R8C Family

IEC 60730-1 Self Test Sample Code API And Demo

REU05B0104-0106
Rev.1.06
Apr 12, 2010

Content

1. Introduction 2

2. Application software safety considerations 2

3. Summary of Demo 3

4. Register Protection 5

5. Guide to the API 5
API_TestCPU 6
API_TestRAMmemory 8
API_TestRamUserStack 10
API_TestROMmemory 11
API_TestMainClockStability 14
API_ReadADopenCircuit 17
API_CheckAD 19
API_DisableFlashRW 21
API_ReadPORresetVal 22
API_MonitorVccAndResetIfLow 24
API_SetPinSRtoReadIO 25
API_StartWDTwithSlowOCO 26
(API_TestWDT) 28

6. More functions 28

7. References and Bibliography 28
Other documents and presentations on testing according to IEC 60730 28

Website and Support 32

1. Introduction

All home appliances intended for the European Union market must today comply with the International Electrotechnical Committee's standard IEC 60730-1, "**Automatic electrical controls for household and similar use**". Both Europe and the U.S. have introduced IEC regulations defining safety requirements in the design of home appliances, but in the U.S. the standard UL1998 is still in use.

White goods microcontroller design teams therefore need to consider what is required in terms of hardware and software in order to comply with the IEC 607301 requirements. Appendix H of IEC 60730-1 specifically defines requirements for 'controls using software'.

This document explains example software test routines that have been developed as a help in fulfilling the standard's Appendix H software requirements for Class B controllers, 'control functions intended to prevent unsafe operation'. See table in 7(7).

Demonstration target

The example project was created for the R8C/35a. Most routines can be used for devices in the R8C/Tiny family. For M16C and H8, there exist corresponding CPU, RAM and ROM test source code. This source code is included in the project.

Tools used

- Renesas M16C Standard Toolchain version 5.45.00. (compiler, assembler, linker):
- E8a in-circuit debugger.

Time measurement conditions

Execution time measurements were done at 20 MHz CPU clock. No compiler optimizations. Measurement was taken with a scope. The 'Un' pin (p2_3) accessible on jumper JA214 on the RSKR8C35A was toggled before and after the function call.

2. Application software safety considerations

Apart from the test routines of the API, here are some safety considerations to take into account when writing a software application for equipment of Class B.

- Monitor for application state machine misbehavior, this could be done by checking for functions called in the wrong sequence using a safety checking state machine.
- Check for uncalled functions.
- Check for functions not finished.
- Ensure the circumstances are right before executing a function, e.g. 'Door locked before motor on'
- Dynamically enable / disable a function's ability to run.

3. Summary of Demo

The demo is made so the tests can be run one by one using a menu system. The idea is to present a simple interface to the user who will later decide how to use the API in his/her product.

Switch 1 on the RSKR8C35A selects a test in order shown in the table.

Switch 2 runs the test. Pass or fail will be displayed on line two of the LCD.

Table 1. List of software API tests and the order in which they appear in demo and are arranged in the source code.

API test function	Switch 1 LCD line 1	Switch 2 LCD line 2	Comments
API_TestCPU	"CPU test"	OK NOT OK	
API_TestRAMmemory	"RAM test"	OK NOT OK	Select type of RAM test.
API_TestRamUserStack	"StckUser"	OK NOT OK	
API_TestROMmemory	"ROM test"	OK NOT OK	Select type of flash memory test.
API_TestMainClockStability	"FtstTyp1" or "FtstTyp2"	OK NOT OK	Select freq. test type 1 or 2, permissible deviation etc.
API_CheckAD	"Adreftst"	OK NOT OK	
API_ReadADopenCircuit	"ADOCtest"	OK NOT OK	
API_DisableFlashRW	"DsblFlsh"	OK NOT OK	*..
API_ReadPORresetVal	"EnablPOR"	OK NOT OK	Follow procedure in source code to set up.
API_MonitorVccAndResetIfLow	"MonVcc "	OK NOT OK	*
API_SetPinSRtoReadIO	"IOread "	OK NOT OK	
API_StartWDTwithSlowOCO	"WDT "	OK NOT OK	*
(API_TestWDT)	"TestWDT "	CPU reset if watchdog is on Else "OK"	User test of working/non-working watchdog, not strictly part of API

*This function depends all or in part of the setting of the OFS register(s) which are not changed when running this function. This is statically set in flash memory when the device is programmed.

Code size

The total code size for the demo including start up code and LCD driver is given below:

RAM: 1262 Bytes

ROM data: 661 Bytes

Program code: 4985 Bytes

Table 2. Execution time and size for the tests. CPU clock frequency is 20 MHz.

API test function	API Execution time	Program code size [bytes]	Max stack usage in call hierarchy [bytes]
API_TestCPU	**51 us**	28	64
API_TestRAMmemory	For testing 86 bytes of demo RAM. RamTest March C 16Bit: **27.0 ms** RamTest March X 16Bit: **14.5 ms** RamTest March X WOM 16Bit: **4.0 ms**.	206	208
API_TestRamUserStack	Tests the user stack, 80h bytes. RamTest March C 16Bit: **138 us**	62	74, at this depth the stack is switched to a new location while testing the normal stack.
API_TestROMmemory	1kB (1024 bytes) of flash tested. Simple CRC-16 with lookup table: **12.4 ms** CRC16-CCITT using bit shift: **9.5 ms** CRC16-CCITT with static table: **11.0 ms**	41	66
API_TestMainClockStability	Method 1: Typical **21 ms** Min. **11 ms** Max. **31 ms**. Method 2: **1.6 us**	22	7
API_CheckAD	**12.0 us**	60	7
API_ReadADopenCircuit	**12.1 us**	55	7
API_DisableFlashRW	**4.8 us**	28	3
API_ReadPORresetVal	9 CPU cycles	4	3
API_MonitorVccAndResetIfLow	**7.0 us.**	48	3

API_SetPinSRtoReadIO	**3.4 us.**	14	3
API_StartWDTwithSlowOCO	**9.2 us.**	27	6

4. Register Protection

There is a safety feature to protect many SFR registers from unintentional writes. The protected registers are CM0, CM1, CM3, OCD, FRA0, FRA1, FRA2, FRA3, PM0 PM1, PD0, OCVREFCR, VCA2, VD1LS, VW0C, VW1C, and VW2C.

