# Application Note for E909.05 and E909.6 how to use isr 4.0

Mechaless Systems GmbH

LIMITED WARRANTY

THERE IS NO WARRANTY FOR THE CORRECTNESS OF THIS PUBLICATION OR THE RELATED CODE EXAMPLES OR RELATED PROGRAMS TO THE EXTENT PERMITED BY APPLICABLE LAW EXCEPT WHEN OTHERWISE STATED IN WRITING. THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PUBLICATION OR THE PUBLISHED CODE EXAMPLES OR RELATED PROGRAMS "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PUBLICATION OR THE PUBLISHED CODE EXAMPLES OR RELATED PROGRAMS IS WITH YOU. SHOULD THE PUBLICATION OR THE PUBLISHED CODE EXAMPLES OR THE RELATED PROGRAMS PROVE TO BE INCORRECT OR DEFECTIVE, YOU ASSUME THE COST OF ALL NECCESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT, UNLESS REQUIRED BY APPLICABLE LAW, WILL THE COPYRIGHT HOLDER OR ANY OTHER PARTY BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTICAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THIS PUBLICATION OR THE PUBLISHED CODE EXAMPLES OR THE RELATED PROGRAMS (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR FAILURE OF THE PROGRAMS/PROGRAM EXAMPLES TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF THE COPYRIGHT HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

# Chapter 1

# Application Note for E909.05 and E909.6 how to use isr File Index

## 1.1 Application Note for E909.05 and E909.6 how to use isr File List

Here is a list of all documented files with brief descriptions:

# Chapter 2

# Application Note for E909.05 and E909.6 how to use isr File Documentation

## 2.1 main.c File Reference

Application example to demonstrate the user defined interrupt handlers with HALIOS IC E909.05 and E909.06. In this Application Note the TIMER1_EVENT interrupt will be replaced by user defined code.

```
#include "firmware.h"
#include "main.h"
```

### Functions

- const uint16_t gui_applicationVersion __attribute__ ((section(".application_version")))
- void isr_timer1_event (void)
- int main (int argc, char ∗argv[ ])

### Variables

- const char gArc_project_number [ ] = "0908503"

### 2.1.1 Detailed Description

Application example to demonstrate the user defined interrupt handlers with HALIOS IC E909.05 and E909.06. In this Application Note the TIMER1_EVENT interrupt will be replaced by user defined code.

**Author:**

Florian Degler, Mechaless Systems GmbH

**Date:**

Created: 2008-11-18

**Author:**

Roland Muenzer, Media System Consulting

**Date:**

Changed: 2008-12-03 Reworked documentation

**Author:**

Florian Degler, Mechaless Systems GmbH

**Date:**

Changed: 2010-05-28 Reworked for firmware V4.0

**Author:**

Markus Kilian, Mechaless Systems GmbH

**Date:**

Changed: 2010-05-31 Reworked for firmware V4.0

**Author:**

Markus Kilian, Mechaless Systems GmbH

**Date:**

Changed: 2010-09-14 Makro request for __cpluplus added. Firmware now runs with c++.

**Id**

main.c,v 1.3 2011/03/17 15:37:34 mki Exp

Definition in file main.c.

## 2.1.2 Function Documentation

### 2.1.2.1 const uint16_t gui_applicationVersion __attribute__ ((section(".application_-version")))

Set a project application version number. Set to a fix area at FLASH to make possible read out in output file and verify the flashed code.

### 2.1.2.2 void isr_timer1_event (void)

This is the example function, that replaces the TIMER1_EVENT interrupt.

**Note:**

To replace an interrupt you have to replace the "0" to "1" in config.h for the corresponding line, e.g. for this TIMER1_EVENT define USE_X_isr_timer1_event as 1

**Remarks:**

Please keep following points in mind when you replace an interrupt function:

- If you replace an isr function, you disable the firmware functionality and have to handle the interrupt by yourself!
- Not all interrupts are enabled by default in the firmware. If you need an interrupt which is not used by the firmware you have to enable the interrupt by yourself.
- If you replace one of the following SPI interrupts, the SPI protocoll of the firmware is disabled.
  - Vector number 16: SPI receive high water
  - Vector number 17: SPI send low water
- If you replace one of the following I2C interrupts, the I2C protocoll of the firmware is disabled and you can’t use tools like HACo with RS232-I2C-bridge to configure the device.
  - Vector number 10: I2C receive command
  - Vector number 13: I2C receive fifo high water

**Note:**

It is important to use the correct name for the interrupt to be replaced. You can simply derive the name from the name in the configuration file config.h by stripping the leading *USE_X_* ...

```
**/
    /*
     *  This interrupt will be fired on every TIMER1_EVENT.
     *
     *  Do something usefull here, e.g. toggle a port pin:
     */
    P0OUT ^= BIT0;

    /*
     *  You have to clear the interrupt flag before returning to the
     *  application.
     */
    TIMER1_IRQCLR = TIMER_IRQCLR;
    /**
```

Definition at line 85 of file main.c.

