

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

Files removed/replaced:

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

New files:

- CLASSB\_REGISTERS/applications/CLASSB\_REGISTERS/classb\_cpu\_gcc\_asm.s
- CLASSB\_REGISTERS/applications/CLASSB\_REGISTERS/classb\_cpu\_iar\_asm.s

Files with diff/changelog:

- CLASSB\_REGISTERS/applications/CLASSB\_REGISTERS/classb\_cpu.h
- CLASSB\_REGISTERS/applications/CLASSB\_REGISTERS/classb\_cpu\_gcc.c
- CLASSB\_REGISTERS/applications/CLASSB\_REGISTERS/classb\_cpu\_iar.c
- CLASSB\_REGISTERS/applications/CLASSB\_REGISTERS/main\_registers.c
- CLASSB\_REGISTERS/documentation/CLASSB\_REGISTERS.rst

## Diff of classb\_cpu.h:

```
----- CLASSB_REGISTERS/applications/CLASSB_REGISTERS/classb_cpu.h -----
index a1b4bc8..9899c69 100644
@@ -2,23 +2,23 @@
/**
 * \file
 *
- * \version 1.0
+ * \version 1.1
 *
 * \brief
 *   Settings and definitions for the CPU registers 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
```

```

@ @ -55,7 +55,7 @ @
#ifndef CLASSB_CPU_H_
#define CLASSB_CPU_H_
#include "error_handler.h"
#include "classb_error_handler.h"

/*! \defgroup classb_registers CPU Register test
@ @ -63,26 +63,23 @ @
/*! \brief This is the self-test module for the registers in the CPU.
/*!
/*! The CPU registers are tested for stuck bits and some coupling faults.
-/*! The test is executed by calling \ref classb_register_test(). The test procedure
-/*! consists of five steps per register:
-/*! -# Write 0x55 to register
+/*! The test is executed by calling \ref classb_register_test().
+/*! The test procedure is written in assembly to have full control over the registers.
+/*!
+/*! -# Store all register values to stack
/*! -# Write 0xAA to register
/*! -# Verify content (0xAA) of register
/*! -# Write 0x55 to register
/*! -# Verify content (0x55) of register
+/*! -# Restore all registers to original value
/*!
-/*! Non-destructive testing is done on the R-registers that are stated to
-/*! need preservation in the compiler documentation and on all IO registers.
-/*! Further, this test returns its result using a register. Therefore, the
-/*! register used to return the value, the stack pointer registers or those
+/*! This test returns its result using a register. Therefore, the
+/*! register used to return the value, the stack pointer registers, status register or those
/*! used within the test (e.g. to save values) are critical in the sense
/*! that the test cannot be executed correctly unless these registers are
/*! working.
/*!
-/*! The error handler for the test is \ref CLASSB_ERROR_HANDLER_REGISTERS().
-/*! It is possible to configure independently the behavior of the test with
-/*! respect to failure in critical and non-critical registers (see \ref CLASSB_ERROR_CRIT
-/*! and \ref CLASSB_ERROR_NON_CRIT).
/*!
/*! Given that GCC and IAR make use of the CPU registers in a different manner,
/*! we have included a compiler-specific function. This reduces execution time
@ @ -118,19 +115,7 @ @
 * non-critical register faults would hang the device.
 */
#define CLASSB_ERROR_NON_CRIT CLASSB_LABEL2
-/*@}
-
-/*! \internal \defgroup reg_check Symbols used to check whether some registers are present.
-/*!
-/*! \brief These symbols are used to check indirectly whether RAMPx and EIND
-/*! registers are present.
-/*!
-/*! RAMPx and EIND registers are present if the memory sizes are large enough.
-/*! It seems like the register constants are defined in the header file even
-/*! if those registers are not present in the device. Therefore, it is
-/*! necessary to check it indirectly through the memory size.
-/*@{
#define __DOXYGEN__

#else
@ @ -148,159 +133,6 @ @
/*! \brief In order to simplify the code, the assembly code is interfaced through macros.
/*@{

-/*! \internal \brief Save R register content to R31.
- *
- * \param reg R-register to store in R31.
- */