API functions that use any of these protected registers, have this noted in their respective chapter in this document.

5. Guide to the API

The following pages show each API test call's syntax and provides explanations.

API_TestCPU

Test of CPU Registers

The CPU test performs write and read back tests of the CPU registers.

Format

```
rslt_t API_TestCPU(void);
```

Return Values

0: OK.
1: NOT OK.

Properties

Prototyped in file iec_tests.c
Implemented in file iec_tests.h

Resources used

None (other than the MCU core).

API execution time

51 us.

Comments

All CPU registers are tested:

- General purpose registers (R0,R1, R2, R3,A0, A1, FB)
- Interrupt Table Registers (INTBL, INTBH)
- User Stack Pointer (USP), Interrupt Stack Pointer (ISP)
  Static Base register (SB)
- Program counter (PC)
- Flag Register (FLG)

They are tested by writes and subsequent readbacks of two different test patterns.

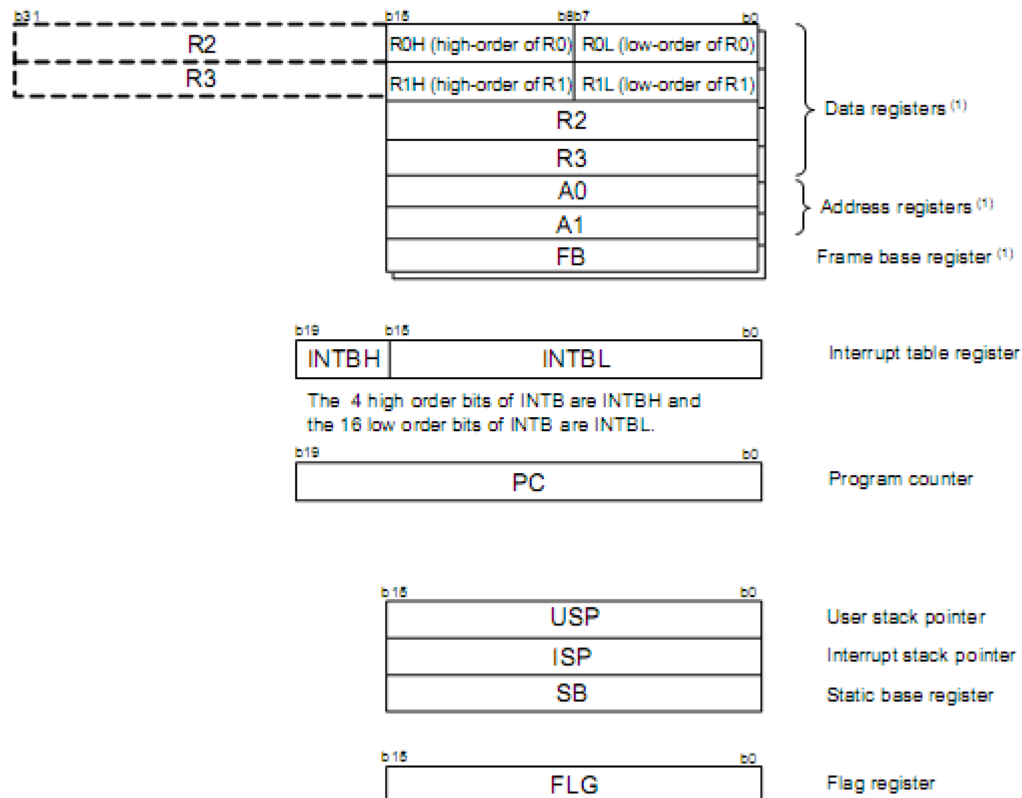

Figure 1. CPU registers of the R8C/35a. All are tested by API_TestCPU. Two different test patterns are written and read back to and from the registers.

API_TestRAMmemory

Test of RAM

RAM memory is tested by three alternative marching RAM tests over the memory range.

Format

```
rslt_t API_TestRAMmemory(void);
```

Return Values

0: OK.
1: NOT OK.

Properties

Prototyped in file iec_tests.c
Implemented in file iec_tests.h

Resources used

None (CPU core).

API execution time

One call to TestRAMmemory() was measured. This consists of multiple calls to the underlying test algorithm. The test covers all of RAM from the SFR area at 0x400 to the stack at 0x45E in the demo, or 86 bytes of RAM.

(1) **Using RamTest_March_C_16Bit**

27.0 ms.

(2) **Using RamTest_March_X_16Bit**

14.5 ms.

(3) **Using RamTest_March_X_WOM_16Bit**

4.0 ms.

See further document 7(2) 下の. More execution times are here listed for various memory sizes and optimizations for the R8C core using an earlier version of the compiler.

Comments

Test alternatives

The test is non-destructive and copies the user data to a buffer. After testing is complete, the original data is restored. The three different RAM tests are variants of bit pattern test read/writes to memory. A customer would only need to select one:

- RAM memory test using March C
- RAM memory test using March X
- RAM March X test algorithm for Word Oriented Memory

Sectioning of memory for non-destructive testing

The memory low address to test cannot be found at run-time, so instead it is inserted when the code is linked, and so will always be up-to-date.

A small 'bss' memory section (static variable section without initial values) *ram_iec_safe* is set aside to be able to do a non-destructive memory test. For a startup test where RAM is not yet used, this restoring of data can be omitted. If so,

pass the argument NULL (0) to the underlying RAM test inside the API. This is actually done for the testing of the *ram_iec_safe* section. Since this section is a test buffer, it does not need to be restored with any original content.

