

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

Files removed/replaced:

- CLASSB_ANALOG/applications/CLASSB_ANALOG/error_handler.h replaced by CLASSB/classb_error_handler.h
- CLASSB_ANALOG/applications/CLASSB_ANALOG/Tiny817xpro.h replaced by CLASSB/utis/oled1_xpro_attiny817.h
- CLASSB_ANALOG/applications/CLASSB_ANALOG/avr_compiler.h replaced by CLASSB/utis/classb_compiler.h

Files with diff/changelog:

- CLASSB_ANALOG/applications/CLASSB_ANALOG/classb_analog.h
- CLASSB_ANALOG/applications/CLASSB_ANALOG/main_analog.c
- CLASSB_ANALOG/documentation/CLASSB_ANALOG.rst

Diff of classb_analog.h:

```
----- CLASSB_ANALOG/applications/CLASSB_ANALOG/classb_analog.h -----
index 8de17bb..a5e6867 100644
@@ -2,23 +2,23 @@
/**
 * \file
 *
 - * \version 1.1
 + * \version 1.2
 *
 * \brief
 *   This is the header file for the ADC and DAC tests.
 *
 - * \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
@@ -54,131 +54,98 @@
 #ifndef CLASSB_ADC_DAC_H_
 #define CLASSB_ADC_DAC_H_

-#include "avr_compiler.h"
-#include "error_handler.h"
+#include "classb_compiler.h"
+#include "classb_error_handler.h"

-#!/ \defgroup adc_dac Analog I/O Test
```

```

+//! \defgroup ADC_DAC Analog I/O Test
+//!
+//! \brief This self-diagnostic test for the ADC, DAC and analog multiplexer.
+//!
+//! This files contains the following tests classb_DAC_ADC_test1(), classb_DAC_ADC_test2()
+//! and classb_adc_dac_window_mode_test().
+//!
+//! classb_adc_dac_window_mode_test() is used to test window mode of the ADC, the DAC, a subset
+//! of the event system, and TCA. This test should only be used if window mode is used. The test
+//! configures TCA to generate an interrupt to change the DAC value. TCA is also configured to
+//! generate an even on compare match. This event is routed trough the event system to the ADC.
+//! The ADC converts the DAC value. If it is inside the configured ADC window interrupts will be
+//! generated for each conversion where this is true. The ADC values on the "border" of the window
+//! is stored and compared at the end of the test.
+//! This files contains the following tests:
+//! classb_DAC_ADC_test1()
+//! classb_DAC_ADC_test2()
+//!
+//! classb_DAC_ADC_test1() test the ADC and the DAC both the ADC and the dac uses a reffrence generated
+//! from the internal bandgap reference. The DAC converts a digital value to analog and the ADC converts
+//! this value and a comparison is done at the end.
+//!
+//! classb_DAC_ADC_test2() is similar to classb_DAC_ADC_test1() with the exception that the ADC is
+//! configured to use VCC as reference voltage. This way the internal bandgap is tested as well.
+//! Note that with this test the tolerance for correct value must be larger than for classb_DAC_ADC_test1().
+//! If this test is used the DAC reference should be adjusted to closer match VCC. This test is not recommended
+//! in battery application or if VCC is very noisy.
+//! classb_ADC_DAC_test1() test the ADC and the DAC both the ADC and the DAC
+//! uses a reference generated from the internal bandgap reference. The DAC
+//! converts a digital value to analog and the ADC converts this value and a
+//! comparison is done at the end.
+//!
+//! classb_ADC_DAC_test2() is similar to classb_ADC_DAC_test1() with the
+//! exception that the ADC is configured to use VCC as reference voltage.
+//! By not using the VREF peripheral as reference to ADC, but keeping the
+//! bandgap based VREF as reference to the DAC, a failure in the latter
+//! can be detected.
+//!
+//! Note: classb_ADC_DAC_test2 limits are based on the application VDD value,
+//! and in addition significant deltas between expected and actual VDD
+//! could give false failures (e.g. battery decay).
+//! Thus limit margins should take these into account.
+//!
+//@{

#include "classb_analog.h"
#include <stdbool.h>
#include "Tiny817xpro.h"
#include "oled1_xpro_attiny817.h"

#define CLASSB_DAC_CONV_VALUE_1 0x3F
+// DAC output set to midpoint of 8-bit resolution (0.55V Ref)
#define CLASSB_DAC_CONV_VALUE_1 0x80

// This defines the upper an lower limit of the ADC reading of test 1.
-// The expected result is approximately 4 times the DAC value. due to the ADC being
-// 10-bit while the DAC is only 8-bit. The tolerance is here set to +-15%
#define CLASSB_ADC_RES_1_H (uint16_t)(CLASSB_DAC_CONV_VALUE_1 * 4) * 1.15
#define CLASSB_ADC_RES_1_L (uint16_t)(CLASSB_DAC_CONV_VALUE_1 * 4) * 0.85
+// The expected result is approximately 4 times the DAC value. due to the
+// ADC being 10-bit while the DAC is only 8-bit. The tolerance is here
+// set to +-15%
+// Both ADC and DAC reference is internal (0.55V).
#define CLASSB_ADC_RES_1_LIMIT_H (uint16_t)(CLASSB_DAC_CONV_VALUE_1 * 4) * 1.25
#define CLASSB_ADC_RES_1_LIMIT_L (uint16_t)(CLASSB_DAC_CONV_VALUE_1 * 4) * 0.75

-// This defines the upper an lower limit of the ADC reading of test 2.
-// The result from the ADC will be dependent on the difference between
-// VDD and the DAC reference value. The range needs to be tuned tho this
-// in a final test.

