

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

Files removed/replaced:

- CLASSB\_CRC/applications/CLASSB\_CRC/error\_handler.h replaced by CLASSB/classb\_error\_handler.h
- CLASSB\_CRC/applications/CLASSB\_CRC/Tiny817xpro.h replaced by CLASSB/utls/oled1\_xpro\_attiny817.h
- CLASSB\_CRC/applications/CLASSB\_CRC/avr\_compiler.h replaced by CLASSB/utls/classb\_compiler.h

Files with diff/changelog:

- CLASSB\_CRC/applications/CLASSB\_CRC/CRC\_16bit\_alg.h
- CLASSB\_CRC/applications/CLASSB\_CRC/CRC\_16bit\_lookup.h
- CLASSB\_CRC/applications/CLASSB\_CRC/CRC\_32bit\_alg.h
- CLASSB\_CRC/applications/CLASSB\_CRC/CRC\_32bit\_lookup.h
- CLASSB\_CRC/applications/CLASSB\_CRC/classb\_crc.h
- CLASSB\_CRC/applications/CLASSB\_CRC/classb\_crc\_hw.h
- CLASSB\_CRC/applications/CLASSB\_CRC/main\_crc.c
- CLASSB\_CRC/documentation/CLASSB\_CRC.rst

Diff of CRC\_16bit\_alg.h:

```
----- CLASSB_CRC/applications/CLASSB_CRC/CRC_16bit_alg.h -----
index 12bb770..38463c8 100644
@@ -2,23 +2,23 @@
/**
 * \file
 *
- * \version 1.0
+ * \version 1.1
 *
 * \brief
 *          16-Bit algorithm CRC implementation
 *
- * \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
@@ -130,4 +130,4 @@
@@ -130,4 +130,4 @@
@@ uint16_t CLASSB_CRC16_EEPROM_SW (eepromtr_t origDataptr, crcbytenum_t numBytes,
return remainder;
```

```

}

-#endif /* CRC_16BIT_ALG_H_ */
\ No newline at end of file
+#endif /* CRC_16BIT_ALG_H_ */

```

## Diff of CRC\_16bit\_lookup.h:

```

----- CLASSB_CRC/applications/CLASSB_CRC/CRC_16bit_lookup.h -----
index 825f3ab..3b63231 100644
@@ -2,23 +2,23 @@
/**
 * \file
 *
- * \version 1.0
+ * \version 1.1
 *
 * \brief
 *          16-Bit Lookup table CRC implementation
 *
- * \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
@@ -147,4 +147,4 @@ uint16_t CLASSB_CRC16_EEPROM_SW (eepromptr_t origDataptr, crcbytenum_t numBytes,

    return remainder;
}
-#endif /* CRC_16BIT_LOOCLKUP_H_ */
\ No newline at end of file
+#endif /* CRC_16BIT_LOOCLKUP_H_ */

```

## Diff of CRC\_32bit\_alg.h:

```

----- CLASSB_CRC/applications/CLASSB_CRC/CRC_32bit_alg.h -----
index 1731968..c2ebd42 100644
@@ -2,23 +2,23 @@

```

```

/**
 * \file
 *
- * \version 1.0
+ * \version 1.1
 *
 * \brief
 *
 *          32-Bit algorithm CRC implementation
 *
- * \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
@@ -117,4 +117,4 @@ uint32_t CLASSB_CRC32_EEPROM_SW (eeprom_ptr_t origDataptr, crcbytenum_t numBytes,
    return (remainder);
}

-#endif /* CRC_32BIT_ALG_H */
\ No newline at end of file
+#endif /* CRC_32BIT_ALG_H */

```

## Diff of CRC\_32bit\_lookup.h:

```

----- CLASSB_CRC/applications/CLASSB_CRC/CRC_32bit_lookup.h -----
index 854fa52..ded80e8 100644
@@ -2,23 +2,23 @@
/**
 * \file
 *
- * \version 1.0
+ * \version 1.1
 *
 * \brief
 *
 *          32-Bit Lookup table CRC implementation
 *
- * \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
@@ -187,4 +187,4 @@ uint32_t CLASSB_CRC32_EEPROM_SW (eeprom_ptr_t origDataptr, crcbytenum_t numBytes,

-#endif /* CRC_32BIT_LOOKUP_H */
\ No newline at end of file
+#endif /* CRC_32BIT_LOOKUP_H */

