

– preliminary –

Coordinates3D

Software-Specification

Version 4.0
04.06.2010



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

Contents

1	Forword.....	3
2	Introduction	4
2.1	Overview.....	4
3	Working with the Coordinates3D library	5
3.1	Integrating the library into the application project	5
4	Coordinates3D reference	6
4.1	Initializing the Coordinates library	6
4.1.1	Hardware initialization	6
4.1.2	Software initialization	7
4.2	Calculating three dimensional coordinates	7
4.2.1	The position array	7
4.2.2	Offset for the coordinate system	8
4.2.3	Zooming the coordinate system	8
4.3	Rotating the coordinate system	9

1 Forword

Thank you for your interest in our HALIOS®-products.

This document describes the contents of the HALIOS® firmware 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
- USB library documentation
- HALIOS® Tools documentation

2 Introduction

This library contains some functions which are calculating the HALIOS[®] sensor signal into three dimensional coordinates. With these coordinates it is possible to develop powerful three dimensional human machine interfaces without deep knowledge of the HALIOS[®] sensor principles.

2.1 Overview

The library Coordinates3D (c3d) depends on the HALIOS[®] Firmware and the HALIOS[®] Tools for IC E909.05 or E909.06. The application has to link against following libraries:

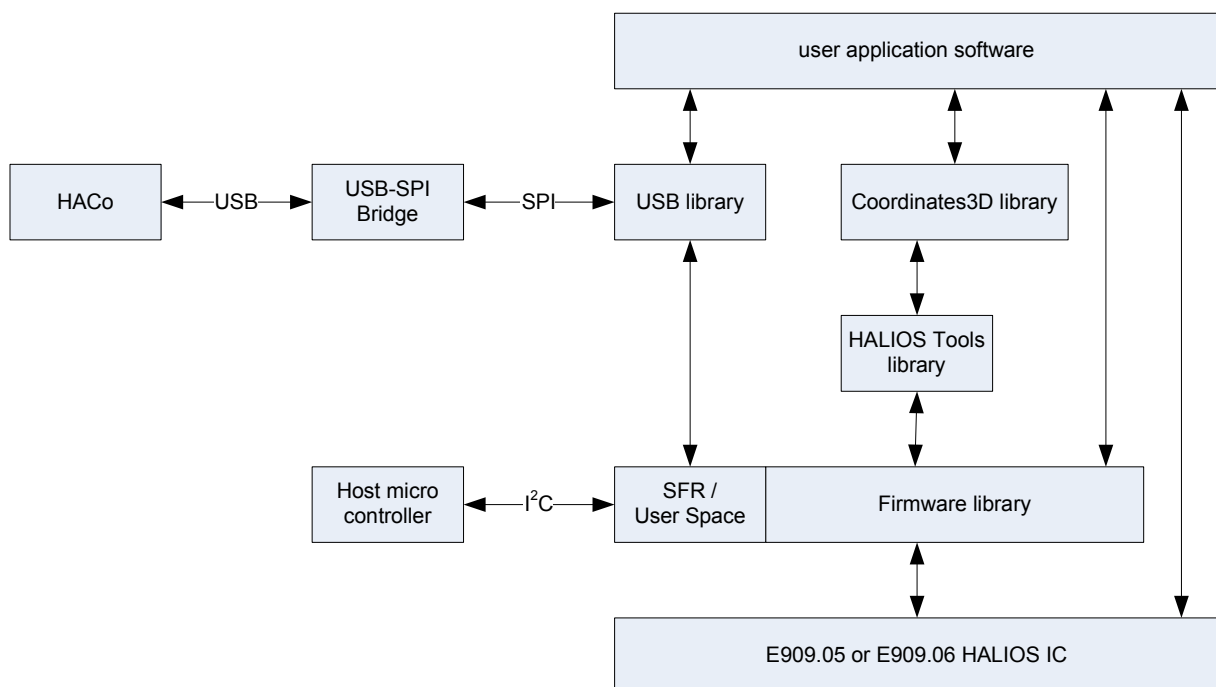
- ▶ lib_firmware_05 or lib_firmware_06
- ▶ lib_haliostools
- ▶ lib_c3d

For USB communication with HALIOS[®] Config tool the application has to be linked against

- ▶ lib_usb

too.

