface 4.
The /On pin is pulled low when the LCD function is enabled, it will go high when the KeyWarrior100 enters suspend state.
To write data to the connected LCD module an output report with ReportID=5 is written with the following format:

ReportID	1	2	3	4	…	62	63
$05 out	flags	data	data	data	data	data	data

"flags" contains the following bits:
7 - RS, Register Select bit
6 - unused, zero
5 - data count MSB
4 - data count
3 - data count
2 - data count
1 - data count
0 - data count LSB

The status of the RS bit is used to set the RS line to the LCD module. This allows access to the Instruction register (RS=0) or Data Register (RS= 1) of the LCD module.
With "data count" the number of bytes to be written is specified. KeyWarrior100 will write up to 62 data bytes to the register specified by the RS bit. The Busy bit of the LCD module is automatically checked and data written only when the LCD module is ready to accept it. There is a 16 ms timeout when waiting for the Busy bit, if this time has passed the transaction is cancled, no special feedback is provided.

To read data from the LCD module an output report with ReportID=6 is sent to interface 1:

ReportID	1	2	3	4	…	62	63
$06 out	flags	$00	$00	$00	$00	$00	$00

"flags" contains the following bits:
7 - RS, Register Select bit
6 - unused, zero
5 - data count MSB
4 - data count
3 - data count
2 - data count
1 - data count
0 - data count LSB

"RS" specifies which register is to be accessed. Data count sets the number of bytes to be read off the LCD (will be ignored if RS=0, only a single read will be done then).

Up to 62 bytes can be read with one request. The data read from the LCD module is returned in input reports with ID 6:

ReportID	1	2	3	4	…	62	63
$06 in	count	data	data	data	data	data	data

"count" specifies the number of bytes returned in this report.

7.4.3 LED matrix function

KeyWarrior100 has the capability to drive a matrix of up to 8x64 LEDs with the aid of a few simple external driver chips.

To enable the LED matrix function a report with ID $14 is sent to interface 5:

ReportID	1	2	3	4	...	62	63
$14 out	enable	$00	$00	$00	$00	$00	$00

enable = $01 enables the LED matrix function, enable = $00 disables it again.

Data to be displayed in the matrix is written in two blocks of 32 bytes:

ReportID	1	2	...	33	34	...	63
$15 out	block	data0	datan	data31	$00	$00	$00

"block" = 0 writes to the first four lines, "block" = 1 to the second four lines.

A "1" bit indicates a "on" LED.

For more information on how to implement the driver circuit for the LE matrix, please refer to the applicaion note AN10 - Driving LED matrix.

7.4.4 Digital LED function

KeyWarrior100 has the capability to drive up to 500 digital LEDs of the type that uses a single digital data line, like the WS2812 or WS2812B.

To enable the digital LED function a report with ID $28 is sent to interface 1:

ReportID	1	2	3	4	5	…	63
$28 out	enable	0Cyc	0Pha	1Cyc	1Pha	$00	$00

"enable" = $01 enables the digital LED function, "enable" = $00 disables it again.

The LED matrix function can be disturbed by the digital LED function. Writing the data to the LEDs can interrupt the matrix update if many LEDs are connected.

To handle different variants of the digital LEDs KW100 requires parameters for the timing of the serial data transfer.

The data signal is generated from a 48 MHz master clock, allowing a timing resolution of 20.83 ns steps. All timining parameters (X) will generate pulse or cycle times of t = (X+1)*20.83 ns.

"0Cyc" is the length of a bitcell signaling a zero bit.

"0Pha" is the length of the high phase for a zero bit.

"1Cyc" is the length of a bitcell signaling a one bit.

"1Pha" is the length of the high phase for a one bit.

Typical timings:

LED type	0Cyc	0Pha	1Cyc	1Pha
SK2812	$39	$0E	$39	$1C
LC8812	$2B	$0E	$48	$2B
WS2812B/SK2812	$3C	$13	$3C	$26
WS2812	$37	$10	$3E	$21
WS2813	$45	$10	$45	$34

Data to be send to the LEDs is transfered in up to 25 blocks containing data for 20 LEDs each. The data is buffered internally in the KW100 and individual blocks can be overwritten to reduce USB traffic when updating.

ReportID	1	2, 3	4	…	63
$29 out	block	count	data 0	data n	data 59

"block" is the block number in the buffer to which the data is written, 0 to 24 are valid.

If the high bit of "block" is set the data is shipped out to the LEDs after the block has been copied to the buffer.