```

## Diff of classb\_crc.h:

```

----- CLASSB_CRC/applications/CLASSB_CRC/classb_crc.h -----
index b2fbb27..7b51b3c 100644
@@ -2,23 +2,23 @@
/**
* \file
*
- * \version 1.0
+ * \version 1.1
*
* \brief
*
*   Settings and defines used in connection with CRC memory tests.
*
* \par Application note:
- *   AVR1610: Guide to IEC60730 Class B compliance with TinyAVR 1-series
+ *   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
@@ -54,8 +54,8 @@
#ifndef __CRC_H__
#define __CRC_H__

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

/**
* \defgroup classb_crc CRC Tests

```

## Diff of classb\_crc\_hw.h:

```

----- CLASSB_CRC/applications/CLASSB_CRC/classb_crc_hw.h -----
index 3629f2e..10edea2 100644
@@ -1,22 +1,22 @@
/**
* \file
*
- * \version 1.0
+ * \version 1.1
*
* \brief Light weight driver for the CRC module
*
* \par Application note:
- * AVR1610: Guide to IEC60730 Class B compliance with TinyAVR 1-series
+ * 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
*
@@ -36,10 +36,10 @@
* 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
@@@ -52,9 +52,8 @@@
#ifndef __CRC_H_HW__
#define __CRC_H_HW__

-#include "avr_compiler.h"
+#include "classb_compiler.h"
#include "classb_crc.h"
-#include "avr/io.h"

/**
@@@ -64,13 +63,11 @@@
*/

-#define CRC_NMI_ENABLED
-
-//!! This enum contains the allowable settings for what regions of flash to scan
-//!! The size of regions is set in flash fuses. Boot section will always be one page or larger.
enum flash_region {classb_crc_hw_whole_flash_boot_app_code_data = 0x00,
-                classb_crc_hw_boot_app_code_section = 0x02,
-                classb_crc_hw_boot_section = 0x03};
+                classb_crc_hw_boot_app_code_section = 0x01,
+                classb_crc_hw_boot_section = 0x02};

-//!! \brief Sets CRCSCAN mode and region to scan
-//!!
@@@ -79,33 +76,17 @@@ enum flash_region {classb_crc_hw_whole_flash_boot_app_code_data = 0x00,
-//!! \note
-//!!
-//!! @param set_region Defines which flash region to scan
-#static inline void classb_crc_hw_set_flash_region_to_scan(enum flash_region set_region)
+static inline void classb_crc_hw_set_flash_region_to_scan(enum flash_region set_region)
{
-    CRCSCAN_CTRLB = set_region;
+    CRCSCAN.CTRLB = set_region;
}

-//!! \brief Start CRC scan
-//!!
-//!! This function starts the crcscan. It can be used with all modes.
-//!! \note
-//!! IF CRC_NMI_ENABLED is defined the ISR will be executed if the CRCSCAN finds an error.
-//!! If the NMI should not be used but the status register is checked instead. the ISR
-//!! should be removed/commented out to save space.
static inline void classb_crc_hw_start(void)
{
-#ifndef CRC_NMI_ENABLED
-    CRCSCAN_CTRLA = 0x03;
-#else
-    CRCSCAN_CTRLA = 0x01;
-#endif
-}
-
-//!! \brief Check if CRC scan is busy
-//!!
-//!! This function can be used in background mode to check if the CRC scan is busy or done.
-#static inline uint8_t crc_busy(void)
-{
-    return 0x01 & CRCSCAN_STATUS;
+    CRCSCAN.CTRLA = 0x03;
+}

#endif /* CRC_H_INCLUDED */

```

## Diff of main\_crc.c:

```
----- CLASSB_CRC/applications/CLASSB_CRC/main_crc.c -----
index 244d2fa..95a4fba 100644
@@ -2,7 +2,7 @@
/**
 * \file
 *
- * \version 1.1
+ * \version 1.2
 *
 * \brief
 * Demo application for the CRC tests.
@@ -15,7 +15,9 @@
 *
 *
 * If there are line breaks in the following remove it before copy paste:
 *
- * "src_cat "$(OutputDirectory)\$(OutputFileName).hex" -intel -crop 0 0x1FFE -fill 0xFF 0 0x1FFE -CRC16_Big_Endian
0x1FFE -broken -o "$(OutputFileName)_crc.hex" -intel -line-length=44"
+ * "src_cat "$(OutputDirectory)\$(OutputFileName).hex" -intel
+ * -crop 0 0x1FFE -fill 0xFF 0 0x1FFE -CRC16_Big_Endian 0x1FFE
+ * -broken -o "$(OutputFileName)_crc.hex" -intel -line-length=44"
 *
 * This will create a file caled "project name"_crc.hex in the
 * debug folder of the project. This is the file that has to be
@@ -36,18 +38,18 @@
 * he boot section has to be configured to be approximately the
 * size of the test program.
 *
- * \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
 *
@@ -67,10 +69,10 @@
 * 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
@@ -82,41 +84,35 @@

-#include "avr_compiler.h"
-#include "Tiny817xpro.h"
+#include "classb_compiler.h"
+#include "oled1_xpro_attiny817.h"
```

```

#include "classb_crc_hw.h"
#include "error_handler.h"
#include <avr/xmega.h>
#include <avr/interrupt.h>
#include "avr/pgmspace.h"
#include "classb_error_handler.h"
#include "ccp.h"

// Address pointer used to write to flash in order to provoke CRC error
volatile uint8_t *flash_ptr;
// Address to flash
#define ADDRES_IN_FLASH (uint8_t*)0x9100
uint8_t *CRC_error_ptr = (uint8_t*)0x9100;

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

int main(void)
{
- // Move interrupt vector back to bootsection
- _PROTECTED_WRITE(CPUINT_CTRLA, CPUINT_IVSEL_bm);
+ // Initialize error variable
+ classb_error = 0;
+ // Move interrupt vector location to bootsection
+ ccp_write_io((void *)&CPUINT_CTRLA, CPUINT_IVSEL_bm);
// Configure button used to induce error
configure_button1();
// Configure LED used to indicate error
configure_LED1();
// Configure CRCSCAN HW
- classb_crc_hw_set_flash_region_to_scan(classb_crc_hw_whole_flash_boot_app_code_data);
- // Initialize error variable
- classb_error = 0;
- // Set address to flash
- flash_ptr = ADDRES_IN_FLASH;
+ classb_crc_hw_set_flash_region_to_scan(classb_crc_hw_whole_flash_boot_app_code_data);
// Enable interrupts
sei();
- while (!classb_error)
- {
- //start flash scan
+ while (1) {
+ // Start flash scan
+ classb_crc_hw_start();
}
}
@@ -128,18 +124,21 @@ ISR(PORTA_PORT_vect)
// clear interrupt flag
BUTTON1_PORT.INTFLAGS = BUTTON1_PIN;

- // fill page buffer with data and set address in NVM controller
- *flash_ptr = 0x55;
+ // fill page buffer with data, this will also select the correct page address in the NVM controller
+ *CRC_error_ptr = 0x55;
// commit page buffer to page indicated by address.
- _PROTECTED_WRITE_SPM(NVMCTRL_CTRLA, NVMCTRL_CMD_PAGWRITE_gc);
+ ccp_write_spm((void *)&NVMCTRL_CTRLA, NVMCTRL_CMD_PAGWRITE_gc);
}

//! \brief CRC NMI error ISR
//! ISR for CRC Error. This interrupt can not be cleared once an error is detected.
//! All error handling must be done in the ISR. It takes a reset to clear interrupt
+//! Turn off LED1 to indicate NMI CRC scan fail interrupt
ISR(CRCSCAN_NMI_vect)
{
classb_error = 1;

+ LED1_OFF;
- while(1);