In "Picture 1" a layer view of an application with E909.05 or E909.06 is shown. This application uses the firmware for E909.05 or E909.06, the USB library, the HALIOS[®] Tools library and the Coordinates3D library.



Picture 1: Layer view

3 Working with the Coordinates3D library

An Integration of the Coordinates3D library into the system requires that the system designer implements the components discussed below, together with a proper handling of their implementation aspects.

3.1 Integrating the library into the application project

Any application software for the HALIOS® IC E909.05 or E909.06 which uses the API functions of the Coordinates3D library has to include the following header and library files (distributed by Mechaless Systems GmbH):

For E909.05 or E909.06:

```
firmware.h      (hardware specification)
usb.h           (header file of the USB library)
haliostools.h  (header file of the HALIOS® Tools library)
c3d.h          (header file of the Coordinates3D 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)
lib_haliostools.r43  (haliosTools)
lib_c3d.r43           (coordinates3D library)
```

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)
lib_haliostools.a  (haliosTools)
lib_c3d.a           (coordinates3D library)
```

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 Coordinates3D reference

The Coordinates3D library makes it possible to calculate three dimensional coordinates from four HALIOS® loop values. To enforce this it uses the filter and calibration functions of the HALIOS® Tools library.

4.1 Initializing the Coordinates library

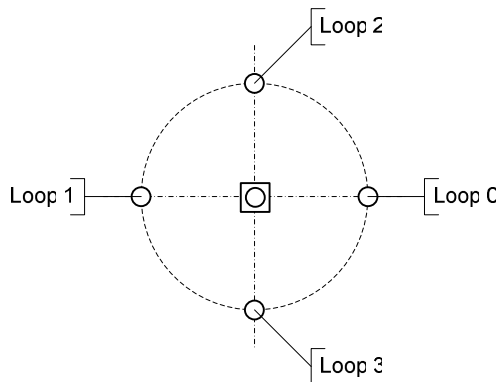
To use the coordinates3d library some hardware and software initialization must be done.

4.1.1 Hardware initialization

To calculate the x coordinates loop 0 and loop 1 is used. Loop 2 and loop 3 are used to calculate the y coordinates. To calculate the z coordinates loop 0 to loop 3 are used.

All four loops have to be configured in that way, that the sender LED in phase B regulates against the compensator in phase A.

Picture “Picture 2” shows the hardware LED configuration.



Picture 2: Hardware configuration

In a simplified manner the x and y coordinates are calculated as described in “Equation 1”.

$$x = loop0 - loop1$$

$$y = loop2 - loop3$$

Equation 1: Calculation of the x and y coordinates

4.1.2 Software initialization

With the function

```
void c3dInitialize(uint16_t xScale, uint16_t yScale, uint16_t zScale);
```

the three dimensional coordinate system, the low pass filter and the calibration are initialized. For details to filtering and calibration please refer to the “E909.05A HALIOS® Tools” software-specification.

The parameters `xScale`, `yScale`, and `zScale` defines the maximum absolute values of the x,y,z-cube. As default values in `coordinates3d.h` the three constants `XSCALE`, `YSCALE` and `ZSCALE` are defined. So if this constants are passed to the parameters of `c3dInitialize()` the coordinate system is defined as shown in “Equation 2”.

$$-xScale \leq x \leq +xScale$$

$$-yScale \leq y \leq +yScale$$

$$0 \leq z \leq zScale$$

Equation 2: Scaling of the coordinate system

The function `c3dInitialize()` should be only called once while the initialization routine of the E909.05A application software because a calibration of the HALIOS® loops is enforced when the function is called.

4.2 Calculating three dimensional coordinates

The function

```
void c3d( uint16_t *loop, uint16_t *quis_loop, int16_t *pos, int16_t xOffs, \
         int16_t yOffs, uint16_t xFact, uint16_t yFact, uint16_t zFact);
```

calculates three dimensional coordinates from the filtered loop values. The filtered loop values are used to make it possible to suppress jittering of the calculated coordinates if the noise of the raw loop values is to big.

4.2.1 The position array

The first parameter