```

```

-#define CLASSB_RegStore_R(reg) \
-    ASSEMBLY(\
-        "mov r31, " #reg "\n" \
-    )
-
-/*! \internal \brief Load R register content from R31.
- *
- * \param reg R-register to restore from R31.
- */
-#define CLASSB_RegRestore_R(reg) \
-    ASSEMBLY(\
-        "mov " #reg ", r31\n" \
-    )
-
-/*! \internal \brief Set high R-register to specified value.
- *
- * \param reg R-register to set (R16-R31).
- * \param value Value to set R-register to.
- */
-#define CLASSB_RegSet_R_HI(reg,value) \
-    ASSEMBLY(\
-        "ldi " #reg ", " #value "\n" \
-    )
-
-/*! \internal \brief Set low R-register to specified value.
- *
- * \param reg R-register to set (R0-R15).
- * \param value Value to set R-register to.
- *
- * \note The value is copied into the specified register via R30.
- */
-#define CLASSB_RegSet_R_LO(reg,value) \
-    ASSEMBLY(\
-        "ldi r30, " #value "\n" \
-        "mov " #reg ", r30\n" \
-    )
-
-/*! \internal \brief Set IO-register to specified value.
- *
- * \param reg IO-address of register to set.
- * \param value Value to set register to.
- *
- * \note The value is copied into the specified register via R30.
- */
-#define CLASSB_RegSet_IO(reg,value) \
-    ASSEMBLY(\
-        "ldi r30, " #value "\n" \
-        "out " #reg ", r30\n" \
-    )
-
-/*! \internal \brief Clear R register.
- *
- * \param reg R-register to clear.
- *
- * \note This macro works for any R-register (R0-R31).
- */
-#define CLASSB_RegClear_R(reg) \
-    ASSEMBLY(\
-        "clr " #reg "\n" \
-    )
-
-/*! \internal \brief Clear IO-register.
- *
- * \param reg IO-address of register to clear.
- *
- * \note The IO-register is cleared via R30.
- */
-#define CLASSB_RegClear_IO(reg) \
-    ASSEMBLY(\
-        "clr r30\n" \

```

```

- "out " #reg ", r30 \n" \
- )
-
-/*! \internal \brief Store IO-register content.
- *
- * \param reg IO-address of register to store.
- *
- * \note The value of the IO-register is stored in R31.
- */
-#define CLASSB_RegStore_IO(reg) \
- ASSEMBLY( \
- "in r31, " #reg " \n" \
- )
-
-/*! \internal \brief Restore IO-register content.
- *
- * \param reg IO-address of register to restore.
- *
- * \note The value of the IO-register is restored from R31.
- */
-#define CLASSB_RegRestore_IO(reg) \
- ASSEMBLY( \
- "out " #reg ", r31 \n" \
- )
-
-/*! \internal \brief Test R16-31 with specified value, jump to specified label on fault
- *
- * \param reg Register to test (R16..R31).
- * \param value Value to test register with.
- * \param label Label to jump to if test fails.
- */
-#define CLASSB_RegTest_R_HI(reg,value,label) \
- ASSEMBLY( \
- "ldi " #reg ", " #value " \n" \
- "cpi " #reg ", " #value " \n" \
- "breq A_" STRINGIZE(LABEL(reg,value,__LINE__)) " \n" \
- "jmp " STRINGIZE(label) " \n" \
- "A_" STRINGIZE(LABEL(reg,value,__LINE__)) ": \n" \
- )
-
-/*! \internal \brief Test R0-15 with specified value, jump to specified label on fault
- *
- * \note R30 is used in the copying values to the register.
- *
- * \param reg Register to test (R0..R15).
- * \param value Value to test register with.
- * \param label Label to jump to if test fails.
- */
-#define CLASSB_RegTest_R_LO(reg,value,label) \
- ASSEMBLY( \
- "ldi r30, " #value " \n" \
- "mov " #reg ", r30 \n" \
- "mov r30, " #reg " \n" \
- "cpi r30, " #value " \n" \
- "breq A_" STRINGIZE(LABEL(reg,value,__LINE__)) " \n" \
- "jmp " STRINGIZE(label) " \n" \
- "A_" STRINGIZE(LABEL(reg,value,__LINE__)) ": \n" \
- )
-
-/*! \internal \brief Test IO register with specified value, jump to specified label on fault
- *
- * \note R30 is used in the copying of values to the register.
- *
- * \param reg Address of IO register to test.
- * \param value Value to test register with.
- * \param label Label to jump to if test fails.
- */
-#define CLASSB_RegTest_IO(reg,value,label) \
- ASSEMBLY( \
- "ldi r30, " #value " \n" \