```

```

#define CLASSB_ADC_RES_2_H (uint16_t)(CLASSB_DAC_CONV_VALUE_1 * 4) * 1.25
#define CLASSB_ADC_RES_2_L (uint16_t)(CLASSB_DAC_CONV_VALUE_1 * 4) * 0.75
-
-
-NO_INIT volatile bool classb_window_mode_test_not_done;
-NO_INIT volatile uint16_t classb_lower_ADC_value;
-NO_INIT volatile uint16_t classb_top_ADC_value;
-NO_INIT volatile uint8_t classb_DAC_data_value;
+// magic number 64E+3 ~ 3.1 ms (@20 Mhz), 8 ms (@8 Mhz).
#define CLASSB_ADC_OUTPUT_SETTLE 64000

// Global error variable.
NO_INIT volatile uint8_t classb_error;

-//! /brief ADC ISR used by the window mode test
-//! This ISR reads the ADC result of the ADC conversions done when the result is within the window.
-//! The upper and lower limits of the ADC window are stored for later comparison.
-//! If window mode is not used in the final application and the test is not executed
-//! this ISR should be commented out or removed. If it is not removed it will take up code space.
-/* Uncomment to run the window mode test
-ISR(ADC0_WCOMP_vect)
+//! \brief Test to see if ADC completes acquisition within expected time frame.
+//!
+//! This test triggers an ADC conversion start, waits until complete or timeout.
+//! If timeout occurs error is classb_error is set.
+//!
+
+static inline void classb_ADC_start_conversion(void)
+{
-    if (classb_lower_ADC_value == 0)
+    // Start ADC conversion
+    ADC0.COMMAND = ADC_STCONV_bm;
+
+    // Wait for conversion to finish or time to run out
+    uint8_t timeout = 16;
+    while (!(ADC0.INTFLAGS & ADC_RESRDY_bm) && timeout != 0)
+    {
-        classb_lower_ADC_value = ADC0_RES;
+        timeout--;
+    }
-    classb_top_ADC_value = ADC0_RES;
-}
-*/

-//! /brief TCA0 ISR used by the window mode test
-//! This ISR changes the DAC data value each time TCA0 overflows.
-//! When the DAC has cycled through all values the test is done.
-//! If window mode is not used in the final application and the test is not executed
-//! this ISR should be commented out or removed. If it is not removed it will take up code space.
-/* Uncomment to run the window mode test
-ISR(TCA0_OVF_vect)
-{
-    TCA0_SINGLE_INTFLAGS = TCA_SINGLE_OVF_bm;
-    if (classb_DAC_data_value == 0xFF)
+    // Clear ADC result ready interrupt flag
+    ADC0.INTFLAGS = 0;
+
+    // If conversion takes longer than expected set the error flag
+    if (timeout == 0)
+    {
-        classb_window_mode_test_not_done = false;
-        return;
+        classb_error = 1;
+    }
-    classb_DAC_data_value += 1;
-    DAC0_DATA = classb_DAC_data_value;
-}
-*/

-//! /brief test ADC and DAC

