AVR998: Guide to IEC60730 Class B compliance with AVR microcontrollers

8-bit AVR® Microcontroller

Application Note

1. Overview

The International Electrotechnical Commission has introduced the IEC60730 referring to household appliances development. Annex H of IEC60730 describes three software classifications. ‘Class B’ software classification, refers to embedded firmware which is intended to prevent unsafe operation of controlled equipment.

Class B concerns a large range of products, including: dishwashers, washing machines, refrigerators, freezers, and cookers. According to IEC60730, household appliance manufacturers must now design their product following Class B rules.

Most of the products listed above use a single chip microcontroller with embedded memory and peripherals, this application note will deal with techniques that can be implemented on a CPU application.

According to IEC60730, Single channel application (usage of one microcontroller only) must have firmware designed based on one of the two techniques below.

- Single channel using functional test
- Single channel using periodic functional test

Single channel using functional test is the most popular mechanism used today and the easiest to implement, most appliance manufacturers now implement single channel with periodic self test in their new designs. This document deals with these two techniques and gives additional guidelines on how to comply with theClass B requirement when using AVR microcontrollers.

All of the features of AVR microcontrollers are tested during factory production. Some features are more sensitive to harsh environment than others. For example, CPU registers, Stack Pointer register and Status register, we consider sufficient to test for stuck bits these registers. On the other hand, a large area on devices such as RAM, Flash or EEPROM memories may need more complex verification algorithm. In this document we give the best compromise from Atmel’s point of view between feature sensitivity and the test to be implemented. Customers may decide to simplify or add extra tests according to their requirements.

7715B–AVR–04/08

2. Single Channel Using Functional Test

The order in which the test is performed is important and the tests must be run in the same order as it is presented below. For example, it is important to test the RAM memory before the Flash content because the Flash content test will use RAM memory to run. Some tests are also firmware intrusive (interrupt tests for example) and may be integrated in end application firmware. All errors in functional test may lead to stop the application and generate an error message.

Errors in periodic tests may lead to watchdog timeout in order to restart the application.

Test implementation is a compromise between startup time, used resources and protection level of the application.

Failure information can also be saved by the application to help failure diagnostics.

2.1 CPU Registers Test (General purpose register R0 to R25, X,Y,Z register test)

Purpose of the test : Control that no bits of these registers are stuck

The CPU registers test will test all the general purpose working registers from R0 to R31 and the Status register. The goal of this test is to detect if one bit of these registers is stuck to ‘1’ or to ‘0’. This test is done by successively writing, reading and checking 0x55 and 0xAA values into those registers. This test is done in assembler language just after Stack pointer initialization R29 to R31 are tested first, then these registers are used for testing R0 to R28.

2.2 Stack Pointer Register Test

Purpose of the test: Test that stack pointer bits are not stuck.

This test is done by successively write, read and check 0x55 and 0xAA value into these registers. No RET or RETI instructions are executed before this test.

2.3 Status Register Test

Purpose of the test : Control that all bits of the register are not stuck to ‘0’ or ‘1’.

The goal of this test is to detect if one bit of the register is stuck to ‘1’ or to ‘0’. This test is done by successively writing, reading and checking 0x55 and 0xAA value into these registers.

The Status register is used by previous tests, so we can consider three ways to reach this test. First, the Status register is fully functional, therefore all previous test results are good. Second, the Status register is wrong, therefore previous tests may have failed as the test itself was good and we cannot reach at this test. Third, the Status register is wrong and we go through the previous tests, but we will stop here. In all cases, any problem with one of the previous tests or the Status register test will lead to stop the program.

2.4 RAM Memory Test

Purpose of the test : Control that no bit of the RAM memory is stuck at ‘1’ or ‘0’.

The RAM test will test all the RAM memory locations. This test is done by successively writing, reading and checking 0x55 and 0xAA value into the RAM memory. The firmware sets a bit in a “RAM Test Status Register”. The program verifies that the bit is set at the end of the test.