The RAM test recovery data buffer needs to be placed in a dedicated space. This section has already been set up in the demo, but is configurable in size and location. To change, open the sect30.inc file and search for the string *ram_iec_safe*. The code of interest looks like this:

```
;RAM Safe area for RAM march test.
  .section   ram_iec_safe_SE,DATA
  .org 400H
;ram_iec_safe_top:
  .section   ram_iec_safe_NE,DATA,ALIGN

;Moved up 'normal' bss for IEC.
  .section   data_SE,DATA
  .org 408H
```

To change a section location in the sect30.inc file, change the .org *address* to another value.

To change the size of the RAM test buffer, change RAM_TEST_BUFSIZE in file iec_tests.c. A larger buffer will make RAM testing faster.

'Normal' bss-RAM will start after the *ram_iec_safe* test buffer, so if RAM_TEST_BUFSIZE is enlarged, the *data_SE* .org section address will need to move up in address. For example, if buffer is 0x20 words (=0x40 bytes) move data_SE origin to 440H.

API_TestRamUserStack

Test of RAM Stack

Stack memory is tested by any of the three RAM tests.

Format

```
rslt_t API_TestRamUserStack(void);
```

Return Values

0: OK.
1: NOT OK.

Properties

Prototyped in file iec_tests.c
Implemented in file iec_tests.h

Resources used

80h bytes of dedicated swap memory for the stack to switch to and use while the regular stack is tested.

API execution time

One call to the API was measured. This consists of multiple calls to the underlying test algorithm. The test covers the whole user stack. Size 80h of RAM.

(1) **Using RamTest_March_C_16Bit**

50 ms. Roughly half is for first testing the temporary user stack first.

Comments

Test alternatives

The test is non-destructive and copies the user stack data to a new temporary stack *ram_iec_temp_ustack* while the test is run. This area is first tested in a destructive test. After that the normal stack is tested. After testing the stack area the test code restores the original stack and then switches back to it.

The stack pointer is tested in the CPU register tests.

A heap shall not be used according to MISRA coding rules.

The sectioning of memory for non-destructive testing of the user stack is the same procedure as for testing RAM.

Alternative stack testing

Write two known numbers onto the stack start and end addresses. The numbers are stored in flash/ ROM.

Read the stack values and compare with the stored ones. (In a case of stack overflow, the stack pointer will reach the end of the stack & overwrite the end number.) If the values have changed, the stack has been corrupted.

API_TestROMmemory

Test of ROM

ROM memory is tested by three alternative CRC methods over the memory range.

Format

```
rslt_t API_TestROMmemory(void);
```

Return Values

0: OK.
1: NOT OK.

Properties

Prototyped in file iec_tests.c
Implemented in file iec_tests.h

Resources used

No MCU peripheral resources are used.

API execution time

1024 bytes (1kB) of flash tested.

Using Simple CRC 16 with lookup table (CRC16_Small_LT)	12.4 ms
Using CRC16 CCITT using bit shift (CRC16_CCITT)	9.5 ms
CRC16 CCITT with static table (CRC16_CCITT_Small_LT)	11.0 ms

Figure 2. The table shows execution times for all included CRC sub tests.

For more execution time tests, see 16.1(4) below. Here, more execution times are listed for various memory sizes and optimizations for the R8C.

A CRC check can be done on flash content & compared to a stored value to ensure no data/code has been corrupted.

There are four alternative functions in the underlying test code to calculate a CRC checksum of the flash memory . The user of this application note is recommended to select one of the four functions. In the API you just uncomment the functions that you wish to use, and comment the others.

.1. **Simple CRC-16 with lookup table**. As this uses a table, it takes up more memory space, but is fast (about same speed as the CRC-CCITT with lookup table.)

2. **CRC-CCITT using bit shifting**. Best 'overall' choice for time and memory footprint. It takes up no table const memory and is only marginally slower than the next, which uses a lookup table.

3. **CRC-CCITT using a table**. Fastest, but takes up more flash.

4. **CRC-CCITT with small lookup table**. Uses much less const memory and little RAM, but rather slow.

Comments

Setting start address, length, and CRC value

At the top of the iec_tests.c file, some ROM test values must be set. The section that contains the CRC value is in a separate ROM test flash memory section *rom_iec_ref* and cannot be included in the memory test. To change a section location, change the .org *address* to another location in the sect30.inc file. This is done similar to the RAM test above.

In file iec_test.c we will make use of the memory labels to calculate start address and length for our check summing range:

```
/*********************
 *    IEC ROM test   *
 *********************/
/* Starting memory address for ROM that is to be IEC-tested. */
static const uint8_t* const romtest_start_addr = (const uint8_t* const)
&rom_NE_top;
static const uint32_t rom_test_length = 0x01000;

/* Memory address for ROM CRC reference value and length to test.
These addresses cannot be found at run-time, only at link time. */
#pragma SECTION rom rom_iec_ref
static const uint16_t crc_ref = 0x4ccc;
```

ROM test start address

This is taken from the section label where the application code starts. In our example *rom_NE_top*, whose address is automatically resolved by the linker. The label *rom_NE_top* is found in the sect30.inc file.

Setting test length directly

Check your map file for *rom_test_length,* the length of your application code in bytes.

Setting test length automatically

The length from one section to another can also be calculated automatically by adding another label to the sect30.inc, e.g. *reset* in sect30.inc:

```
const uint32_t rom_test_length = ((const uint32_t) ( (uint32_t)&reset -
(uint32_t)&rom_NE_top));
```

The address *reset* is at the end of memory. This is prepared for in the demo. Uncomment the line above in iec_test.c to use it, and comment the line right before it.

Marked in the map file below in Figure 3is the area for *rom_test_length* when the ‘automatic’ length calculation formula is used.

Setting the CRC check reference value

After finishing development of your application, the value of the CRC checksum must be found and entered into the reference value *crc_ref*. In the demo this can be done by pressing Switch 3 on the RSK at the same time as API_TestROMmemory is started by pressing Switch 1. The CRC value should show up on the LCD display. Enter this value as *crc_ref* and reprogram the part.

Note that the download debug kernel cannot be used when doing this if it is part of the checked flash memory region, as the debugger will not be there when the device is programmed standalone.

Address(size)	Section	Label:
000000(000400)		
000400(000008)	[D] ram_iec_safe_NE	
000408(00006a)	[D] data_NE	[G] 000408:_data_SE_top [G] 000408:_tra_underflows [G] 00040a:_tot_tra_underflows [G] 00040c:_tot_trb_underflows [G] 000416:_IEC_API_function [G] 000446:_menu_string [G] 000466:__pool [G] 00046c:__memt
000472(000078)	[D] bss_NE	[G] 000472:_mainfreq_meas_type2_result [G] 0004da:_cpu_test_result [G] 0004e2:__mbase [G] 0004e4:__mnext [G] 0004e6:__msize
0004ea(000004)	[D] data_NO	[G] 0004eb:_mainfreq_meas_flag
0004ee(000100)	[D] stack	[G] 00056e:_stack_top
0005ee(000300)	[D] heap	
0008ee(007712)		
008000(000002)	[R] rom_iec_ref_NE	[G] 008000:_rom_iec_ref_top
008002(00000e)		
008010(000050)	[R] rom_NE	[G] 008010:_rom_NE_top [G] 00805a:_rom_test_length
008060(0000ab)	[R] rom_NO	
00810b(0000f5)		
008200(00000c)	[C] char_inverse	[G] 008200:_char_inverse_top
00820c(0000f4)		
008300(0012d9)	[C] program	[G] 008300:_HardwareSetup [G] 008300:_prog_top [G] 00830e:_ConfigureOperatingFrequency [G] 008356:_ConfigurePortPins [G] 008398:_EnablePeripheralModules [G] 00839a:_ConfigureInterrupts [G] 0083aa:_InitialiseDisplay [G] 008458:$DisplayString [G] 0084b0:$LCD_write [G] 0084dc:$LCD_nibble_write [G] 008558:_DisplayDelay [G] 008586:_main [G] 0085ce:_Initialise [G] 0085dc:_API_ReadAD [G] 008614:_InitTimerRA [G] 008624:_InitTimerRB [G] 008636:_TimerRAhandler [G] 0086d0:_TimerRBhandler [G] 0086f2:_CPU_TestAll [G] 008702:_TestGPRs [G] 0087d0:_TestIntRegs [G] 00881e:_TestStackRegs [G] 008896:_TestFlagReg [G] 0088ca:_TestPCReg [G] 0088f0:$CRC16_CCITT_Small_LT [G] 0089a2:$CRC16_CCITT [G] 008a24:$CRC16_Small_LT [G] 008aca:_IECtestInit [G] 008aec:_RunIECtestMenu [G] 008ba6:_API_TestMainClockStability [G] 008bbc:_API_TestCPU [G] 008bd6:_API_TestRAMmemory [G] 008ca4:_API_TestROMmemory [G] 008ccc:_API_ReadADopenCircuit [G] 008d04:_API_CheckADref [G] 008d40:_API_DisableFlashRW [G] 008d5c:_API_EnablePOR [G] 008d60:_API_MonitorVccAndResetIfLow [G] 008d90:_API_StartWDTwithSlowOCO [G] 008daa:_API_SetPinSRtoReadIO [G] 008db8:_API_DummyTest [G] 008dbc:_coreErrorHandler [G] 008dc2:_RamTest_March_C_16Bit [G] 00908a:_RamTest_March_X_16Bit [G] 009316:_RamTest_March_X_WOM_16Bit [G] 00957e:_InitWatchDog [G] 0095a2:_WDT_ISR
0095d9(000001)		
0095da(0000a4)	[C] interrupt	[G] 0095da:start [G] 00967b:$exit [G] 00967b:_exit [G] 00967d:dummy_int
00967e(00006a)	[R] data_NEI	
0096e8(000004)	[R] data_NOI	
0096ec(0067f0)		
00fedc(0000fc)	[R] vector	
00ffd8(000003)		
00ffdb(000001)	[R] ofs2	
00ffdc(000024)	[R] fvector	[G] 00ffdc:_ofs2_top [G] 00fffc:_reset
010000(0effff)		

Figure 3. A memory map for the demo project. It shows the memory region in red that will be tested when the project length is automatically calculated. The length is the address difference of the assembler labels *reset* and *rom_NE_top*. In this case FFDCh – 8010h = 7FECh or 32748 bytes.

API_TestMainClockStability

Test of CPU Clock

The CPU clock speed is compared with a reference clock.

Format

```
rslt_t API_TestMainClockStability(void);
```

Return Values

0: OK.
1: NOT OK.

Properties

Prototyped in file iec_tests.c
Implemented in file iec_tests.h

Resources used

- Timer RA and Timer RA interrupt.
- Timer RB and Timer RB interrupt if method 2 is used.

API execution time

Method 1

21 ms typical, min 11 ms, max 31 ms.

Method 2

1.6 us.

The second method's implementation has the interrupts running continuously, so a result only needs to be 'polled'. The timers could be freed up completely between measurements. To do this, init timers A and B at the API call, and then modify the Timer RA interrupt to set a 'freq test finished' flag when the measurement is finished. The main application would later evaluate the result when the flag has been set. The execution time for this function would then increase with the time to call InitTimerRA() and InitTimerRB().

Comments

The main clock on the RSKR8C35a test is based on the crystal X1. The sample code uses the RSK's 32.7 kHz on-board clock, or 'sub-clock', to check if the oscillator used as the CPU clock is out of range. The sub-clock is a separate oscillator and thus an independent time source.

