

This document highlights the changes between v1.0 and v1.1 of the Class B Library - Frequency.

Files removed/replaced:

- CLASSB\_FREQUENCY/applications/CLASSB\_FREQUENCY/error\_handler.h replaced by CLASSB/classb\_error\_handler.h
- CLASSB\_FREQUENCY/applications/CLASSB\_FREQUENCY/Tiny817xpro.h replaced by CLASSB/utis/oled1\_xpro\_attiny817.h
- CLASSB\_FREQUENCY/applications/CLASSB\_FREQUENCY/avr\_compiler.h replaced by CLASSB/utis/classb\_compiler.h
- CLASSB\_FREQUENCY/applications/CLASSB\_FREQUENCY/classb\_rtc\_common.h replaced by CLASSB/classb\_rtc.h

Files with diff/changelog:

- CLASSB\_FREQUENCY/applications/CLASSB\_FREQUENCY/classb\_freq.h
- CLASSB\_FREQUENCY/applications/CLASSB\_FREQUENCY/main\_frequency.c
- CLASSB\_FREQUENCY/documentation/CLASSB\_FREQUENCY.rst

Diff of classb\_freq.h:

```
----- CLASSB_FREQUENCY/applications/CLASSB_FREQUENCY/classb_freq.h -----
index c33b930..730feca 100644
@@ -2,23 +2,23 @@
/**
 * \file
 *
- * \version 1.1
+ * \version 1.2
 *
 * \brief
 *          Settings and definitions for the CPU frequency test.
 *
- * \par Application note:
- *   AVR1610: Guide to IEC60730 Class B compliance with TinyAVR 1-series
+ * \par Application note:
+ *   AN2632: Guide to IEC60730 Class B compliance with TinyAVR 1-series
 *
 * \par Documentation
 *   For comprehensive code documentation, supported compilers, compiler
 *   settings and supported devices see readme.html
 *
 * \author
- *   Microchip Technology: http://www.microchip.com \n
- *   Support at http://www.microchip.com/support/ \n
+ *   Microchip Technology: http://www.microchip.com
+ *   Support at http://www.microchip.com/support/
 *
- * Copyright (C) 2017 Microchip Technology. All rights reserved.
+ * Copyright (C) 2019 Microchip Technology. All rights reserved.
 *
 * \page License
 *
@@ -38,10 +38,10 @@
 * 4. This software may only be redistributed and used in connection with an
 * Microchip AVR product.
 *
- * THIS SOFTWARE IS PROVIDED BY Microchip "AS IS" AND ANY EXPRESS OR IMPLIED
+ * THIS SOFTWARE IS PROVIDED BY MICROCHIP "AS IS" AND ANY EXPRESS OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE
- * EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL Microchip BE LIABLE FOR
+ * EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL MICROCHIP BE LIABLE FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
```

```

* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
@@ -54,12 +54,10 @@
#ifndef CLASB_FREQ_H_
#define CLASB_FREQ_H_

#include "avr_compiler.h"
#include "classb_rtc_common.h"
#include "error_handler.h"
#include "classb_compiler.h"
#include "classb_error_handler.h"
#include "classb_rtc.h"
#include "classb_freq.h"
#include <avr/interrupt.h>
#include "error_handler.h"
#include <stdbool.h>

//! \defgroup cpu_freq CPU Frequency Test
@@ -118,7 +116,7 @@
//!      Reference = F_tc * Interrupt_Period_rtc / F_rtc
//!      Note that multiplying by 1e0L forces the preprocessor to do operations with
//!      higher precision.
#define CLASSB_TC_COUNT_REF          (uint16_t) ((1e0L * CLASSB_TC_FREQ) * RTC_INTERUP_PERIOD_TIME)
#define CLASSB_TC_COUNT_REF          (uint16_t) ((1e0L * CLASSB_TC_FREQ) *
RTC_INTERRUPT_PERIOD_TIME)

//! \internal \brief Maximum absolute difference between the reference and estimated frequencies
//!
@@ -153,7 +151,7 @@ volatile bool freq_test_not_done = true;
ISR(TCA0_OVF_vect)
{
    // Clear the interrupt flag
-    TCA0_SINGLE_INTFLAGS = TCA_SINGLE_OVF_bm;
+    TCA0.SINGLE.INTFLAGS = TCA_SINGLE_OVF_bm;
    // Increase the overflow counter.
    classb_tc_ovf_cnt++;
    // If the TC had more overflows than expected, the error should be handled.
@@ -167,7 +165,7 @@ ISR(TCA0_OVF_vect)
ISR(RTC_CNT_vect)
{
    // Clear the interrupt flag
-    RTC_INTFLAGS = RTC_OVF_bm | RTC_CMP_bm;
+    RTC.INTFLAGS = RTC_OVF_bm | RTC_CMP_bm;

    // Compute the absolute difference between reference and counter.
    classb_tc_ovf_cnt = (classb_tc_ovf_cnt > CLASSB_TC_COUNT_REF) ? (classb_tc_ovf_cnt -
CLASSB_TC_COUNT_REF) : (CLASSB_TC_COUNT_REF - classb_tc_ovf_cnt);
@@ -186,13 +184,13 @@ ISR(RTC_CNT_vect)
static inline void classb_freq_test(void)
{
    // Wait for RTC synchronization to complete
-    while (RTC_STATUS);
+    while (RTC.STATUS);
    // Enable RTC
-    RTC_CTRLA = RTC_RTCEN_bm;
+    RTC.CTRLA = RTC_RTCEN_bm;
    // Wait for RTC synchronization to complete
-    while (RTC_STATUS);
+    while (RTC.STATUS);
    // Enable TCA0
-    TCA0_SINGLE_CTRLA = TCA_SINGLE_ENABLE_bm;
+    TCA0.SINGLE.CTRLA = TCA_SINGLE_ENABLE_bm;

    // Note this can be modified so that something
    // else is done while the test is being performed
@@ -200,9 +198,9 @@ static inline void classb_freq_test(void)
    // for long periods of time.
    while (freq_test_not_done);
    // Disable RTC and TCA0
-    RTC_CTRLA = 0;
+    RTC.CTRLA = 0;