Another test allowing to control the address encoder is to write the complement of the address of a location into RAM location. There is also a read and verify mechanism to control the values stored. The initial values of the RAM memory are (when doing periodic functional tests) saved before the test and restored after to enable stacked data to be retrieved after the test.

March B test is implemented (See “ANNEX : March B Test description” on page 7) to test the all RAM. The test is divided in two parts tested separately. Between each part test, the stack content is saved to the other part and the stack pointer points to the other part. The size and overlap of both parts are configurable and must be set according to physical memory organization.

2.5 Flash Memory Control

Purpose of the test: Control the flash content.

The Flash memory test must prevent any flash corruption. It can be done by a simple checksum of the flash content, or a more complex and time consuming Cyclic Redundancy Check. The reference result is stored in a particular place of the Flash memory at programming time, the calculated result is compared to this reference result at running time.

See document [5] listed in Section 7. on page 7

2.5.1 *Note on the use of the SPM instruction*

We recommend when it is possible to not use the SPM instruction in a harsh environment.

The SPM (Store Program Memory) is the instruction that allows to write into the Flash memory.The SPM instruction can access the entire Flash, including the boot load section. For example, if a function uses the SPM instruction, and a power loss occurs the Flash memory can be corrupted.

The protection level for the Boot Loader section against SPM intruction use can be selected by the Boot Loader Lock bits available on most of AVR products.

2.6 EEPROM Memory Test/Control

Purpose of the test : Control the EEPROM contents.

The EEPROM memory test must prevent any EEPROM corruption It can be done by a simple checksum of the EEPROM content, or a more complex and time consuming Cyclic Redundancy Check. As content of the EEPROM may vary during application life, the optimal solution should be to update the reference result at each write of the EEPROM to be able to have a dynamic comparison of the memory content during the product life.

See document [5] listed in Section 7. on page 7

2.7 Watchdog Test

Purpose of the test : Verify that the watchdog is functional

This test will check the functionality of the watchdog reset. Upon reset, the test will check if a reset occurs from watchdog reset. If not, the watchdog will be started and the test will wait until it occurs.

Note : The watchdog must be fuse enabled.

2.8 Interrupt functionality test

Purpose of the test : Test if the interrupt controller works correctly.

All interrupts which are not used by the application can be activated by the interrupt test function in order to check the correct behavior of the interrupt controller. The interruption is activated by software and the corresponding interrupt vector generates a signal to the interrupt test function.

The interrupts which are used by the application can be checked as described in the “Interrupt Periodic Event” chapter.

2.9 I/O registers

Purpose of the test : Test that I/O bits are not stuck at ‘1’ or ‘0’.

The I/0 test will test all the I/0. This test is done by successively writing, reading and checking 0x55 and 0xAA value into the I/0 registers**. This test is application dependant and can be run only if the hardware allows it.**

2.10 Clock frequency

Purpose of the test : Test of internal clock frequency.

To test the internal clock of the processor, one needs to have a reference clock available. For applications using external crystals, we can use the internal RC oscillator to check crystal presence and also to verify frequency oscillation of the external crystal.

Another way is to use a communication bus like SPI to measure the SPI clock duration. The result will be compared to the theoretical value.

The firmware sets a bit in the “Clock Test Status Register”. The program verifies that the bit is set at the end of the test.

2.11 Analog to Digital Converter

Purpose of the test : Test the ADC analog functions.

Free analog inputs can be wired to external known voltages to control the correct behavior of the ADC.

This test can also be done by using the internal bandgap reference as the ADC input. For example, the ATmega16 internal bandgap reference delivers a 1.22V voltage which can be regularly converted to test the ADC.

3. Single Channel Using Periodic Functional Test

Periodic tests are embedded with the firmware and regularly check that all is functioning normally. Prior to the execution of the application firmware, functional tests are executed. All errors in periodic tests may lead to watchdog time out in order to restart the application.

3.1 Interrupt Periodic Event

Purpose of the test : Check that interrupt occurs regularly in a defined lapse of time. In the same way, by the use of a counter, the test can detect if an interrupt occurs too frequently.