Two methods are presented. To use a method, uncomment it's macro definition in *timeradc.h*. That is, to use method two, edit *timeradc.h* to be:

```
//#define MEAS_CPU_CLOCK_METHOD_1        1
#define MEAS_CPU_CLOCK_METHOD_2          1
```

Also set the percentage frequency deviation allowed, e.g.

```
#define PERCENT_DEV_ALLOWED              2
```

Note that the permitted clock deviation must take into consideration the instability of the used reference clock as well, in the sample code case sub-clock fC.

The mean measurement value must also be set. This will depend on the value of the Timer RA (and RB) reset values, and on the frequency of the clocks. This value can be calculated or measured empirically. See file *timeradc.h*. There is one for each method.

```
#define CLOCK_MEAS_MEAN        0x0003670
```

Finally, the number of sample measurements to take, to average the measurement for method 2 is determined by

```
#define NR_FMEAS_SAMPLES       20
```

The higher the number, the more accurate the frequency measurement will be, and the more it will be averaged over time.

The minimum and maximum frequency measurement values will be calculated by the NC30 preprocessor.

Using one timer

In the sample code, Timer RA is clocked with the 32.7 kHz sub-clock and triggers an interrupt after 10 ms (can be changed, see below) and then toggles a flag. The measuring function increments in a loop as long as the flag is high. In the original sample code this is set to be for 10 ms (ten interrupt periods).

When taking a measurement, the MCU increments a counter during the whole measurement period, and so is completely occupied during the measurement time. This counting may not be disturbed. Therefore the interrupt flag register is saved and loaded with a value of one less than the TimerRA interrupt level, so only Timer RA may interrupt the measurement (except watchdog, reset and NMI interrupts). The counter is then compared with a constant reference value for pass/fail. The IPL is afterwards restored (normally this will be to 0).

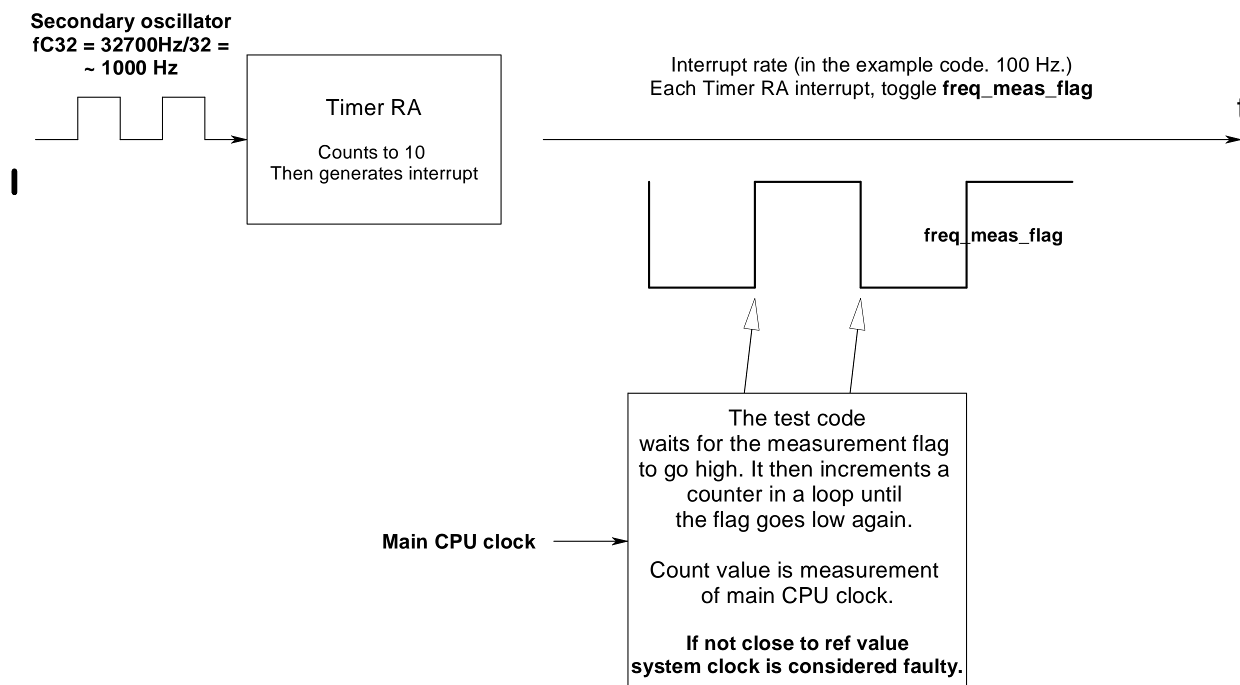

Figure 4. How frequency is checked using only one timer in the demo. The CPU is occupied during the test.

The interrupt frequency and thus the length of time the MCU is occupied can easily be changed.

If Timer RA settings or NR_FREQ_MEAS_TMR_IRQS is changed, CLOCK_MEAS_MEAN and must be adjusted.

Using two timers

This method uses both Timer RA and Timer RB. Timer A is used to measure the number of Timer B cycles (interrupts) that have occurred between Timer A interrupts.

Using two timers alleviates the MCU from being occupied when the measurement is taken. The drawback is two timer resources are used with this method.

The source code builds a sum over several TimerRA periods (NR_FMEAS_SAMPLES) to increase accuracy.