```

```

//!
//! In this test the ADC and DAC has the reference derived from the bandgap reference.
//! The ADC does a conversion of the DAC and compares the result to a set range.
-static inline void classb_DAC_ADC_test1(void)
+static inline void classb_ADC_DAC_test1(void)
{
-    //Start ADC conversion
-    ADC0_COMMAND = ADC_STCONV_bm;
-    //Wait for conversion to finish or time to run out
-    uint8_t timer = 16;
-    while (!(ADC0_INTFLAGS & ADC_RESRDY_bm) )
-    {
-        timer--;
-        if (timer == 0)
-        {
-            break;
-        }
-    }
-
-    //Clear ADC result ready interrupt flag
-    ADC0_INTFLAGS = 0;
-
-    //If conversion takes longer than expected set the error flag
-    if (timer == 0)
-    {
-        classb_error = 1;
-    }
+    // Start ADC conversion, and check completion within timeout.
+    classb_ADC_start_conversion();
+
-    //Read ADC result
-    uint16_t adc_res = ADC0_RES;
+    uint16_t adc_res = ADC0.RES;
-
-    //If ADC result is not within an acceptable range set the error flag
-    if ((CLASSB_ADC_RES_1_L > adc_res) || (adc_res > CLASSB_ADC_RES_1_H))
+    if ((CLASSB_ADC_RES_1_LIMT_L > adc_res) || (adc_res > CLASSB_ADC_RES_1_LIMT_H))
-    {
-        classb_error = 1;
-    }
}
@@@ -186,131 +153,56 @@@ static inline void classb_DAC_ADC_test1(void)

//! \brief Test to see if band gap is correct
//!
-//! This test sets VCC as reference voltage to the ADC. While the DAC stil usses teh bandgap as reffrence.
+//! This test sets VCC as reference voltage to the ADC.
+//! While the DAC still uses the bandgap as reference.
//!
-//! Note that this test will give different results depending on what VCC is, and the test tolerance
-//! has to be adjusted. As VCC can be expected to have more noise than the internal bandgap the tolerance has
-//! to be larger.
-static inline void classb_DAC_ADC_test2(void)
+//! The limits are VCC dependent, thus need to be provided.
+//!
+static inline void classb_ADC_DAC_test2(uint16_t high_limit, uint16_t low_limit)
{
-    /* Changes reference to ADC to the external voltage and do new test to
-    * test the band gap reference
+    /* Changes reference to ADC to the external voltage
+    * and do new test to test the band gap reference
+    */
-    ADC0_CTRLIC |= ADC_REFSEL_VDDREF_gc;
-    //Start ADC conversion
-    ADC0_COMMAND = ADC_STCONV_bm;
-
-    uint8_t timer = 16;
-    while (!(ADC0_INTFLAGS & ADC_RESRDY_bm) || timer != 0)
-    {
-        timer--;
-    }

```

```

- //Clear ADC result ready interrupt flag
- ADC0_INTFLAGS = 0;
-
- //If conversion takes longer than expected set the error flag
- if (timer == 0)
- {
-     classb_error = 1;
- }
+ uint8_t adc_ctr_orig = ADC0.CTRL0;
+ ADC0.CTRL0 |= ADC_REFSEL_VDDREF_gc;
+
+ // Start ADC conversion, and check completion within timeout.
+ classb_ADC_start_conversion();
+ //Read ADC result
- uint16_t adc_res = ADC0.RES;
+ uint16_t adc_res = ADC0.RES;
-
- //If ADC result is not within an acceptable range set the error flag
- if ((CLASSB_ADC_RES_2_L > adc_res) || (adc_res > CLASSB_ADC_RES_2_H))
+ if ((low_limit > adc_res) || (adc_res > high_limit))
+ {
+     classb_error = 1;
+ }
-}
-
-//!! /brief Testing of the ADC window mode
-//!!
-//!! In this test the ADC does a conversion of the DAC each time TCA sends an over flow event
-//!! Each time TCA gets a Compare match interrupt the DAC output is incremented. The ADC will
-//!! generate an interrupt each time it converts a value within the window. In this way a
-//!! sweep of values converted is performed. At the moment only the border values of the ADC
-//!! window is stored and compared.
-static inline void classb_adc_dac_window_mode_test(void)
-{
-    // initialize test variables in case test is run without reset.
-    classb_window_mode_test_not_done = true;
-    classb_lower_ADC_value = 0;
-    classb_top_ADC_value = 0;
-    classb_DAC_data_value = 0;
-
-    //set initial value for DAC
-    DAC0_DATA = 0;
-
-    // Configure Event system to route signal from TC to ADC
-    // Set TCA Compare channel as event generator on synchronous event channel 0
-    EVSYS_SYNCCH0 = EVSYS_SYNCCH0_TCA0_CMP0_gc;
-    // Set ADC as event user to event channel
-    EVSYS_ASYNCUSER1 = EVSYS_ASYNCUSER1_SYNCCH0_gc;
-
-    // Configure TCA to generate event and interrupt
-    // Generate interrupt on overflow
-    TCA0_SINGLE_INTCTRL = TCA_SINGLE_OVF_bm; // | TCA_SINGLE_CMP0_bm; // | TCA_SINGLE_CMP0_bm
-    // Set the overflow value for when to update DAC output
-    TCA0_SINGLE_PER = 0xFF;
-    // Set the compare value for when events will be generated to start ADC conversion
-    TCA0_SINGLE_CMP0 = 0x7F;
-
-    TCA0_SINGLE_DBGCTRL = TCA_SINGLE_DBGRUN_bm;
-
-    // reconfigure ADC to Window mode
-    // disable ADC
-    ADC0_CTRLA = 0;
-    // set lower ADC window value
-    ADC0_WINLT = 0x7F;
-    // set upper ADC window value
-    ADC0_WINHT = 0x1FF;
-    // set window comparison mode to trigger when ADC result is inside the window
-    ADC0_CTRLB = ADC_WINCM_INSIDE_gc;
-    // set ADC to do a conversion each time an event occurs.
-    ADC0_EVCTRL = ADC_STARTEN_bm;

```

```

- // enable ADC window mode interrupt
- ADC0_INTCTRL = ADC_WCMP_bm;
-
- // enable the ADC
- ADC0_CTRLA = ADC_ENABLE_bm;
- // enable TCA
- TCA0_SINGLE_CTRLA = TCA_SINGLE_ENABLE_bm | TCA_SINGLE_CLKSEL_DIV256_gc;
-
- // test is running
- sei();
- while (classb_window_mode_test_not_done)
- {
-     //sleep_cpu();
- }
- cli();
- // disable TCA
- TCA0_SINGLE_CTRLA = 0;
-
- // check the top and bottom ADC values
- if (classb_lower_ADC_value == 0 || classb_lower_ADC_value >= classb_top_ADC_value)
- {
-     classb_error = 1;
- }
- // NOTE: This check is not very thorough, one should also check that top and bottom ADC values are in an expected
range.
+ // Setback reference of ADC.
+ ADC0_CTRLA = adc_ctrc_orig;
+ }
+
+ //! \brief Configuration needed by all ADC DAC tests
+ //!
+ //! This function configures the ADC and DAC with the basic configuration needed by all tests
-static inline void classb_base_configuration_of_DAC_and_ADC(void)
+static inline void classb_ADC_DAC_base_config(void)
+{
+    // Set ADC and DAC Reference to 0.55V.
+    VREF_CTRLA = ((uint8_t) VREF_ADC0REFSEL_0V55_gc | (uint8_t) VREF_DAC0REFSEL_0V55_gc);
+
+    // Set a starting value for the DAC
- DAC0_DATA = CLASSB_DAC_CONV_VALUE_1;
+ DAC0_DATA = CLASSB_DAC_CONV_VALUE_1;
+    // Enable DAC but do not output on pin
- DAC0_CTRLA = DAC_ENABLE_bm;
+ DAC0_CTRLA = DAC_ENABLE_bm;
+    // wait for DAC output to settle
- for(volatile uint16_t i = 0; i < 64000; i++);
+ for(volatile uint16_t i = 0; i < CLASSB_ADC_OUTPUT_SETTLE; i++);
+
+    // Set DAC as input to ADC
- ADC0_MUXPOS = ADC_MUXPOS_DAC0_gc;
- // Divide ADC clock
- ADC0_CTRLA = ADC_PRESC_DIV8_gc;
+ ADC0_MUXPOS = ADC_MUXPOS_DAC0_gc;
+ // Divide ADC clock, and default ADC internal ref
+ ADC0_CTRLA = ((uint8_t) ADC_PRESC_DIV8_gc | (uint8_t) ADC_REFSEL_INTREF_gc);
+    // Enable ADC
- ADC0_CTRLA = ADC_ENABLE_bm;
+ ADC0_CTRLA = ADC_ENABLE_bm;
+}

+ //! \brief Undo basic configuration
+ @@ -318,48 +210,14 @@ static inline void classb_base_configuration_of_DAC_and_ADC(void)
static inline void classb_undo_configuration_of_test_1_and_2(void)
+{
+    // undo DAC configuration
- DAC0_CTRLA = 0;
- DAC0_DATA = 0;
+ DAC0_CTRLA = 0;
+ DAC0_DATA = 0;