```
int16_t *pos
```

is a pointer to an array of signed integer with three elements. The meaning of the three elements is as follows:

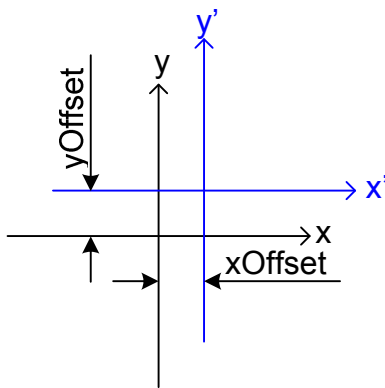
pos[0]	x coordinate
pos[1]	y coordinate
pos[2]	z coordinate

4.2.2 Offset for the coordinate system

To make the function more flexible it is possible to define with the two parameters

```
int16_t xOffs,
int16_t yOffs
```

an offset for the x and the y axis of the coordinate system. If e.g. the sensor LEDs are not symmetrical positioned the resulting scrolling of the coordinate system can be compensated with these offset parameters. This example is illustrated in "Picture 3".



Picture 3: x and y offset

4.2.3 Zooming the coordinate system

The c3d() function calculates the x, y and z coordinates and normalizes them, so that the coordinates have the ranges as shown in "Equation 3".

$$-1 \leq x \leq 1$$

$$-1 \leq y \leq 1$$

$$0 \leq z \leq 1$$

Equation 3: Normalized coordinates

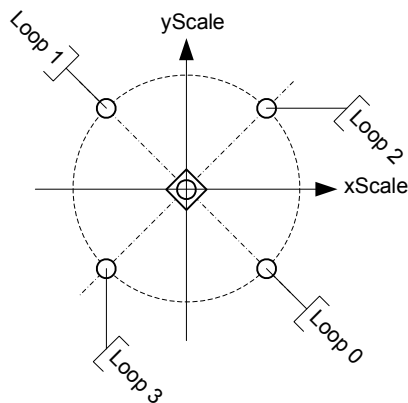
To reach the ranges shown in "Equation 2: Scaling of the coordinate system" the normalized coordinates are multiplied with the passed parameters

```
uint16_t xFact,
uint16_t yFact,
uint16_t zFact
```

and afterward clipped to the `xScale`, `yScale` and `zScale` values. This means, that it is possible to zoom in and out the coordinate system with the `xFact`, `yFact` and `zFact` parameters.

4.3 Rotating the coordinate system

If the LED positions are twisted against the wanted coordinate system, as shown in “Picture 1”, then the function `void c3dRotFlip(int16_t *pos, int16_t rot, uint16_t flip);` rotates the calculated coordinate system to the target coordinate system.



Picture 4: Rotated coordinate system

The first parameter `*pos` is the three dimensional position array which contains the elements `{x, y, z}`.

The second parameter `rot` determines the angle in degrees in which the coordinates will be rotated. Allowed values for `rot` are: 0, 45, 90, 135, 180, 225, 270 and 315.

The third parameter `flip` determines how to mirror the calculated position on the axis. Only the following values for `flip` are allowed:

- `M3D_FLIP_OFF` nothing will be mirrored,
- `M3D_FLIP_X` the position will be mirrored on the x axis,
- `M3D_FLIP_Y` the position will be mirrored on the y axis,
- `M3D_Flip_XY` the position will be mirrored on the x and y axis.

The resulting position, after rotating and flipping, is stored in the passed parameter `*pos`.

Appendix

A. Record of Revisions

Version	Date	Author	State	Comment
1.0	2008-01-15	MOST	Release	
2.0	2008-04-11	MOST	Release	Chapter 0 and 0 updated. Chapter 5 removed.
3.0	2008-10-11	MKI	Release	With firmware 3.0 mouse3d is now coordinates 3D (c3d). Chapter 4.2 updated. Chapter 5, Sample application removed.
4.0	2010-05-04	MKI	Release	Documentation for firmware 4.0 reworked