```

```

- "out " #reg ", r30 \n" \
- "in r30, " #reg " \n" \
- "cpi r30, " #value " \n" \
- "breq A_" STRINGIZE(LABEL(reg,value,__LINE__)) " \n" \
- "jmp " STRINGIZE(label) " \n" \
- "A_" STRINGIZE(LABEL(reg,value,__LINE__)) ": \n" \
- )
-//@}
-
//! \defgroup reg_func Functions
//! \brief This is the self-test for CPU registers.
//@{

```

## Diff of classb\_cpu\_gcc.c:

```

----- CLASSB_REGISTERS/applications/CLASSB_REGISTERS/classb_cpu_gcc.c -----
index d35c657..9432422 100644
@@ -2,23 +2,23 @@
/**
 * \file
 *
- * \version 1.0
+ * \version 1.1
 *
 * \brief This file contains the function definition for a CPU register test
 * that is compatible with AVR GCC.
 *
 * \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
@@ -51,12 +51,14 @@
 * DAMAGE.
 */

-#include "avr_compiler.h"
+#include "classb_compiler.h"
#include "classb_cpu.h"
#include <stdbool.h>

```

```

//! \cond GCC

+ #if defined (__GNUC__)
+
+ /*
+  * \brief CPU register self-diagnostic routine for AVR GCC.
+  *
+  * @ @ -68,284 +70,34 @ @
+  *   -# Critical registers
+  *   -# Return value register: R24
+  *   -# stack pointer: SPL/SPH
+  *   -# register %file: R31 -- R30
+  *   -# Non-critical registers
+  *   -# register %file: R29 -- R25, R23 -- R0
+  *   -# extended addressing registers: RAMPD/X/Y/Z, EIND
+  *   -# registerfile: R31 -- R30
+  *   -# status register: SREG (except interrupt flag)
+  *   -# Non-critical registers
+  *   -# register file: R0 -- R23, R25 -- R29
+  *
+  * \retval false No register fault detected.
+  * \retval true Register fault detected.
+  *
+  * \note Interrupts must be disabled during this test.
+  */
+
+ extern void Register_test_GCC_asm();
+
+ bool classb_register_test(void)
+ {
+     -
+     - // The error flag used locally
+     -     register bool error asm("r24") = true;
+     -
+     - // Test R24 first because it is the return register!
+     -     CLASSB_RegSet_R_HI(R24,0x55);
+     -     CLASSB_RegTest_R_HI(R24,0xAA,CLASSB_ERROR_CRIT);
+     -     CLASSB_RegTest_R_HI(R24,0x55,CLASSB_ERROR_CRIT);
+     -
+     - // Then test R31 and R30 because they are used in the rest of the tests.
+     -     CLASSB_RegSet_R_HI(R31,0x55);
+     -     CLASSB_RegTest_R_HI(R31,0xAA,CLASSB_ERROR_CRIT);
+     -     CLASSB_RegTest_R_HI(R31,0x55,CLASSB_ERROR_CRIT);
+     -
+     -     CLASSB_RegSet_R_HI(R30,0x55);
+     -     CLASSB_RegTest_R_HI(R30,0xAA,CLASSB_ERROR_CRIT);
+     -     CLASSB_RegTest_R_HI(R30,0x55,CLASSB_ERROR_CRIT);
+     -
+     - // SPH
+     -     CLASSB_RegStore_IO(0x3D);
+     -     CLASSB_RegSet_IO(0x3D,0x55);
+     -     CLASSB_RegTest_IO(0x3D,0xAA,CLASSB_ERROR_CRIT);
+     -     CLASSB_RegTest_IO(0x3D,0x55,CLASSB_ERROR_CRIT);
+     -     CLASSB_RegRestore_IO(0x3D);
+     -
+     - // SPL
+     -     CLASSB_RegStore_IO(0x3E);
+     -     CLASSB_RegSet_IO(0x3E,0x55);
+     -     CLASSB_RegTest_IO(0x3E,0xAA,CLASSB_ERROR_CRIT);
+     -     CLASSB_RegTest_IO(0x3E,0x55,CLASSB_ERROR_CRIT);
+     -     CLASSB_RegRestore_IO(0x3E);
+     -
+     - // Test R29-R28 non-destructively.
+     -     CLASSB_RegStore_R(R29);
+     -     CLASSB_RegSet_R_HI(R29,0x55);
+     -     CLASSB_RegTest_R_HI(R29,0xAA,CLASSB_ERROR_NON_CRIT);
+     -     CLASSB_RegTest_R_HI(R29,0x55,CLASSB_ERROR_NON_CRIT);
+     -     CLASSB_RegRestore_R(R29);
+     -
+ }

```

```

- CLASSB_RegStore_R(R28);
- CLASSB_RegSet_R_HI(R28,0x55);
- CLASSB_RegTest_R_HI(R28,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R28,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R28);
-
- // Test R27-R25, R23-R18 destructively.
- CLASSB_RegSet_R_HI(R27,0x55);
- CLASSB_RegTest_R_HI(R27,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R27,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_HI(R26,0x55);
- CLASSB_RegTest_R_HI(R26,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R26,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_HI(R25,0x55);
- CLASSB_RegTest_R_HI(R25,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R25,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_HI(R23,0x55);
- CLASSB_RegTest_R_HI(R23,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R23,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_HI(R22,0x55);
- CLASSB_RegTest_R_HI(R22,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R22,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_HI(R21,0x55);
- CLASSB_RegTest_R_HI(R21,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R21,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_HI(R20,0x55);
- CLASSB_RegTest_R_HI(R20,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R20,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_HI(R19,0x55);
- CLASSB_RegTest_R_HI(R19,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R19,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_HI(R18,0x55);
- CLASSB_RegTest_R_HI(R18,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R18,0x55,CLASSB_ERROR_NON_CRIT);
-
- // Test R17-R0 non-destructively.
- CLASSB_RegStore_R(R17);
- CLASSB_RegSet_R_HI(R17,0x55);
- CLASSB_RegTest_R_HI(R17,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R17,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R17);
-
- CLASSB_RegStore_R(R16);
- CLASSB_RegSet_R_HI(R16,0x55);
- CLASSB_RegTest_R_HI(R16,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R16,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R16);
+     volatile register bool error asm("r30") = false;

- CLASSB_RegStore_R(R15);
- CLASSB_RegSet_R_LO(R15,0x55);
- CLASSB_RegTest_R_LO(R15,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R15,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R15);
+     Register_test_GCC_asm();
+
+     if (error == true) {
+         CLASSB_ERROR_HANDLER_REGISTERS();
+     }

- CLASSB_RegStore_R(R14);
- CLASSB_RegSet_R_LO(R14,0x55);

```

```

- CLASSB_RegTest_R_LO(R14,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R14,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R14);
-
- CLASSB_RegStore_R(R13);
- CLASSB_RegSet_R_LO(R13,0x55);
- CLASSB_RegTest_R_LO(R13,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R13,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R13);
-
- CLASSB_RegStore_R(R12);
- CLASSB_RegSet_R_LO(R12,0x55);
- CLASSB_RegTest_R_LO(R12,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R12,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R12);
-
- CLASSB_RegStore_R(R11);
- CLASSB_RegSet_R_LO(R11,0x55);
- CLASSB_RegTest_R_LO(R11,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R11,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R11);
-
- CLASSB_RegStore_R(R10);
- CLASSB_RegSet_R_LO(R10,0x55);
- CLASSB_RegTest_R_LO(R10,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R10,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R10);
-
- CLASSB_RegStore_R(R9);
- CLASSB_RegSet_R_LO(R9,0x55);
- CLASSB_RegTest_R_LO(R9,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R9,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R9);
-
- CLASSB_RegStore_R(R8);
- CLASSB_RegSet_R_LO(R8,0x55);
- CLASSB_RegTest_R_LO(R8,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R8,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R8);
-
- CLASSB_RegStore_R(R7);
- CLASSB_RegSet_R_LO(R7,0x55);
- CLASSB_RegTest_R_LO(R7,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R7,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R7);
-
- CLASSB_RegStore_R(R6);
- CLASSB_RegSet_R_LO(R6,0x55);
- CLASSB_RegTest_R_LO(R6,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R6,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R6);
-
- CLASSB_RegStore_R(R5);
- CLASSB_RegSet_R_LO(R5,0x55);
- CLASSB_RegTest_R_LO(R5,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R5,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R5);
-
- CLASSB_RegStore_R(R4);
- CLASSB_RegSet_R_LO(R4,0x55);
- CLASSB_RegTest_R_LO(R4,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R4,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R4);
-
- CLASSB_RegStore_R(R3);
- CLASSB_RegSet_R_LO(R3,0x55);
- CLASSB_RegTest_R_LO(R3,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R3,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R3);
-

```



```

- CLASSB_RegStore_R(R2);
- CLASSB_RegSet_R_LO(R2,0x55);
- CLASSB_RegTest_R_LO(R2,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R2,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R2);
-
- CLASSB_RegStore_R(R1);
- CLASSB_RegSet_R_LO(R1,0x55);
- CLASSB_RegTest_R_LO(R1,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R1,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R1);
-
- CLASSB_RegStore_R(R0);
- CLASSB_RegSet_R_LO(R0,0x55);
- CLASSB_RegTest_R_LO(R0,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R0,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R0);
-
- // Test IO registers non-destructively.
- // RAMPD
-     #ifdef CLASSB_HAS_BIGMEM
- CLASSB_RegStore_IO(0x38);
- CLASSB_RegSet_IO(0x38,0x55);
- CLASSB_RegTest_IO(0x38,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_IO(0x38,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_IO(0x38);
-     #endif
-
- // RAMPX
-     #ifdef CLASSB_HAS_BIGMEM
- CLASSB_RegStore_IO(0x39);
- CLASSB_RegSet_IO(0x39,0x55);
- CLASSB_RegTest_IO(0x39,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_IO(0x39,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_IO(0x39);
-     #endif
-
- // RAMPY
-     #ifdef CLASSB_HAS_BIGMEM
- CLASSB_RegStore_IO(0x3A);
- CLASSB_RegSet_IO(0x3A,0x55);
- CLASSB_RegTest_IO(0x3A,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_IO(0x3A,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_IO(0x3A);
-     #endif
-
- // RAMPZ
-     #if defined(CLASSB_HAS_BIGMEM) || defined(CLASSB_HAS_BIGFLASH)
- CLASSB_RegStore_IO(0x3B);
- CLASSB_RegSet_IO(0x3B,0x55);
- CLASSB_RegTest_IO(0x3B,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_IO(0x3B,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_IO(0x3B);
-     #endif
-
- // EIND
-     #ifdef CLASSB_HAS_BIGFLASH
- CLASSB_RegStore_IO(0x3C);
- CLASSB_RegSet_IO(0x3C,0x55);
- CLASSB_RegTest_IO(0x3C,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_IO(0x3C,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_IO(0x3C);
-     #endif
-
- // SREG (Not interrupt flag!)
- CLASSB_RegStore_IO(0x3F);
- CLASSB_RegSet_IO(0x3F,0x55);
- CLASSB_RegTest_IO(0x3F,0x2A,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_IO(0x3F,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_IO(0x3F);

```



```

- * 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
@@ -51,297 +51,54 @@
* DAMAGE.
*/
#include "avr_compiler.h"
#include "classb_compiler.h"
#include "classb_cpu.h"
//! \cond IAR

//! \addtogroup classb_cpu_registers
//@{
#ifndef IAR
+
+ #if defined(__ICCAVR__)
+
+ /*
+ * \brief CPU register self-diagnostic routine for IAR.
+ *
+ * This function tests the CPU registers, preserving those registers specified
- * in the AVR GCC compiler documentation. Note that if there should be a fault
+ * in the IAR compiler documentation. Note that if there should be a fault
+ * in a register that needs preservation, its value would not be restored.
+
+ * Registers are tested in this order:
+ * -# Critical registers
+ * -# Return value register: R16
+ * -# stack pointer: SPL/SPH
- * -# register %file: R31 -- R30
- * -# Non-critical registers
- * -# register %file: R29 -- R17, R15 -- R0
- * -# extended addressing registers: RAMPD/X/Y/Z, EIND
+ * -# register: R31 -- R30
+ * -# status register: SREG (except interrupt flag)
+ * -# Non-critical registers
+ * -# register file: R29 -- R17, R15 -- R0
+
+ * \retval false No register fault detected.
+ * \retval true Register fault detected.
+
+ * \note Interrupts must be disabled during this test.
+ */
- bool classb_register_test(void)
+extern bool Register_test_IAR_asm();
+
+bool classb_register_test(void)
{
    // IAR should use R16 for this variable.
- bool error = true;
-
- // Test R16 first because it is the return register! FAIL -> hang device.
- CLASSB_RegSet_R_HI(R16,0x55);
- CLASSB_RegTest_R_HI(R16,0xAA,CLASSB_ERROR_CRIT);
- CLASSB_RegTest_R_HI(R16,0x55,CLASSB_ERROR_CRIT);
-
- // Then test R31 and R30 because they are used in the rest of the tests.
- CLASSB_RegSet_R_HI(R31,0x55);
- CLASSB_RegTest_R_HI(R31,0xAA,CLASSB_ERROR_CRIT);
- CLASSB_RegTest_R_HI(R31,0x55,CLASSB_ERROR_CRIT);
-
- CLASSB_RegSet_R_HI(R30,0x55);
- CLASSB_RegTest_R_HI(R30,0xAA,CLASSB_ERROR_CRIT);