At each interrupt vector address, the firmware sets a bit in a user defined “IT Test Status Register”. The program verifies periodically that bits are set, then clears the “IT Test Status Register”. On errors a watchdog event is called.

Interruptions which are not used by the application can be used by the test to regularly check the interrupt controller.

3.2 Function Periodic Event

Purpose of the test : Check that some functions are called regularly in a defined lapse of time.

Using the same mechanism than interrupt test, each function sets a bit in “Function Event Status Register”. The program verifies periodically that bits are set, then clears the “ Function Event Status Register”. On errors, a watchdog event is called.

3.3 Error Event Detection

Purpose of the test : Check that every function gives a correct value on a regular basis.

This test is slightly different from the previous ones. This test checks that functions give a valid value. It will allow to restart if values are out of range or if any features experience trouble and do not send back the correct values (ADC conversion result for example). The periodic function will generate a watchdog event on wrong values or on several wrong values from specific functions.

4. Firmware Description

Attached to this application note, the reader can find an example written for the ATmega16 microcontroller. The purpose of this firmware is to show how to add a class B test to the main application. According to this application, the user can easily enhance the test by using the features that are not used by the application. This firmware can be adapted for other AVR microcontrollers.

Source files include a readme.html file which lauches an html description of the firmware.

The main.c file is the main of the application.

The low_level_init.c file embeds classB test functions which must be executed before the initialization of the C context.

The classB.c file contains the classB tests.

5. Additional Guidelines

5.1 External Communication

All communication must be secured by transfer redundancy when possible in order to enable the receiver to check for data corruption during transfers. Time out detection may be implemented to prevent failure and endless loops.

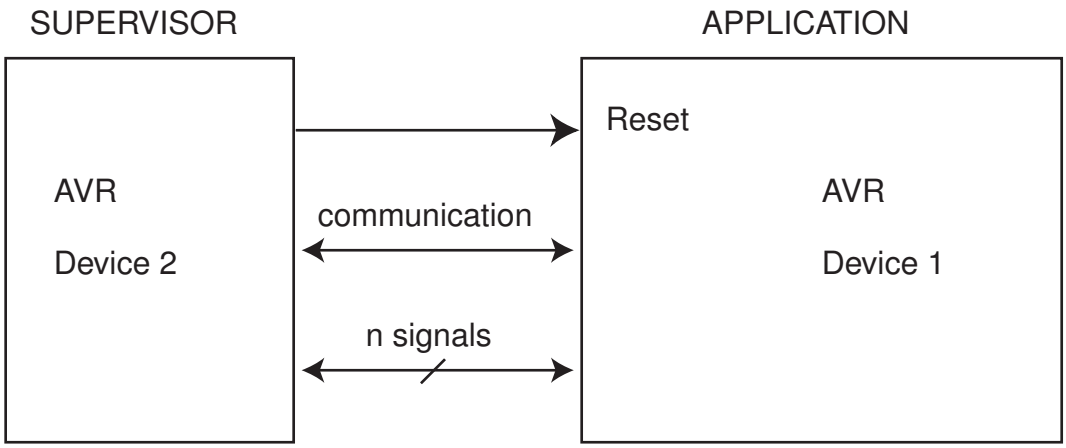
5.2 Watchdog

Intensive watchdog use prevents code fault, and error values from the firmware.

See ‘AVR132: Using the Enhanced Watchdog Timer’ listed in Section 7. on page 7.

6. Appendix

The best way to secure a design is to use double channel configuration with two devices checking each other. The following picture shows such implementation. Of course, double channel implementation is more expensive than single one, but it allows to have the best security coverage as each part can check what the other part is doing, and also be used as an external watchdog to prevent any oscillator issue (CPU or watchdog oscillator).



With double channel configuration, security can be increased. Examples below show what kind of functional interconnection can be implemented to secure the application.

- Device 2 can check that device 1 functions by communication means or any periodic external signal time out, then resets Device 1 on any issue.
- Device 1 can check that device 2 is running by any external signal time out too.
- When device 1 wants to perform any external action, it can send a message (by any means available such as SPI, TWI, UART….) to device 2 to confirm that the action is done. Then device 2 can check the action result and inform device 1, that the action is correctly done.
- Device 2 can also save the test result of device 1 for analysis in case of system failure.