#### 2.1.2.3 int main (int *argc*, char ∗ *argv*[ ])

main

**Parameters:**

← ***argc*** dummy parameter

← ***argv*** dummy parameter

```
**/
    loopConf_t t_loopConf;

    /**
    * Initializes the HALIOS SFRs and set up the basic functions of hardware.
    * @n It is recommand to call this function as first call.
    *
    * @post The system is configured:
    *  - The trimmvalues are read from InfoBlock and set to
    *    mclk and wkclk (only at (E909.05)
    *  - Following interrupts are enabled:
    *      - HALIOS measurement ready
    *      - wakeup timer
    *  - Following GPIO settings are used:
    *      - The RDY_PIN will set as output,
    *          if no readypin is needed set RDY_PIN as 0
    *  - Wakup timer enabled and set to 10 ms, used for sample time
    *  - One HALIOS loop enabled and set up (one LED against compensator).
    *
    * @param [in] BIT0 Set a GPIO as trigger pin for measurement, use only one bit.
    *              If not needed set to 0.
    */
```

```
    haliosInitialize(BIT0);

    /*
     *  To replace an interrupt do the following steps:
     *
     *  1st: For save behaviour disable the peripheral interrupt
     */
    IRQ_MASK_L &= ~VBL_TIMER1_EVENT;

    /*
     *  !!!! 2nd: Overwrite the ISR !!!! already done using config.h switch
     */

    /*
     *  3rd: Configure the peripheral
     *  Set timer1 value to 10ms
     *
     *  ATTENTION: 1st write the value to TIMER1_VAL_L and
     *  2nd to TIMER1_VAL_H!
     */
    TIMER1_VAL_L = T1_VAL_L;
    TIMER1_VAL_H = T1_VAL_H;

    TIMER1_CTRL = TIMER_ENABLE          /* Enable timer */
        | TIMER_LOOP            /* Timer in loop mode */
        | TIMER_RESET           /* Reset timer */
        | TIMER_PWD;

    /*
     *  4th and finally: Enable the peripheral interrupt
     */
    IRQ_MASK_L |= VBL_TIMER1_EVENT;

    /**
    * Set the projectnumber (eight characters) to g_sfr.project_number to make
    * readable about the constant reading mechanism @ref paramCheckSfr.
    *
    * @param[in]   gArc_project_number Pointer to a string. The maximum numbers of eight ch
    */
    paramSetProjectNr((uint8_t*)(gArc_project_number));

    /** Setup the register of the watchdog timer0.
    *
    * Configure the watchdog in milliseconds (ms).
    *
    * @param[in] 500 Watchdogtime in ms.
    * @n Must be smaller than 500 seconds (s)!
    * @n Higher Values will ignore and set to 500 s
    */
    deviceSetWatchdogTime (500UL);
```

```
/** set IO port function to GPIO for all pins */
P0CFG = 0;

/**
* Define which communication device will be used and enable or disable the
* related interrupts.
* @n This function is optional. If this function is not called, communication
* devices set all to off.
*
* @param[in] DEVCOM_I2C set communication to I2C
*  - For no communication device use (@ref DEVCOM_NO_COMM)
*  - For I2C (@ref DEVCOM_I2C)
*  - For SPI (@ref DEVCOM_SPI).
*  - For SPI and I2C (@ref DEVCOM_I2C | @ref DEVCOM_SPI)
*/
deviceSetCommDevice(DEVCOM_I2C);

/**
* Call this function to show the last reset reason at a pin
* by a significant bit pattern.
* @n This function is optional. Use only if you don't want to
* do your own fail state.
* @n
* @n Count the blink sequence of the output pin:
* - 4 times blinking: watchdog reset
* - 5 times blinking: CPU register parity error
* - 6 times blinking: FLASH uncorrectable bit error
* - 7 times blinking: RAM perity error
* - 8 times blinking: Trap
* @n @n
* @param[in] outputPin Define the pin which should do the failState show
* @param[in] inputPin Define the pin which break the failState show.
*                           Set to 0 if now break is required
*/
failState(BIT2, BIT3);

/**
*  Compute the checksum over all words in "Parameter FLASH Area".
*  If the Checksum proofs "Valid Data", data is copied from the
*  "Parameter FLASH Area" into RAM.
*
* @return
*          - -1:   No valid data found.
*          - else: Number of copied words.
*/
if (deviceRestore() == -1)
{
    /**
    *  Set the sample time in milli seconds. The wakeup timer
    *  of the Analog Control Module is used for the timing.
    *  Depending on the communication device the micro-controller
```

```
*  switches to STANDBY or STOP mode.
*
* @note time in milli seconds, must be between 2 and 32, only even
* values are accepted. (See also description of the Analog Control Module).
*/
paramSetSampleTime(8);

/**
* Set the amount of active loops.
*
* @param[in] count Amount of active loops. @a count must be less or equal to
*  @ref LOOPMAXCOUNT.
* @return An element of the @a HaliosCode enumeration:
*      - HALIOS_OK: No error occured
*      - HALIOS_PARAM: Wrong parameter for count passed.
*/
haliosSetLoopCount(4);

/**
* Configuration of the 1st loop.
* This is an example how to use type loopConf_t for loop configuration.
* The values are indices for the LED current of the ASIC.
*/
t_loopConf.loopNr = 0;
t_loopConf.ledConf = H_LED3B| H_LED5A| H_AON | H_ACCON;
t_loopConf.phaseA.range = 10;
t_loopConf.phaseA.offset = 22;
t_loopConf.phaseB.range = 15;
t_loopConf.phaseB.offset = 15;
t_loopConf.iConfC = 15;
t_loopConf.DC_offset = 0;
t_loopConf.PreAmp = 0;
t_loopConf.ClockConf = 0;

/**
* Store the configuration data into the virtuel loops at SFR by using
* a struct @ref LoopConf.
*
* @param[in] t_LoopConfig The LED and current configuration.
*
* @return An element of the @ref HaliosCode enumeration:
*      - HALIOS_OK:    No error occured
*      - HALIOS_PARAM: Wrong parameter in @a t_LoopConfig passed.
*/
haliosLoopInit(t_loopConf);

/**
* Store the configuration data into the virtuel loops at SFR by direct access.
*
* @note No validation check will done. It is recommand to use
* the function @ref haliosLoopInit.
```

```
     *
     * @param[in] loopNr    0 .. @ref LOOPMAXCOUNT
     * @param[in] ledConf   LED and measurement configuration.
     * @param[in] iClockConf Measurement Configuration HALIOS Clock
     * @param[in] iConfA    Current configuration for phase A.
     * @param[in] iConfB    Current configuration for phase B.
     * @param[in] iConfC    Current configuration for the compensator offset.
     * @param[in] iPreAmp   Preamplifier Configuration
     */
     haliosLoopInitialize(1, 20993, 0, 875, 495, 27, 0);
     haliosLoopInitialize(2, 20996, 0, 810, 495, 25, 0);
     haliosLoopInitialize(3, 21056, 0, 908, 495, 29, 0);

     /**
     * Set System Status to be used for @ref deviceWaitForTimer during wait
     * until timer has elapsed or a interrupt wakes up the system.
     * @n This function is optional. If not called system status is STANDBY.
     * @n
     * @param[in]   SystemStatus    Selects system mode for deviceWaitForTimer
     *  - DEVSET_RUN:        Keep System in RUN Mode in deviceWaitForTimer
     *  - DEVSET_STANDBY:    Switch to STANDBY Mode in deviceWaitForTimer
     *  - DEVSET_STOP:       Switch to STOP Mode in deviceWaitForTimer
     *  - DEVSET_OFF:        Switch to OFF Mode in deviceWaitForTimer
     *
     * Keep in mind that spi-usb communication only works in RUN and in STANDBY mode.
     */
     deviceSetSystemStatus(DEVSET_STANDBY);

     /**
     * Write a value to the userspace
     *
     *  These values could be read using I2C Protocol or
     *  USB (and HACo).
     *
     *  The size of user space is 255 words.
     */
     paramSetValue(0, 500);
}

/**
* Check the contents of SFR and does any special functions.
* If the content of a SFR register has changed the new values will be copied
* into the corresponding firmware functions or corresponding hardware registers.
*   - Set size of SFR and user space to address @ref BUFFSIZE at SFR
*   - Set constant reading values to SFR controled by @ref READ_CONST_CMD
*   - Set systemStatus
*   - Set Communication device
*   - Set sampletime
*   - Use spezial functions (use careful)
*   - Set main clock (ANALOG_MCLK) (Only E909.05)
*   - Set wakeup clock (ANALOG_WKCLK) (Only E909.05)
```

```
*   - Set HALIOS frequency (Only E909.06)
*   - Set number of Loops to g_sfr.loopCount
*/
paramCheckSfr();

/** Set GPIO 2..5 as output pins */
P0DIR &= ~(BIT2 | BIT3 | BIT4 | BIT5);

/** Set application bit and Version */
g_sfr.inst_libs |= BIT15;
deviceCheckVersion(BIT15, gui_applicationVersion);

/**
*
*  Do the measurement in an endless loop
*
*/
while (1)
{
    /**
    * Start and retrigger the watchdog timer. This is an inline function.
    *
    * @note At E909.06: After first call of watchdog it is not possible
    * to disable the watchdog or change the watchdog time.
    *
    */
    KICKDOG();

    /**
    * Check the contents of SFR and does any special functions.
    * If the content of a SFR register has changed the new values will be copied
    * into the corresponding firmware functions or corresponding hardware registers.
    *   - Set size of SFR and user space to address @ref BUFFSIZE at SFR
    *   - Set constant reading values to SFR controled by @ref READ_CONST_CMD
    *   - Set systemStatus
    *   - Set Communication device
    *   - Set sampletime
    *   - Use spezial functions (use careful)
    *   - Set main clock (ANALOG_MCLK) (Only E909.05)
    *   - Set wakeup clock (ANALOG_WKCLK) (Only E909.05)
    *   - Set HALIOS frequency (Only E909.06)
    *   - Set number of Loops to g_sfr.loopCount
    */
    paramCheckSfr();

    /**
    * Do the HALIOS measurement of all configurated loops.
    * - Enable the analog part
    * - Start one Warmup to engage the analog part
    * - Start the configured measurements
    * - disable the analog part
```

```
* - count up the @ref TIME_STAMP
*
*  When haliosMeasure() is called with parameter HALIOS_RDYON,
*  the configured PIN in haliosInitialize() will be switched on
*  when entering the haliosMeasure() function,
*  and will be switched off when haliosMeasure() is left.
*
* @param[in] readyPin @ref HaliosCode
*              - @ref HALIOS_RDYON GPIO is used as ready pin.
*              - @ref HALIOS_RDYOFF GPIO is not used as ready pin.
*/
haliosMeasure(HALIOS_RDYOFF);

/**
 * To get the measurement result use the function
 * haliosGetResult ()
 */
if (haliosGetResult(0) > 500)
{
    /** Set GPIO Pin2 if measurement result is above 500 */
    P0OUT |= BIT2;
}
else
{
    /** Reset GPIO Pin2 if measurement result is below 500 */
    P0OUT &= ~BIT2;
}

/**
*  Wait until the timer has elapsed.
*/
deviceWaitForTimer();

/**
```