```

```

- CLASSB_RegTest_R_HI(R30,0x55,CLASSB_ERROR_CRIT);
-
- // SPH
- CLASSB_RegStore_IO(0x3D);
- CLASSB_RegSet_IO(0x3D,0x55);
- CLASSB_RegTest_IO(0x3D,0xAA,CLASSB_ERROR_CRIT);
- CLASSB_RegTest_IO(0x3D,0x55,CLASSB_ERROR_CRIT);
- CLASSB_RegRestore_IO(0x3D);
-
- // SPL
- CLASSB_RegStore_IO(0x3E);
- CLASSB_RegSet_IO(0x3E,0x55);
- CLASSB_RegTest_IO(0x3E,0xAA,CLASSB_ERROR_CRIT);
- CLASSB_RegTest_IO(0x3E,0x55,CLASSB_ERROR_CRIT);
- CLASSB_RegRestore_IO(0x3E);
-
- // Test R29-R24 non-destructively.
- CLASSB_RegStore_R(R29);
- CLASSB_RegSet_R_HI(R29,0x55);
- CLASSB_RegTest_R_HI(R29,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R29,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R29);
-
- CLASSB_RegStore_R(R28);
- CLASSB_RegSet_R_HI(R28,0x55);
- CLASSB_RegTest_R_HI(R28,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R28,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R28);
-
- CLASSB_RegStore_R(R27);
- CLASSB_RegSet_R_HI(R27,0x55);
- CLASSB_RegTest_R_HI(R27,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R27,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R27);
-
- CLASSB_RegStore_R(R26);
- CLASSB_RegSet_R_HI(R26,0x55);
- CLASSB_RegTest_R_HI(R26,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R26,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R26);
-
- CLASSB_RegStore_R(R25);
- CLASSB_RegSet_R_HI(R25,0x55);
- CLASSB_RegTest_R_HI(R25,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R25,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R25);
-
- CLASSB_RegStore_R(R24);
- CLASSB_RegSet_R_HI(R24,0x55);
- CLASSB_RegTest_R_HI(R24,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R24,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R24);
-
- // Test R23-R17 destructively.
- CLASSB_RegSet_R_HI(R23,0x55);
- CLASSB_RegTest_R_HI(R23,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R23,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_HI(R22,0x55);
- CLASSB_RegTest_R_HI(R22,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R22,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_HI(R21,0x55);
- CLASSB_RegTest_R_HI(R21,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R21,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_HI(R20,0x55);
- CLASSB_RegTest_R_HI(R20,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R20,0x55,CLASSB_ERROR_NON_CRIT);
-