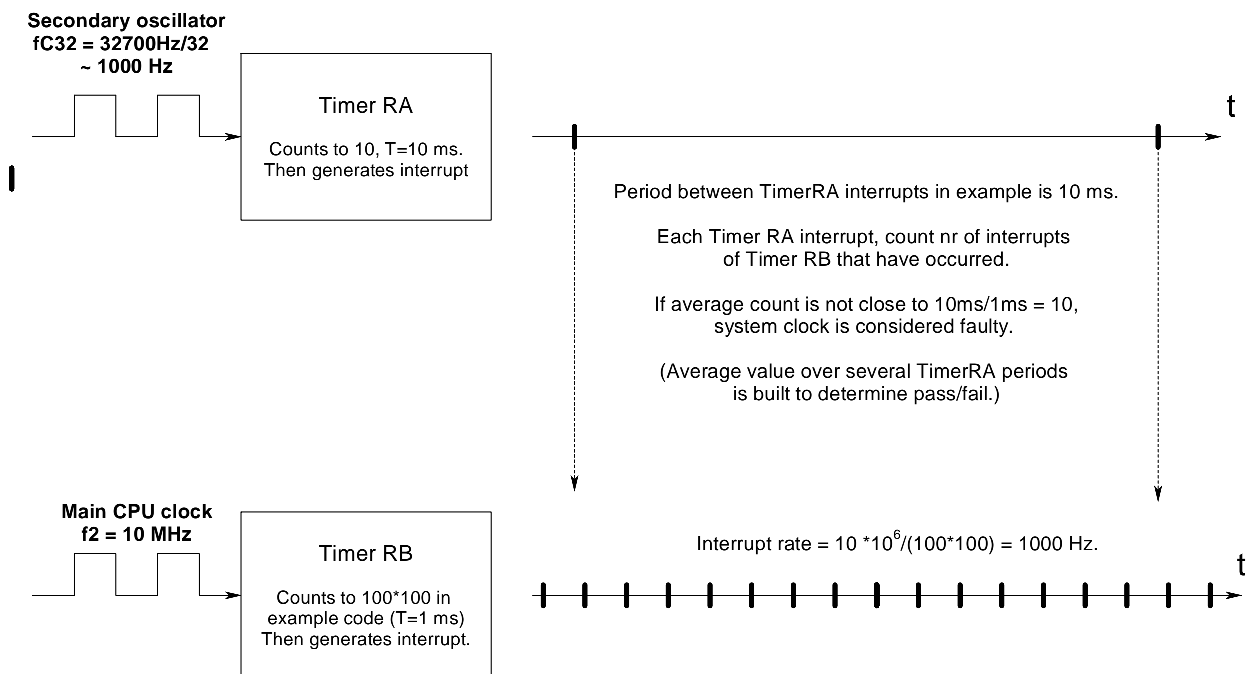

Figure 5. Test of CPU clock frequency using two timers freeing up the MCU during frequency testing. A mean value over several TimerRA periods (NR_FMEAS_SAMPLES) is calculated to increase accuracy.

Other ways to detect frequency deviation

There are more ways to detect frequency deviation, such as using a port pin PWM signal over an RC filter & read voltage on the filter's slope. This will give a voltage slope vs. frequency.

API_ReadADopenCircuit

A-D Sensor Connection Check

Tests whether a sensor is broken or disconnected.

Format

```
rslt_t API_ReadADopenCircuit(void);
```

Return Values

0: OK.
1: NOT OK.

Properties

Prototyped in file iec_tests.c
Implemented in file iec_tests.h

Resources used

- Watchdog Timer circuit.
- On Chip Oscillator

API execution time

12.1 us.

Comments

This function checks if an A-D conversion value is out of bounds.

If the A-D channel input is pulled weakly high to Vcc (or low to ground) in parallel to the sensor, the A-D channel input voltage will be Vcc (or ground) if the sensor is disconnected. If this should occur, it can be detected by reading the AD input channel value and comparing it to a maximum (or minimum) allowable value. For this to work, the allowable range of the sensor may not extend up to Vcc if a pull-up resistor is used, or go all the way down to 0 V if the input is weekly pulled in parallel to 0 V.

To more quickly check if an A-D input is stuck at an extreme value, the A/D Open-Circuit Detection Assist Function is used as described in the HW manual 7(6). This Open-Circuit Detection Assist function reduces influence of a previously converted channel by charging the chopper amp capacitor to Vcc (or ground) before starting a conversion. When a sensor is selected by the A-D input selector, precharging the A-D input enables a faster check whether the value is out of the acceptable A-D sensor range.

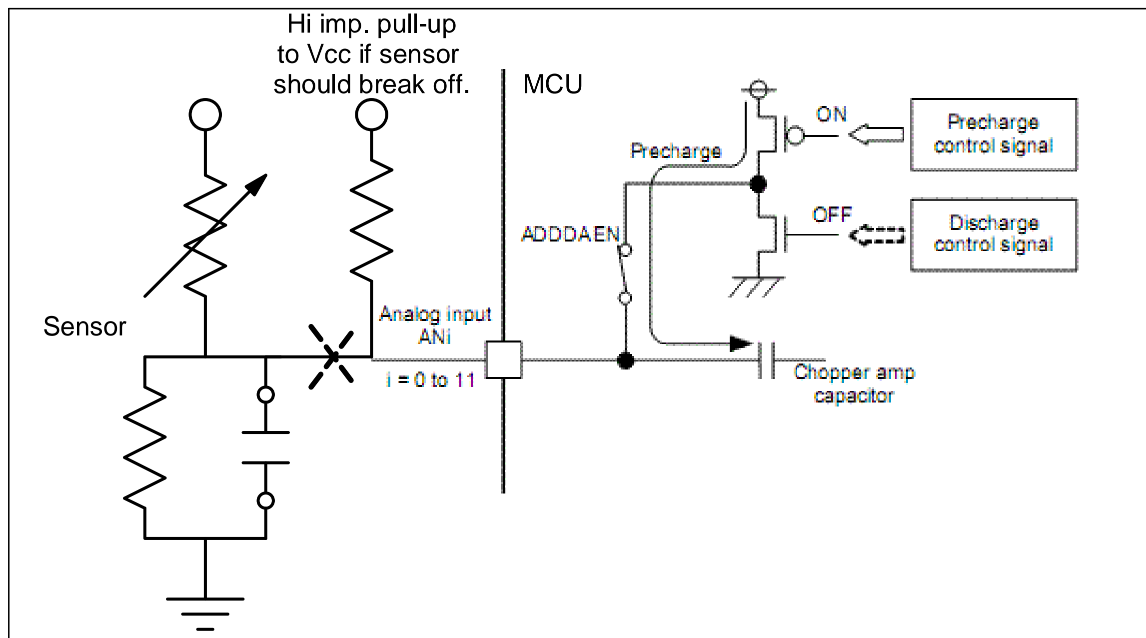

Figure 6. The A/D Open-Circuit Detection Assist Function increases the speed of a 'sensor plausibility check' by pre-charging the A-D input before reading the conversion value. The sensor is then read and checked to be below the Vcc value (e.g. 3FF with 10 bits), signifying that the gage is still connected.

NOTE: If this test, which precharges the A-D input, is run immediately prior to measuring a regular A-D reading, e.g. measuring the On-Chip Voltage Reference, the observed OCVREF value may be affected. Therefore, sufficient sampling time is necessary for the A/D conversion input timing.

API_CheckAD

Test of A-D reading using On-Chip Voltage Reference

A fault in the A-D converter (or a variation in VREF) can be confirmed using the on-chip reference voltage.

This function tests that the A-D is functioning properly by connecting the R8C35 MCU On-Chip Voltage Reference to the A-D comparator and taking a measurement. Its value should be within an expected range defined in constant memory.

Format

```
rslt_t API_CheckADref(void);
```

Return Values

0: OK.
1: NOT OK.

Properties

Prototyped in file iec_tests.c
Implemented in file iec_tests.h

Resources used

- A-D converter. No regular AD channel is occupied, the reference is set apart from AN0-AN11.

API execution time

12.0 us.

Comments

The test connects the On-Chip Voltage Reference to the A-D input and then does a conversion. The value input to the comparator is Vref and OCVREF. The test checks whether the A-D reading is within a range of the On-Chip Voltage Reference value of 1.34V. If the measurement is outside a range, the A-D unit is considered faulty. Of course, a bad value could also mean that Vref or OCVREF have changed, though this cannot be checked by the test, nor is it likely.

The A-D in the R8C/35a demo is clocked at 10 MHz (f1 / 2). This frequency is low enough for a VCC down to 3.0 V.

When measuring the OCVREF, φAD should be 4MHz or below. Use maximum A/D sampling time (see HW manual). This is the time for VIN electric potential to stabilized from 0V or Vcc to the value of OCVREF. In addition, a short delay must be added in the firmware immediately after the point where OCVREF is connected before reading the value to make sure that it has time to charge the AD input completely. That is, right after ADEX0 and OCVREFAN are set to one.

A/D conversion results may contain erroneous data due to possible variances in the internal power supply voltage during conversion. This is caused by the flash memory and A/D converter sharing the same power supply in the MCU. Countermeasure: Take an average of more than one conversion result. This is according to Technical Note TN-SH7-A719A/E.

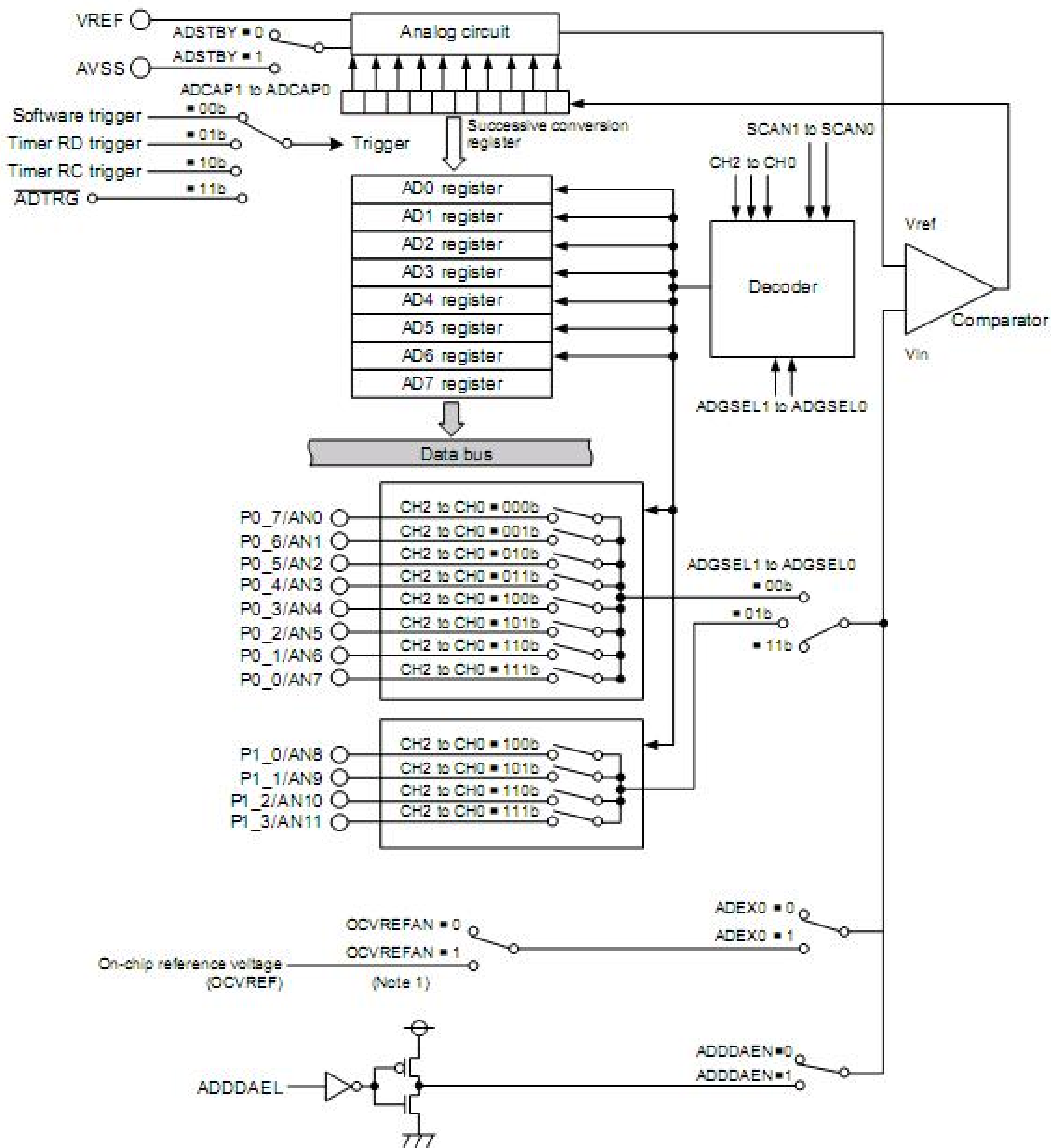

Figure 7. The internal reference voltage Vref is measured. The read value is then checked against a memory reference range.