"count" is ignored until the high bit of "block" indicates that the data is to be transfered to the LEDs. Then it specifies the number of LEDs to write. 1 to 500 is the valid range, byte 2 is LSB.

The data is then shipped to the LEDs with the timing specified on enabling the function.

7.4.5 Reading special mode function status

To get the information which special modes are currently in use, send a report with ReportID=$FE to interface 5:

ReportID	1	2	3	4	...	62	63
$FE out	$00	$00	$00	$00	$00	$00	$00

This will immediately return an input report with ID $FE that contains flags for the special mode functions. Non zero means the function is active:

RepID	1	2	3	4	..	9	…	62	63
$FE in	I2C	LCD	0	LEDmat	0	WSLED	0	0	0

7.4.6 Getting current pin status

Due to the way Windows implements HID support KeyWarrior100 is unable to continuously send its status. HID class devices do have a function that allows the host to set the rate at which reports should be repeated if there is no change to the data. Windows does set this rate to zero for KeyWarrior, which means KeyWarrior may send data only if there are changes.

To be able to get the current status from KeyWarrior it does support a special mode function that always returns the current status of all pins.

To get the port status just send a report with ID $FF to interface 5:

ReportID	1	2	3	4	…	62	63
$FF out	$00	$00	$00	$00	$00	$00	$00

This will result in the current pin status to be returned immediately in an input report with ID $FF with the following format:

ReportID	1	2	3	4	…	62	63
$FF in	Port0	Port1	$00	$00	$00	$00	$00

8. DC characteristics

	Parameter	Min	Max	Units	Remarks
V_{dd}	Operating voltage	2.0	3.6	V	typ. 3.3 V
I_{dd}	Operating supply current		30	mA	
I_{sb}	Suspend mode current		350	μA	internally active
I_{ol}	Sink current on port pins		25	mA	max. combined all pins 80 mA
V_{ol8}	Output low voltage		0.4	V	I = 8 mA
V_{oh8}	Output high voltage	V_{ddi}-0.4		V	I = 8 mA
V_{ol20}	Output low voltage		1.3	V	I =20 mA
V_{oh20}	Output high voltage	V_{ddi}-1.3		V	I =20 mA
R_{up}	Pull up/down resistors	25	55	kΩ	typ. 40 kΩ
V_{ith}	Input threshold voltage	0.7 x V_{ddi}		V	

8.1 AC characteristics

	Parameter	Min	Max	Units	Remarks
	Keyboard Matrix Scan Timing				
t_{scan}	Scanning interval	4		ms	
t_{scansu}	Matrix drive to read setup time	typ. 40		μs	
$t_{debounce}$	Debounce time	3x t_{scan}		ms	

8.2 Absolute maximum ratings

Storage Temperature	-65°C to +150°C
Ambient Temperature operating	-40°C to +85°C
Supply voltage on Vdd, Vdda, Vddi relative to Vss	-0.3 V to +4 V
DC input voltage	-0.3 V to +4 V
DC input voltage (for 5 V tolerant pins)	-0.3 V to Vddi+4 V
Maximum current into all ports	80 mA
Power Dissipation	max. 476 mW
Static discharge voltage	>2000 V
Latch-up current	>200 mA

9. Ordering Information

Partname	Order Code	Description	Package
KeyWarrior100	KW100-LF100	Standard part	LQFP100
KeyWarrior100-MOD	KW100-MOD	PCB module with mini USB plug and voltage regulator on board	module
KeyWarrior100EVAL	KW100EVAL	Evaluation kit	module

9.1 Shipping info

LQFP100 chips come in trays of 90 units each.
Modules and evaluation kits come in antistatic bags as single units.

9.2 USB VendorID and ProductID

By default all KeyWarrior chips are shipped with the USB VendorID of Code Mercenaries (0x7C0 or decimal 1984).
The ProductID will be assigned by Code Mercenaries.
On request chips can be equipped with the customers VendorID and ProductID. VendorIDs can be obtained from the USB Implementers Forum <www.usb.org>
Such custom chips are subject to additional costs and minimum order volumes.
The ProductID for the standard KeyWarrior100 chips are:

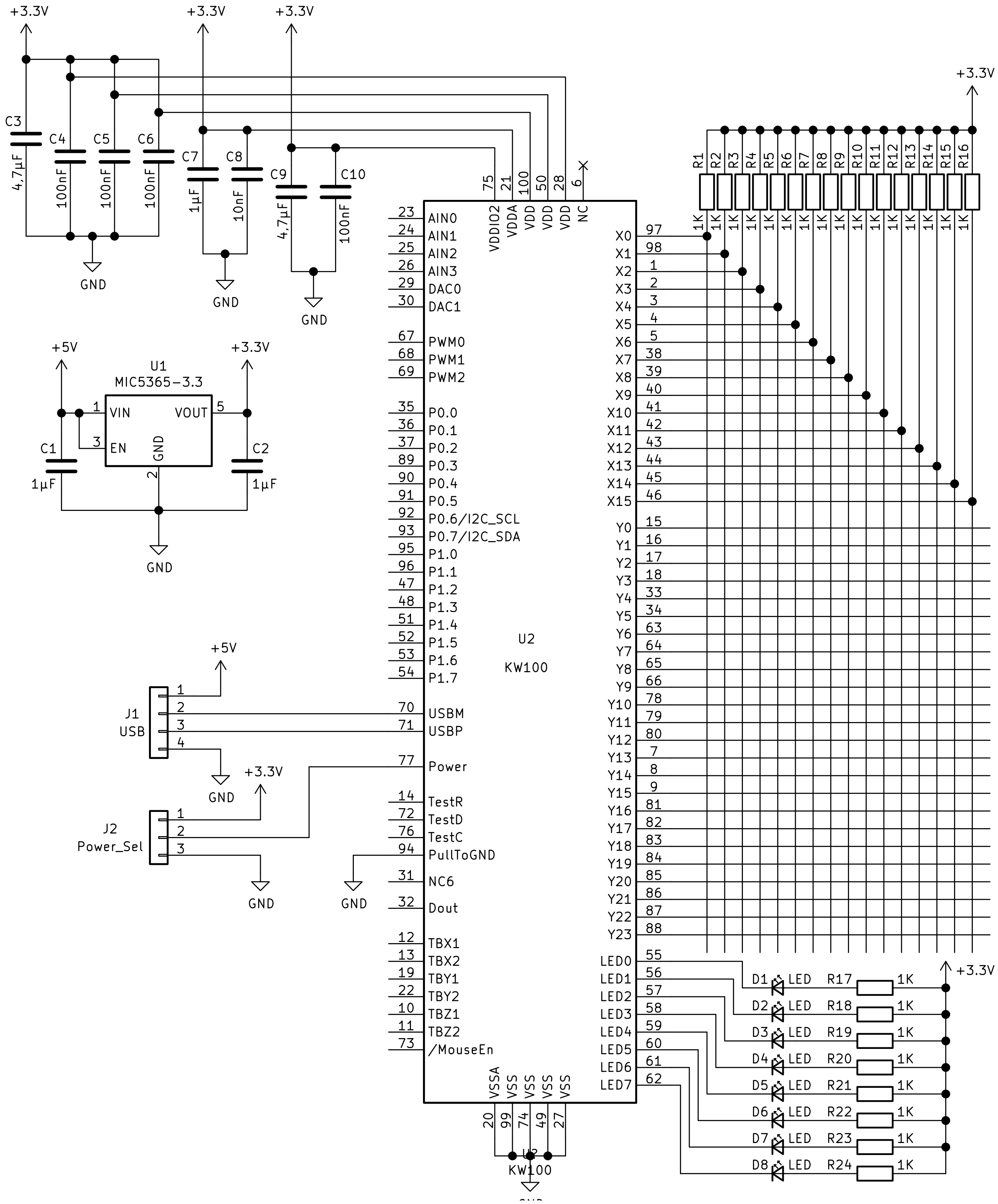
KeyWarrior100	0x0181

ProductIDs are independent of the package type.

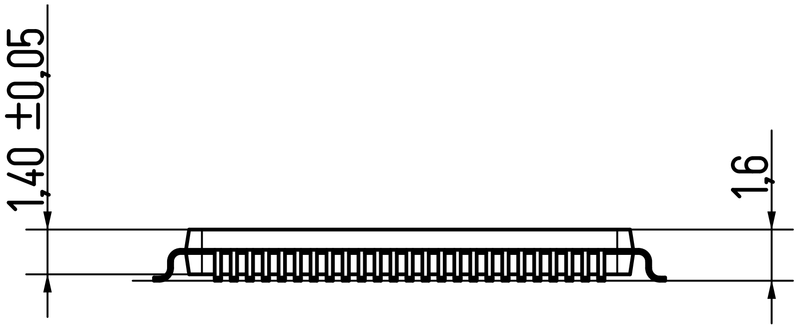
9.3 Serial numbers

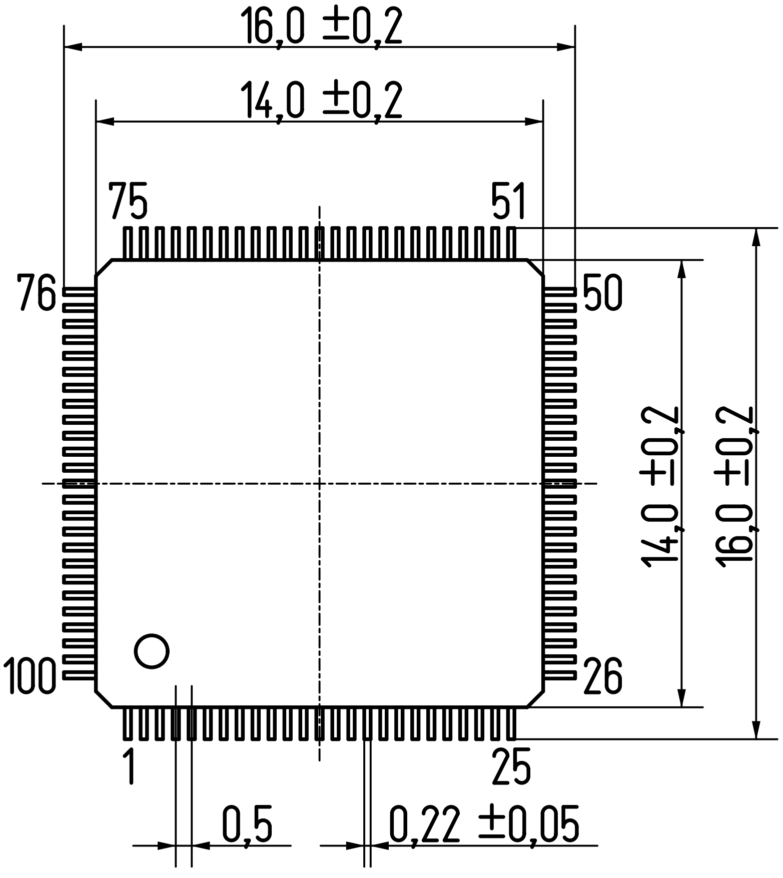
The serial number of KeyWarrior100 is not a USB standard serial number. It is accessible only via the proprietary commands defined for reading the device data of KeyWarrior100.
The serial numbers are factory assigned and can neither be changed by the customer nor ordered specifically.

10. Typical application for KeyWarrior100



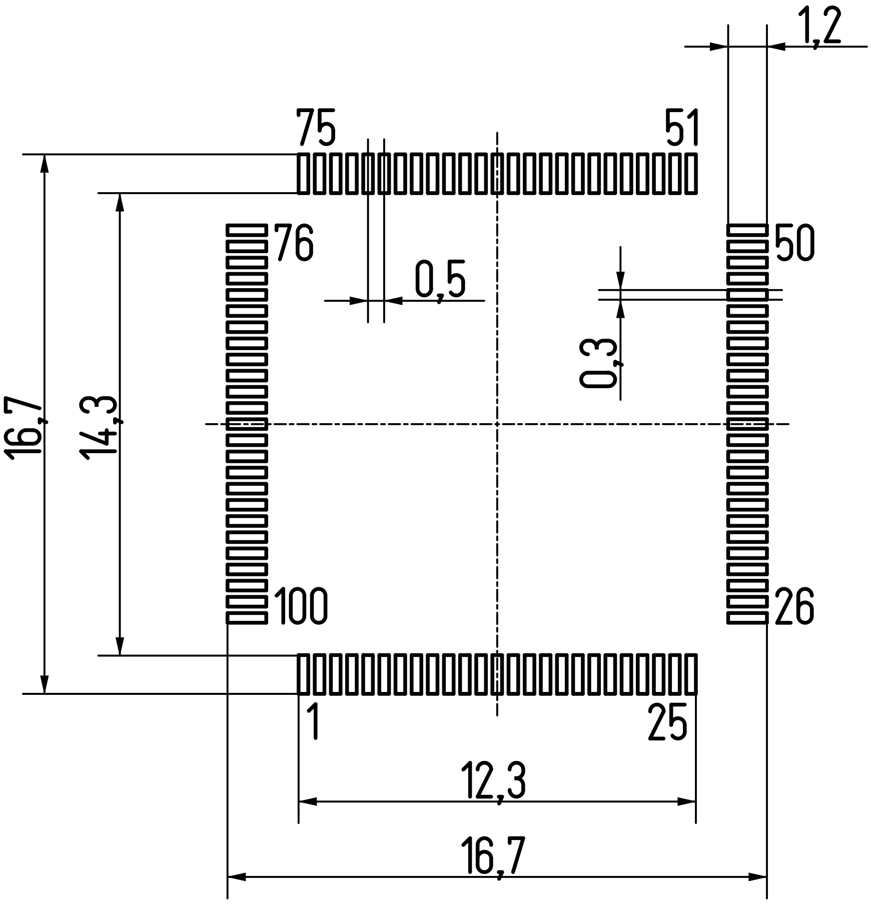
11. Package dimensions
100 Pin LQFP - 16 x 16 mm outer contour 0.5 mm pitch





Package thickness: 1.4 mm ±0.05 mm
All dimensions: mm

11.1 Recommended footprint



All dimensions: mm

12. ESD considerations

KeyWarrior has an internal ESD protection to withstand discharges of more than 2000V without permanent damage. However ESD may disrupt normal operation of the chip and cause it to exhibit erratic behaviour.

For the typical office environment the 2000V protection is normally sufficient. Though for industrial use additional measures may be necessary.

When adding ESD protection to the signals special care must be taken on the USB signal lines. The USB has very low tolerance for additional resistance or capacitance introduced on the USB differential signals.

12.1 EMC considerations

KeyWarrior uses relatively low power levels and so it causes few EMC problems.

To avoid any EMC problems the following rules should followed:

- Put the 100 nF ceramic capacitors right next to the power supply pins of the chip and make sure the PCB traces between the chips power pins and the capacitor are as short as possible.
- Run the power supply lines first to the capacitor, then to the chip.
- Make the matrix lines only as long as absolutly necessary.
- Keep the two USB signal lines close to each other, route no other signal between them. USB uses differential signalling so the best signal quality with lowest RF emission is achieved by putting these lines very close to each other.

13. Revision history

The current shipping version of KeyWarrior100 is V1.0.1.F

V1.0.1.F - Added Slow mde for four quadrant mouse sensor to handle very sensitive force sensors.

V1.0.1.B - Added diagonal directions for the key mouse function and toggle function for the DAC and PWM outputs.

V1.0.1.8-V1.0.1.A were not released for KW100.

V1.0.1.7 - Fixed a problem with USB that could lead to bidirectional USB interfaces to lock up the In direction. This problem was observed only with JW28A12L under very special timing conditions but the patch was applied to all chips with bidirectional interfaces to avoid this potential problem.

V1.0.1.6 - Moved I2C function to the special mode interface. A bug in the driver for the Intel Extensible Host Controller prevents devices with 7 interfaces to work if connected behind a hub.

V1.0.1.5 - Fixed a race condition in the USB protocol, this could prevent proper operation

V1.0.1.4 - Added intelligent LED function

V1.0.1.3 - Key tables did still contain test data. Now they are empty (0xFF) as documented.

V1.0.1.2 - not released

V1.0.1.1 - Initial release version

13.1 Document revision history

V1.0.0.6 - Added version info for V1.0.0.F
V1.0.0.5 - Added version info for V1.0.0.B
V1.0.0.4 - Added version info for V1.0.1.7
V1.0.0.3 - Added description of intelligent LED function and bugfixes
V1.0.0.2 - Pin numbers for TestD and /MouseEn were switched in the schematic in section 10.
V1.0.0.1 - Version information updated.

14. RoHS compatibility

KeyWarrior100 conforms to the requirements that are necessary to use it in a RoHS compliant device.

Legal stuff

The information contained herein is subject to change without notice. Code Mercenaries makes no claims as to the completeness or correctness of the information contained in this document.

Code Mercenaries assumes no responsibility for the use of any circuitry other than circuitry embodied in a Code Mercenaries product. Nor does it convey or imply any license under patent or other rights.

Code Mercenaries products may not be used in any medical apparatus or other technical products that are critical for the functioning of lifesaving or supporting systems. We define these systems as such that in the case of failure may lead to the death or injury of a person. Incorporation in such a system requires the explicit written permission of the CEO of Code Mercenaries.

Trademarks used in this document are properties of their respective owners.

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