```

```

- CLASSB_RegSet_R_HI(R19,0x55);
- CLASSB_RegTest_R_HI(R19,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R19,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_HI(R18,0x55);
- CLASSB_RegTest_R_HI(R18,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R18,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_HI(R17,0x55);
- CLASSB_RegTest_R_HI(R17,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_HI(R17,0x55,CLASSB_ERROR_NON_CRIT);
+ bool error = false;

- // Test R15-R4 non-destructively.
- CLASSB_RegStore_R(R15);
- CLASSB_RegSet_R_LO(R15,0x55);
- CLASSB_RegTest_R_LO(R15,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R15,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R15);
+ error = Register_test_IAR_asm();

- CLASSB_RegStore_R(R14);
- CLASSB_RegSet_R_LO(R14,0x55);
- CLASSB_RegTest_R_LO(R14,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R14,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R14);
+ if (error == true) {
+ CLASSB_ERROR_HANDLER_REGISTERS();
+ }

- CLASSB_RegStore_R(R13);
- CLASSB_RegSet_R_LO(R13,0x55);
- CLASSB_RegTest_R_LO(R13,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R13,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R13);
+ return (error);

- CLASSB_RegStore_R(R12);
- CLASSB_RegSet_R_LO(R12,0x55);
- CLASSB_RegTest_R_LO(R12,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R12,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R12);
-
- CLASSB_RegStore_R(R11);
- CLASSB_RegSet_R_LO(R11,0x55);
- CLASSB_RegTest_R_LO(R11,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R11,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R11);
-
- CLASSB_RegStore_R(R10);
- CLASSB_RegSet_R_LO(R10,0x55);
- CLASSB_RegTest_R_LO(R10,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R10,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R10);
-
- CLASSB_RegStore_R(R9);
- CLASSB_RegSet_R_LO(R9,0x55);
- CLASSB_RegTest_R_LO(R9,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R9,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R9);
-
- CLASSB_RegStore_R(R8);
- CLASSB_RegSet_R_LO(R8,0x55);
- CLASSB_RegTest_R_LO(R8,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R8,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R8);
-
- CLASSB_RegStore_R(R7);
- CLASSB_RegSet_R_LO(R7,0x55);
- CLASSB_RegTest_R_LO(R7,0xAA,CLASSB_ERROR_NON_CRIT);

```

```

- CLASSB_RegTest_R_LO(R7,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R7);
-
- CLASSB_RegStore_R(R6);
- CLASSB_RegSet_R_LO(R6,0x55);
- CLASSB_RegTest_R_LO(R6,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R6,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R6);
-
- CLASSB_RegStore_R(R5);
- CLASSB_RegSet_R_LO(R5,0x55);
- CLASSB_RegTest_R_LO(R5,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R5,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R5);
-
- CLASSB_RegStore_R(R4);
- CLASSB_RegSet_R_LO(R4,0x55);
- CLASSB_RegTest_R_LO(R4,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R4,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_R(R4);
-
- // Test R3-R0 destructively.
- CLASSB_RegSet_R_LO(R3,0x55);
- CLASSB_RegTest_R_LO(R3,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R3,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_LO(R2,0x55);
- CLASSB_RegTest_R_LO(R2,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R2,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_LO(R1,0x55);
- CLASSB_RegTest_R_LO(R1,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R1,0x55,CLASSB_ERROR_NON_CRIT);
-
- CLASSB_RegSet_R_LO(R0,0x55);
- CLASSB_RegTest_R_LO(R0,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_R_LO(R0,0x55,CLASSB_ERROR_NON_CRIT);
-
- // Test IO registers non-destructively.
- // RAMPD
- #ifdef CLASSB_HAS_BIGMEM
- CLASSB_RegStore_IO(0x38);
- CLASSB_RegSet_IO(0x38,0x55);
- CLASSB_RegTest_IO(0x38,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_IO(0x38,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_IO(0x38);
- #endif
-
- // RAMPX
- #ifdef CLASSB_HAS_BIGMEM
- CLASSB_RegStore_IO(0x39);
- CLASSB_RegSet_IO(0x39,0x55);
- CLASSB_RegTest_IO(0x39,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_IO(0x39,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_IO(0x39);
- #endif
-
- // RAMPY
- #ifdef CLASSB_HAS_BIGMEM
- CLASSB_RegStore_IO(0x3A);
- CLASSB_RegSet_IO(0x3A,0x55);
- CLASSB_RegTest_IO(0x3A,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_IO(0x3A,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_IO(0x3A);
- #endif
-
- // RAMPZ
- #if defined(CLASSB_HAS_BIGMEM) || defined(CLASSB_HAS_BIGFLASH)
- CLASSB_RegStore_IO(0x3B);
- CLASSB_RegSet_IO(0x3B,0x55);