Definition at line 108 of file main.c.

References gArc_project_number, T1_VAL_H, and T1_VAL_L.

## 2.2 main.h File Reference

### Defines

- #define APPLICATION_VERSION 103UL
- #define TIME1 10UL
- #define T1_VAL (TIME1 ∗ F_CPU)
- #define T1_VAL_H (T1_VAL / 65536UL)
- #define T1_VAL_L (T1_VAL % 65536UL)

### 2.2.1 Detailed Description

Header file for the example application.

**Author:**

Miroslav Ostric, Mechaless Systems GmbH

**Date:**

Created: 2007-03-13

**Author:**

Roland Muenzer, Media System Consulting

**Date:**

Changed: 2008-11-26 added comments, added missing include "firmware.h"

**Author:**

Florian Degler, Mechaless Systems GmbH

**Date:**

Changed: 2010-05-28 Reworked for firmware V4.0

**Author:**

Markus Kilian, Mechaless Systems GmbH

**Date:**

Changed: 2010-05-31 Reworked for firmware V4.0 added comments, removed obsolete include "firmware.h"

**Author:**

Markus Kilian, Mechaless Systems GmbH

**Date:**

Changed: 2010-07-13 Due to compatibility for GCC firmware library 4.01 available. Application version set to 1.01.

**Author:**

Markus Kilian, Mechaless Systems GmbH

**Date:**

Changed: 2010-08-12 Application version set to 1.02 Firmware library updated to 4.03 HALIOS tools library updated to 4.01

**Author:**

Markus Kilian, Mechaless Systems GmbH

**Date:**

Changed: 2011-03-17 Application version set to 1.03 Firmware library updated to 4.05

Definition in file main.h.

## 2.2.2 Define Documentation

### 2.2.2.1 #define APPLICATION_VERSION 103UL

Version number for the application.

Definition at line 36 of file main.h.

### 2.2.2.2 #define TIME1 10UL

Time for TIMER 1 in ms.

Definition at line 39 of file main.h.

#### 2.2.2.3 #define T1_VAL (TIME1 ∗ F_CPU)

Define TIME1 as CPU Clock Ticks instead of ms

F_CPU is CPU frequency in kHz. F_CPU is defined in the E90905A.h processor characterization file (see ../include/e90905a.h)

Definition at line 47 of file main.h.

#### 2.2.2.4 #define T1_VAL_H (T1_VAL / 65536UL)

T1_VAL Ticks High Word (T1_VAL / 65536)

Definition at line 49 of file main.h.

#### 2.2.2.5 #define T1_VAL_L (T1_VAL % 65536UL)

T1_VAL Ticks Low Word (T1_VAL % 65536)

Definition at line 51 of file main.h.

# Index