```

```

        // undo ADC configuration
-       ADC0_CTRLA = 0;
-       ADC0_MUXPOS = 0;
+       ADC0_CTRLA = 0;
+       ADC0_MUXPOS = 0;
        // undo additional setting from test 2
-       ADC0_CTRLC = 0;
+       ADC0_CTRLC = 0;
    }

-static inline void classb_undo_configuration_of_window_mode_test(void)
-{
-       // undo TCA0 configuration
-       TCA0_SINGLE_INTCTRL = 0;
-       TCA0_SINGLE_PER = 0;
-       TCA0_SINGLE_CMP0 = 0;
-       // undo event configuration
-       EVSYS_SYNCCH0 = 0;
-       EVSYS_ASYNCUSER1 = 0;
-       // undo ADC configuration
-       ADC0_WINLT = 0;
-       ADC0_WINHT = 0;
-       ADC0_CTRLB = 0;
-       ADC0_EVCTRL = 0;
-       ADC0_INTCTRL = 0;
-       ADC0_CTRLA = 0;
-       ADC0_MUXPOS = 0;
-}
-
-/*!
- * \internal
- * \brief This test writes a value to the DAC, and checks that the ADC readouts are
- *        within the expected range.
- */
-static inline void classb_dac_adc_test(void)
-{
-       classb_base_configuration_of_DAC_and_ADC();
-
-       classb_DAC_ADC_test1();
-
-       //DAC_ADC_test2();
-
-       //adc_dac_window_mode_test();
-}
/** @}*/
#endif /* CLASSB_ADC_DAC_H_ */
\ No newline at end of file

```

Diff of main_analog.c:

```

----- CLASSB_ANALOG/applications/CLASSB_ANALOG/main_analog.c -----
index de178e0..4c5bba2 100644
@@ -2,7 +2,7 @@
/**
 * \file
 *
- * \version 1.0
+ * \version 1.1
 *
 * \brief
 *        This in an demo application for the ADC and DAC tests.
@@ -13,18 +13,18 @@
 *        than expected. If the example is modified to use test 2 or the window test the
 *        range used must be updated.
 *
- * \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
+ *
@@ -44,10 +44,10 @@
+ * 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
@@ -57,35 +57,72 @@
+ * DAMAGE.
+ */

#include "avr_compiler.h"
#include "classb_compiler.h"
#include "classb_analog.h"
#include "avr_interrupt.h"
#include "Tiny817xpro.h"
#include "oled1_xpro_attiny817.h"
+
+// This defines the upper and lower limit of the ADC reading of test 2.
+// The result from the ADC will be:
+// * 10 bit ADC with reference VCC.
+// * VCC (1.8V ~ 5.0V). example for VCC = 5.0 V.
+// * DAC mid point (0x80 = 0.275) with reference 0.55V
+// * Nominal expected value ( (0.275 * 1024) / 5.0 = 56 )
+
+// Possible VDD used in application
+#define APP_1V8_VALUE_2 1.8
+#define APP_3V3_VALUE_2 3.3
+#define APP_5V0_VALUE_2 5.0
+
+#define APP_ADC_SIZE 0x400
+
+// DAC output set to midpoint of 8-bit resolution (0x80 @ 0.55V Ref)
+#define APP_DAC_VALUE_2 0.28
+
+// VDD down scaling factor
+#define APP_VDD_CONV_VALUE_2 (uint16_t) ( ( APP_DAC_VALUE_2 * APP_ADC_SIZE ) / APP_3V3_VALUE_2 )
+#define APP_ADC_RES_2_LIMIT_H (uint16_t) ( APP_VDD_CONV_VALUE_2 * 1.20 )
+#define APP_ADC_RES_2_LIMIT_L (uint16_t) ( APP_VDD_CONV_VALUE_2 * 0.80 )

//! \brief Global error flag
NO_INIT volatile uint8_t classb_error;

int main(void)
{