```

```

- CLASSB_RegTest_IO(0x3B,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_IO(0x3B,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_IO(0x3B);
- #endif
-
- // EIND
- #ifdef CLASSB_HAS_BIGFLASH
- CLASSB_RegStore_IO(0x3C);
- CLASSB_RegSet_IO(0x3C,0x55);
- CLASSB_RegTest_IO(0x3C,0xAA,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_IO(0x3C,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_IO(0x3C);
- #endif
-
- // SREG (Not interrupt flag!)
- CLASSB_RegStore_IO(0x3F);
- CLASSB_RegSet_IO(0x3F,0x55);
- CLASSB_RegTest_IO(0x3F,0x2A,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegTest_IO(0x3F,0x55,CLASSB_ERROR_NON_CRIT);
- CLASSB_RegRestore_IO(0x3F);
-
- ASSEMBLY(
- // Continue here if test was successful: clear return register and return.
- "clr R16 \n"
- "jmp END \n"
- // If a fault in return value or stack registers was detected, hang the device.
- "CLASSB_LABEL1: jmp CLASSB_LABEL1 \n"
- // If a fault was detected, set return register.
- "CLASSB_LABEL2: ldi R16, 0x01 \n"
- "END: \n"
- );
-
- // Handle global error. This will not be reached if there was a fault in
- // a) critical registers and CLASSB_ERROR_CRIT was defined as CLASSB_LABEL1.
- // b) non-critical registers and CLASSB_ERROR_NON_CRIT was defined as CLASSB_LABEL1.
- if (error == true) {
- CLASSB_ERROR_HANDLER_REGISTERS();
- }
-
- return (error);
- }
+
#endif
//@}

```

## Diff of main\_registers.c:

```

----- CLASSB_REGISTERS/applications/CLASSB_REGISTERS/main_registers.c -----
index a1d7f8b..68124a2 100644
@@ -2,7 +2,7 @@
/**
 * \file
 *
- * \version 1.0
+ * \version 1.1
 *
 * \brief This is a demo application for the CPU registers test.
 *
@@ -12,17 +12,17 @@
 *
 * user configuration.
 *
 * \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
*
@@ -42,10 +42,10 @@
* 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
@@ -54,20 +54,27 @@
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
* DAMAGE.
*#include "avr_compiler.h"
+#include "classb_compiler.h"
#include "classb_cpu.h"
#include "Tiny817xpro.h"
+#include "oled1_xpro_attiny817.h"
+

NO_INIT volatile uint8_t classb_error;

/*! \brief Example for register test
*
* This test checks the CPU registers for fault
+ *
+ * When compiling for GCC, remove classb_cpu_iar_asm.s from project.
+ *
+ * When compiling for IAR, exclude classb_cpu_gcc_asm.s from build.
*/
int main(void)
{
    // Initialize error variable
    classb_error = 0;
    // LED1 is used to indicate error
+
    configure_LED1();

    while(!classb_error)
@@ -77,4 +84,6 @@ int main(void)

    // If we reach here, an error has been detected. Turn off LED.
    LED1_OFF;
-}
\ No newline at end of file
+
+
+}

```

## Diff of CLASSB\_REGISTERS.rst:

```
----- CLASSB_REGISTERS/documentation/CLASSB_REGISTERS.rst -----
```



index f404b1f..ec88ed6 100644

@ @ -4,21 +4,22 @ @ Introduction

This is a demo application for the CPU registers test.

The application is made to execute on a ATtiny817 Xplained Pro

-with the OLED1 Xplained connected to the EXT1 header.

+with the OLED1 Xplained connected to the EXT1 header.

-The test runs continuously in a loop. If there is an error

-the CPU goes to an infinite loop or sets the global error flag

-\ref classb\_error. This depends on the kind of error and the

-user configuration.

+The test runs continuously in a loop. LED 1 will be on when everything is OK.

+If there is an error in the critical registers test, the CPU goes to an infinite loop or sets the global error flag.

+If there is an error in the non critical registers, LED 1 will be turned off.

-To provoke an error the test must be executed in a debug

-session and a breakpoint placed such that registers can be manually

+To provoke an error the test must be executed in a debug

+session and step in the code and stop so that registers can be manually altered.

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

@ @ -26,9 +27,22 @ @ 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. Remove classb\_cpu\_iar\_asm.s from project.

+4. 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. Set RSTACK depth to 0x40 to be able to safely pop all registers.

+4. Change linker file using iar\_cfgtiny817\_classB.xcl located in utils folder.

+5. Exclude classb\_cpu\_gcc\_asm.s from build.

+6. Build and flash into supported evaluation board.