Device 2 has a role of supervisor only, it can be very low cost. For example an ATtiny13 can be used to supervise an ATmega128.

7. Application Note References

[1] AVR040 : EMC Design Considerations

[2] AVR042 : AVR Hardware Design Considerations

[3] AVR132 : Using the Enhanced Watchdog Timer

[4] AVR180 : External Brown-Out Protection

[5] AVR236 : CRC check of Program Memory

[6] IEC60730 : Automatic electrical controls for household and similar use

[7] March B : Various literatures about testing Static Random Access Memories are available on Internet. The annex of this document describes the Class B test.

8. ANNEX : March B Test description

Static Random Access Memories' test is performed using MarchB algorithm. Using march conventions, this algorithm could be described as follow:

{**Down**(*w0*); **Up**(*r0,w1,r1,w0,r0,w1*); **Up**(*r1,w0,w1*); **Down**(*r1,w0,w1,w0*); **Down**(*r0,w1,w0*)}

Meaning:

1) Initialise the whole memory to '0' from last to first bit

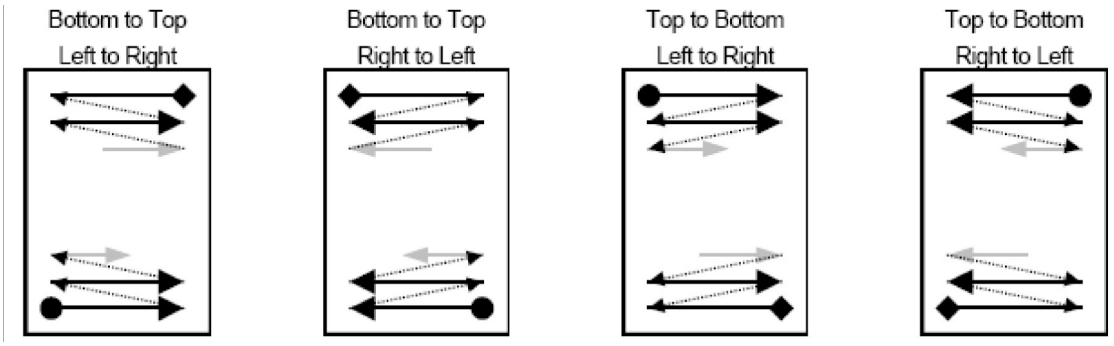
2) For each bit, from first to last one, read '0', write '1', read '1', write '0', read '0', write '1', then go to next bit.

3) For each bit, from first to last one, read '1', write '0', write '1', then go to next bit.

4) For each bit, from last to first one, read '1', write '0', write '1', write '0', then go to next bit.

5) For each bit, from last to first one, read '0', write '1', write '0', then go to next bit.

The walking order specified in the algorithm (Down, Up, Up, Down, Down) refers to physical addresses, not logical ones. It is relative to the first ‘Down’ which physically can correspond to four implementations:



Fault Coverage

This algorithm can catch the following miss-processes in SRAM areas:

Stuck At Faults (SAF): The bit is stuck at a state and cannot be written.

Transition Faults (TF): Stuck at a state, once in that state. By instance, the bit's value is '1', it can be written to '0' but once at '0' it cannot be set to '1' any more.

Inversions Coupling Faults (CFin): A transition on a bit inverts the state of a second bit.

Idempotent Coupling Faults (CFid): A transition on a bit forces a second bit to a certain state.

Coupling Faults Bridging (BF): Two bits are shorted. The resulting state of those bits is a logical AND or a logical OR between the previous states of those bits.

State Coupling Faults (SCF): A coupled cell is forced to a certain value only if a coupling cell is in a given state.

NOTE: In some very specific configurations of linked faults, some of those faults could be undetected. Those configurations are considered to impact less than 1 part per million.

Getting a fail during this test means that one of the previous faults was found without distinction of kind.

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
Enter Product Line E-mail

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature