

– preliminary –

USB library

Software-Specification

Version 4.0
04.06.2010



Mechaless Systems GmbH
Albert-Nestler-Str. 10
76131 Karlsruhe
Germany

Contents

1	Foreword	3
2	Introduction	4
2.1	Overview.....	4
3	Working with the USB library	5
3.1	Connecting the MAX3420E with the HALIOS® IC.....	5
3.2	Integrating the USB library into the application project.....	6
4	USB library reference	7
4.1	Integration the USB library into an user application	7
4.1.1	Polling	7
4.1.2	Interrupt handling	8

1 Foreword

This document describes the contents of the E909.05A USB library in detail and shows an example how to use this library. Please read this guide completely and carefully. Further information can be found on our website: <http://www.mechaless.com>

If you have any questions or comments regarding this product or documentation, please contact:

Mechaless Systems GmbH
Albert-Nestler-Str. 10
76131 Karlsruhe
Germany

<http://www.mechaless.com>
info@mechaless.com

Phone.: +49 (0)721 / 62698-0
Fax: +49 (0)721 / 62698-11

Additional information can be found in the following documents:

- HALIOS® Basics
- HACo Operating Instructions
- Data specification sheet regarding IC used (e.g. E909.05 or E909.06)
- E909.05 or E909.06 firmware library documentation
- HALIOS firmware specification

2 Introduction

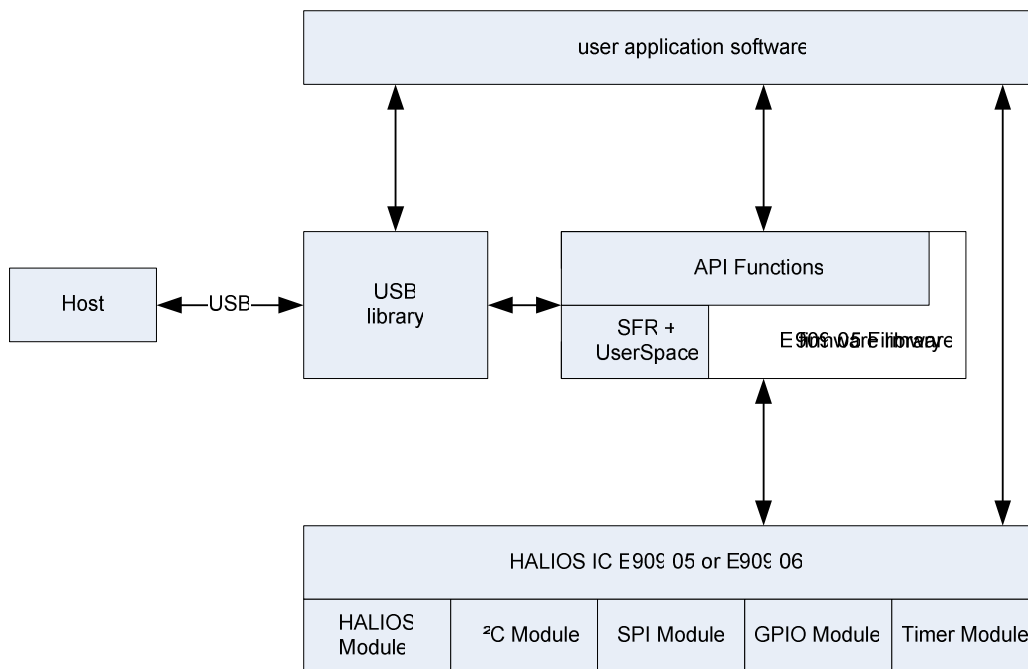
This library uses the SPI peripheral of the E909.05 or E909.06 to establish an USB communication with MAX3420E SPI-USB bridge.

2.1 Overview

This library depends on the firmware library, so that an application which uses the USB library must be linked against the libraries lib_firmware05 or lib_firmware06 for and lib_usb.

The USB library enables the application engineer to communicate via USB with the SFR area and the User Space of the HALIOS[®] firmware (see firmware specification for detailed description).

In "Picture 1" a layer view of an application is shown which uses the USB library and the HALIOS[®] firmware.



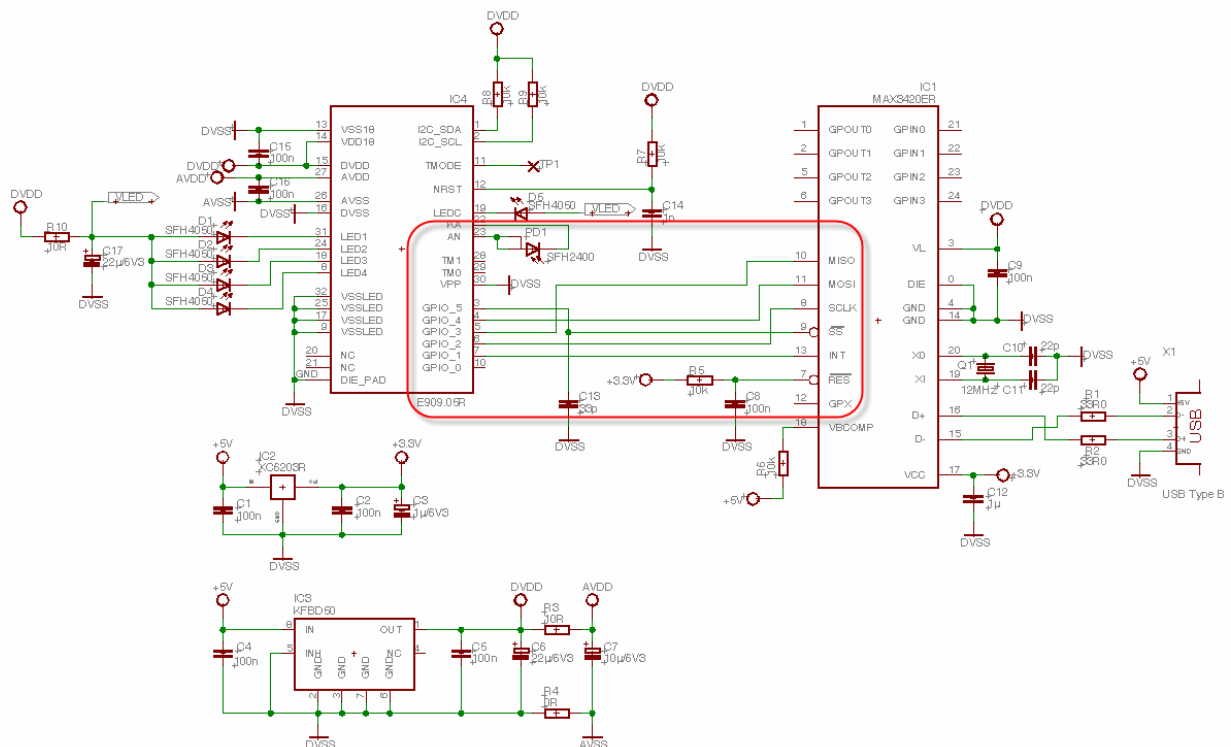
Picture 1

3 Working with the USB library

Integration of the firmware API into the system requires that the system designer implements the components discussed below, together with a proper handling of their implementation aspects.

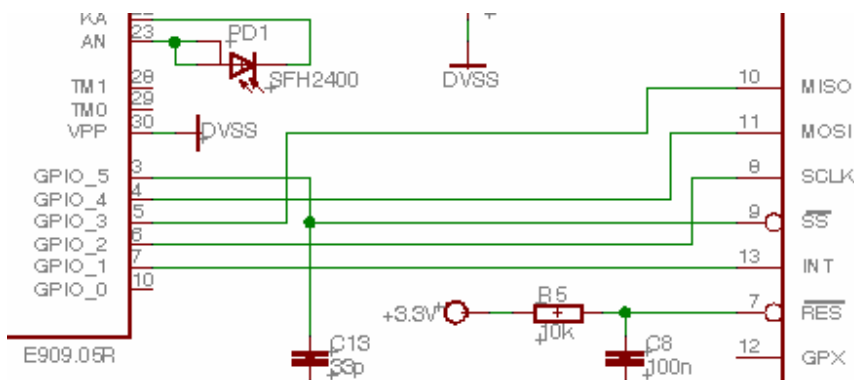
3.1 Connecting the MAX3420E with the HALIOS® IC

The circuit in “Picture 2” shows how to connect the MAX3420E USB-SPI bridge with the HALIOS® IC.



Picture 2

The part of the circuit which is marked with the red border is zoomed out in “Picture 3”.



Picture 3

In "Picture 3" is apparent that the GPIOs which are used for USB communication are not available for the user application.

3.2 Integrating the USB library into the application project

Any application software for the HALIOS[®] IC E909.05 or E909.06 which is linked against the firmware library and the USB library has to include the following files (distributed by Mechaless Systems GmbH) into the IAR project:

Every application software for the E909.05A has to include the following header file (distributed by Mechaless Systems GmbH):

For E909.05 or E909.06:

```
firmware.h      (hardware specification)
usb.h           (header file of the USB library)
```

Firmware libraries for IAR:

```
lib_firmware05.r43 or lib_firmware06.r43  (the firmware library)
lib_callback.r43                          (fill callback functions if user don't define)
lib_usb.r43                               (only needed for USB communication with HACo)
```

Firmware libraries for GCC:

```
lib_firmware05.a or lib_firmware06.a      (the firmware library)
lib_callback.a                            (fill callback functions if user don't define)
lib_usb.a                                  (only needed for USB communication with HACo)
```

To test and for development you can use the GCC IDE. The recommended C-IDE for further software development is the IAR Embedded Workbench.

4 USB library reference

The USB library configures the SPI module of the HALIOS® in the way that the USB library is able to configure and communicate with the MAX3420E SPI-USB bridge from Maxim. Furthermore a protocol is implemented and the firmware registers itself as a HID device in full speed mode so that the HALIOS® configuration tool (HACo) is able to communicate with the E909.05 or E909.06.

4.1 Integration the USB library into an user application

The USB library is designed as simple as possible for the application developer. The library therefore consists only of two functions:

- ▶ `usbInitialize()`
- ▶ `usbHacoHandleIrqs()`

With these two functions the whole USB communication is managed on the embedded side. To initialize the SPI module of the HALIOS® IC and to configure the MAX3420E the function

```
usbInitialize(USB_PART, calltime)
```

has to be called while the initializing procedure of the E909.05 or E909.06.

The parameter `USB_PART` can be `USB_PART_ON` or `USB_PART_OFF`. If you want use the GPIO pins at the Maxim IC during initialize (e.g. autocalibration) you can use the `usbInitialize(USB_PART_OFF, 8)` than the SPI is initialized and the USB part is stopped.

After the command `usbInitialize(USB_PART_ON, 8)` the `usbHandleIrq()` should call just in time when the INT pin (PIN 1) at E909.05 goes to low.

The parameter call time initialize the USB-Master how often to call values from E909.05. Possible values are 4, 8, 16, 32 in ms. This is a guaranteed time, master can call the values faster than this timing.

The command `usbHacoHandleIrqs ()` does all USB tasks:

- ▶ register the HALIOS® IC as a HID device at the PC,
- ▶ register the HALIOS® IC as a full speed device,
- ▶ register the HALIOS® IC in interrupt mode with an interrupt time of 8 milliseconds,
- ▶ handle each USB HID interrupt generated from the PC,
- ▶ communication with the HACo software.

4.1.1 Polling

In the main loop of the user application the following code has to be executed:

```
if (gui_doUsb == 1)
{
    /**
     * If USB-request occurred during a measurement
     * clear the wakeupEnd flag
     */
    g_status0.wakeupEnd = 0;

    gui_doUsb = 0;

    /** Do transmission */
    usbHacoHandleIrqs();
}
```

4.1.2 Interrupt handling

Add the following code:

For USB interrupt request:

```
volatile uint16_t gui_doUsb = 1;

void isr_gpio_falling (void)
{
    if (PONEDGE_STAT & BIT1)
    {
        gui_doUsb = 1;
        g_status0.wakeupEnd = 1;
    }

    PONEDGE_CLR = 0x3F;
}
```

For initialization:

```
/**
 * Initialize the SPI module and the MAX3420E SPI-USB bridge.
 *
 * @post GPIO 2..5 configured for SPI
 */
usbInitialize(USB_PART_ON, 8);

/** Config Interrupt */
PONEDGE_EN |= BIT1;
IRQ_MASK_H |= VBH_GPIO_FALLING;
```

And for the USB task in main loop as in 4.1.1:

```
if (gui_doUsb == 1)
{
    /**
     * If USB-request occurred during a measurement
     * clear the wakeupEnd flag
     */
    g_status0.wakeupEnd = 0;

    gui_doUsb = 0;

    /** Do transmission */
    usbHacoHandleIrqs();
}
```

For detailed information see application note d1_usb

Appendix

A. Record of Revisions

Version	Date	Author	State	Comment
1.0	2007-11-29	MOST	Release	
2.0	2008-04-11	MOST	Release	Chapter 4.2.1, 4.2.2 and 5 removed.
3.0	2008-12-03	FDE	Release	usbInitialize with parameters; Interrupt handling added
4.0	2010-05-04	MKI	Release	Reworked for firmware 4.0