```



```

- // LED 1 is used to indicate error. When ON there is no error when OFF this indicates an error
+ // LED1 classb_ADC_DAC_test1() (ON = PASS, OFF = FAIL)
+ configure_LED1();
- // The button is used to induce an error. See below ISR
+
+ // LED2 classb_ADC_DAC_test2() (ON = PASS, OFF = FAIL)
+ configure_LED2();
+
+ // Setup buttons so a press triggers ISR below.
+ configure_button1();
+ configure_button2();

+ // initialize classb_error to 0
+ classb_error = 0;

+ // Initialize ADC and DAC
+ classb_ADC_DAC_base_config();

+ // Interrupts enabled in the system
+ sei();
- while(!classb_error)
+ while(!classb_error)
+ {
-     classb_dac_adc_test();
- };
+     classb_ADC_DAC_test1();
+     if (classb_error)
+     {
+         LED1_OFF;
+         break;
+     }
+     classb_ADC_DAC_test2(APP_ADC_RES_2_LIMIT_H, APP_ADC_RES_2_LIMIT_L);
+     if (classb_error)
+     {
+         LED2_OFF;
+         break;
+     }
+ };

- // If this is executed there has been an error.
- // Turn off LED
- LED1_OFF;
- while(1);
}

/*! \brief Interrupt for Button 1 press
@@ -95,8 +132,19 @@ int main(void)
*/
ISR(PORTA_PORT_vect)
{
- //Reset interrupt
- BUTTON1_PORT.INTFLAGS = BUTTON1_PIN;
- //Change reference voltage to ADC
- VREF_CTRLA = VREF_ADC0REFSEL_2V5_gc;
+ if (BUTTON1_PORT.INTFLAGS == BUTTON1_PIN)
+ {
+     // Change reference voltage to ADC
+     VREF_CTRLA |= VREF_ADC0REFSEL_1V1_gc;
+     // Reset interrupt
+     BUTTON1_PORT.INTFLAGS = BUTTON1_PIN;
+ }
+
+ if (BUTTON2_PORT.INTFLAGS == BUTTON2_PIN)
+ {
+     // Change reference voltage to DAC & ADC
+     VREF_CTRLA |= ((uint8_t) VREF_ADC0REFSEL_1V1_gc | (uint8_t) VREF_DAC0REFSEL_1V1_gc);
+     // Reset interrupt
+     BUTTON2_PORT.INTFLAGS = BUTTON2_PIN;
+ }
}

```

}

Diff of CLASSB_ANALOG.rst:

----- CLASSB_ANALOG/documentation/CLASSB_ANALOG.rst -----

index 25aac5d..bf1ab06 100644

@@ -6,27 +6,50 @@ This in an demo application for the ADC and DAC tests.

The application is made to execute on a ATtiny817 Xplained Pro
with the OLED1 Xplained connected to the EXT1 header.

+This demo runs continuously 2 ADC/DAC tests, with 2 buttons
+to induce a failure in each test respectively. Upon failure
+the corresponding LED is turned off. A device reset will clear
+error and restart tests.

+

+Both tests are similar in that the DAC is input to the ADC.

+

+Test 1 both DAC and ADC use the internal bandgap based VREF peripheral as
+reference. A change in ADC reference level, from the expected one, will
+be reflected in an ADC measurement outside the expected limits.

+

+Test 2 DAC uses the internal bandgap based VREF peripheral as reference, but
+the ADC uses external VDD as reference. A change of the DAC internal
+reference level or VDD, will be reflected in an ADC measurement outside
+the expected limits. Note that the limits are VDD dependent, and thus
+are provided by the application.

-This Example runs ADC test 1 where the internal bandgap is used as reference for both the
-DAC and the ADC. To get this test to fail an interrupt has been added to
-change the reference voltage to the ADC. This causes the ADC result to be smaller
-than expected. If the example is modified to use test 2 or the window test the
-range used must be updated.

Related documents / Application notes

-This application is not 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

- 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.