```

```

        // Disable RTC
-       TCA0_SINGLE_CTRLA = 0;
+       TCA0.SINGLE_CTRLA = 0;
        // Disable TCA
        freq_test_not_done = true;
    }
@@ -212,10 +210,10 @@ static inline void classb_freq_test(void)
static inline void classb_TCA_configure(void)
{
    // Set period value/Overflow value
-       TCA0_SINGLE_PER = CLASSB_TC_PER;
+       TCA0.SINGLE.PER = CLASSB_TC_PER;
    // Enable TCA overflow Interrupt
-       TCA0_SINGLE_INTCTRL = TCA_SINGLE_OVF_bm;
+       TCA0.SINGLE.INTCTRL = TCA_SINGLE_OVF_bm;
}

-#endif /* CLASB_FREQ_H_ */
\ No newline at end of file
+#endif /* CLASB_FREQ_H_ */

```

## Diff of main\_frequency.c:

```

----- CLASSB_FREQUENCY/applications/CLASSB_FREQUENCY/main_frequency.c -----
index 6abfa57..bfe8065 100644
@@ -19,18 +19,18 @@
*
*       button. This causes the measured frequency to change.
*
*
- * \par Application note:
- *   AVR1610: Guide to IEC60730 Class B compliance with TinyAVR 1-series
+ * \par Application note:
+ *   AN2632: Guide to IEC60730 Class B compliance with TinyAVR 1-series
*
* \par Documentation
*   For comprehensive code documentation, supported compilers, compiler
*   settings and supported devices see readme.html
*
* \author
- *   Microchip Technology: http://www.microchip.com \n
- *   Support at http://www.microchip.com/support/ \n
+ *   Microchip Technology: http://www.microchip.com
+ *   Support at http://www.microchip.com/support/
*
- * Copyright (C) 2017 Microchip Technology. All rights reserved.
+ * Copyright (C) 2019 Microchip Technology. All rights reserved.
*
* \page License
*
@@ -50,10 +50,10 @@
* 4. This software may only be redistributed and used in connection with an
* Microchip AVR product.
*
- * THIS SOFTWARE IS PROVIDED BY Microchip "AS IS" AND ANY EXPRESS OR IMPLIED
+ * THIS SOFTWARE IS PROVIDED BY MICROCHIP "AS IS" AND ANY EXPRESS OR IMPLIED
* WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
* MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE
- * EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL Microchip BE LIABLE FOR
+ * EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL MICROCHIP BE LIABLE FOR
* ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
@@ -64,9 +64,9 @@
*/

#include "avr_compiler.h"

```

```

+##include "classb_compiler.h"
#include "classb_freq.h"
-##include "Tiny817xpro.h"
+##include "oled1_xpro_attiny817.h"

NO_INIT volatile uint8_t classb_error;

@@ -108,5 +108,5 @@ ISR(PORTA_PORT_vect)
    // Clear interrupt flag
    BUTTON1_PORT.INTFLAGS = BUTTON1_PIN;
    // Change TCA period to induce error
-    TCA0_SINGLE_PER = 0x7FFF;
+    TCA0.SINGLE.PER = 0x7FFF;
+}

```

## Diff of CLASSB\_FREQUENCY.rst:

```

----- CLASSB_FREQUENCY/documentation/CLASSB_FREQUENCY.rst -----
index 8bbafaf..e014105 100644
@@ -17,13 +17,15 @@ if the error flag is set, the main program will exit the
loop and switch off the LED.

```

In order to test that the self-diagnostic routine is correct,

- the TC period can be modified on-the-fly by pressing a button. This causes the measured frequency to change.
- +the TC period can be modified on-the-fly by pressing BUTTON 1.
- +This causes the measured frequency to change.

Related documents / Application notes

- This application is described in the following application note: To be published
- +This application is described in the following application note:
- +`Guide to IEC 60730 Class B Compliance with tinyAVR 1-series
- <<http://www.microchip.com/wwwappnotes/appnotes.aspx?appnote=en604502>>`\_

Supported evaluation kit

@@ -31,9 +33,19 @@ Supported evaluation kit

- ATTiny817-XPRO

-Running the demo

+Running the demo using GCC

1. Press Download Pack and save the .atzip file
2. Import .atzip file into Atmel Studio 7, File->Import->Atmel Start Project.

- 3. Build and flash into supported evaluation board
- +3. Build and flash into supported evaluation board.

+Running the demo using IAR

- +1. Check "IAR Embedded Workbench", press Download Pack and save the .atzip file.
- +2. Follow the steps on how to download and import a Start project into IAR found in "How to open in IDEs"
- + found under export project.
- +3. Change linker file using iar\_cfgtiny817\_classB.xcl located in utils folder.
- +4. Build and flash into supported evaluation board.