```



```
+     while (1)
+     ;
+ }
```

## Diff of CLASSB\_CRC.rst:

```
----- CLASSB_CRC/documentation/CLASSB_CRC.rst -----
index 88c7507..fe4a76d 100644
@@ -1,26 +1,30 @@
Introduction
=====

-Demo application for the CRC tests.
+Demo application for the Class B CRC tests.

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

-The HW CRC module expects the CRC value at the end of flash.
-The srec_cat command is used to generate a hex file containing
-the correct CRC value. This is done by adding the following
-string to the post build command
-(Go to Project->Properties >> Open "Build Event"):
-
-"srec_cat "$(OutputDirectory)$(OutputFileName).hex" -intel -crop 0 0x1FFE -fill 0xFF 0 0x1FFE -CRC16_Big_Endian 0x1FFE -
broken -o "$(OutputFileName)_crc.hex" -intel -line-length=44"
+When pushing Button 1 on the OLED1 Xplained board, the application
+will insert an error into the flash and the CRC scan will fail.
+This will trigger the CRCSCAN Non-Maskable Interrupt where LED1 will
+turn off. As this error is in flash, a reset is not enough to fix the
+issue, and the device needs to be reprogrammed to make the example
+pass again.

-There is an appnote and examples for how to use the CRCSCAN
-module with examples.
+The HW CRC module expects the CRC value at the end of flash.
+This is done by adding command line events as described in the
+GCC and IAR instructions below.

-This will create a file called "project name"_crc.hex in the
-debug folder of the project. This is the file that has to be
-used when running CRC scan. The original hex file will generate
-an error immediately after the first pass of the CRC scan.
+Atmel Studio (GCC) use the srec_cat command to generate a new hex
+file containing the correct CRC value. This will create a file named
+[project name]_crc.hex in the output folder of the project.
+This is the file that has to be programmed when running a CRC scan.
+The original hex with missing CRC value will generate an error at
+first pass of the CRC scan. The CRCSCAN application note linked
+below describes how to configure the srec_cat command for boot
+only and application only configurations.

Note:
In order to test the CRC module a write to flash is performed.
@@ -28,14 +32,17 @@ Note:
It is not possible to write from the boot section to the boot
section not from application data section to application data
section. For this reason the device fuse has to be configured.
- he boot section has to be configured to be approximately the
+ The boot section has to be configured to be approximately the
size of the test program. For this application BOOTEND = 0x04
will work.

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>>` \_  
+ - `CRCSCAN on Devices in the tinyAVR 1-Series  
<<http://www.microchip.com/wwwappnotes/appnotes.aspx?appnote=en599876>>` \_

#### Supported evaluation kit

-----  
@@ -43,12 +50,28 @@ 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 the application with the above post build command
- 4. Change the boot section fuse BOOTEND to 0x4 in the device programming dialog (CTRL+SHIFT+P)
- 5. Flash the generated crc\_crc.hex file into the device using the device programming dialog
- 6. Test if the test can fail by writing to flash using the button
- +3. Add the following post build command in Project Properties(Alt+F7)->Build Events:  
+  
+ \*\*\*"srec\_cat "\$ (OutputDirectory)\\\$(OutputFileName).hex" -intel -crop 0 0x1FFE -fill 0xFF 0 0x1FFE -CRC16\_Big\_Endian  
0x1FFE -broken -o "\$(OutputFileName)\_crc.hex" -intel -line-length=44\*\*\*  
+4. Build the application
- +5. Change the boot section fuse BOOTEND to 0x4 in the device programming dialog (CTRL+SHIFT+P)
- +6. Flash the generated \*\*[project name]\_crc.hex\*\* file into the device using the device programming dialog
- +7. Test if the test can fail by writing to flash using the button

+

+

##### +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. In Project Options(Alt+F7)->Linker do the following:  
+  
+ a) In Config, override default linker configuration file to [your\_project\_folder]/utils/\*\*iar\_cfgtiny817\_classB.xc|\*\*.  
+ b) In Extra Options, add the command line options \*\*\*-h'FF'(CODE)0-(...X\_FLASH\_END)\*\*\* and \*\*\*-J2,crc16,x,,,#0xffff\*\*\*.  
+4. Change the boot section fuse BOOTEND to 0x4 with your chosen programmer/debugger.
- +5. Build and flash into supported evaluation board.