The actual OCVREF value was measured to be 2.25 V with a of Vcc=5.11 V for the demo.

The ADEX0 bit in the ADCON1 register and the OCVREFAN bit in the OCVREFCR register select the on-chip reference voltage.

API_DisableFlashRW

Protecting Flash Memory

This function protects program and data flash memory from unintentional erases or writes.

Format

```
rslt_t API_DisableFlashRW(void);
```

Return Values

0: OK.
1: NOT OK.

Properties

Prototyped in file iec_tests.c
Implemented in file iec_tests.h

Resources used

-

API execution time

4.8 us.

Comments

Out of reset, the erase and write operations on user program flash memory can be disabled by the FMR0 register. There are also individual lock bits with which to protect the data flash blocks. These are set by the OFS register.

API_ReadPORresetVal

Using the Built-in Power On Reset

Starts up the MCU when Vcc rises up above Vdet0.

Format

```
rslt_t API_ReadPORresetVal(void);
```

Return Values

5: 3.80 V selected (Vdet0_3).
4: 2.85 V selected (Vdet0_2).
3: 2.35 V selected (Vdet0_1).
2: 1.90 V selected (Vdet0_0).
1: NOT OK.

Properties

Prototyped in file iec_tests.c
Implemented in file iec_tests.h

Resources used

- MCU reset circuit.

API execution time

9 CPU cycles. The function is simply an explanation on how to set up the MCU.

Comments

Power On Reset is an MCU feature that starts up the MCU when Vcc rises up above Vdet0. Internal resets are made possible by using an external capacitor & diode, but without the need of an external IC.

If the /Reset pin is pulled high to VCC, of a value Vdet0 or above, the low-speed on-chip oscillator clock starts counting. When it counts to 8, the MCU enters the reset sequence. For R8C devices other than the R8C/35a, the voltage level needs to rise by at least trth mV/ms (Figure 8).

Do not use the .X30 file with HEW to program the device standalone as the OFSREG data is not included.

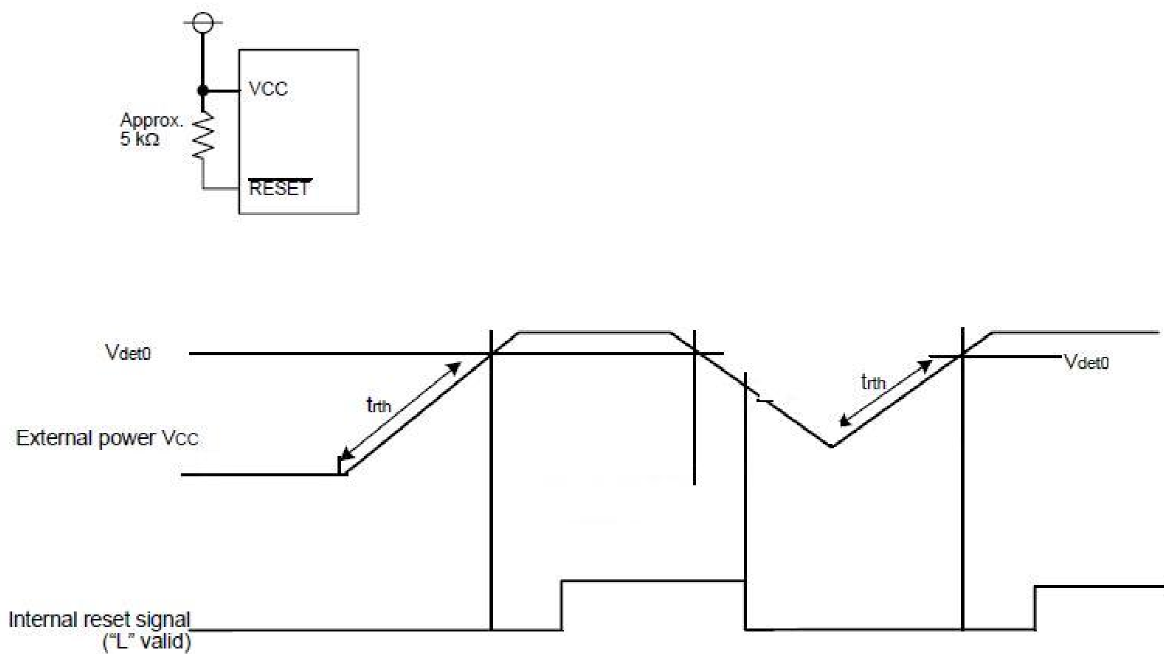

Figure 8 shows the reset signal going low (active) for a rise above Vdet0

To use this functionality, tie the /Reset pin to Vcc. A reset will then occur automatically when Vcc rises above Vdet0. All you need to do code-wise is set Vdet0 which may be chosen between: 3.80 V, 2.85 V, 2.35 V, and 1.90 V. The voltage level is set with the OFS-register. See the source code for comments on how to set this.

See application note in 7(5) for more details on Power On Reset.

API_MonitorVccAndResetIfLow

Monitor DC Power Supply Fluctuation

Power on resets will occur when Vcc falls below Vdet0.

Format

```
rslt_t API_MonitorVccAndResetIfLow(void);
```

Return Values

5: 3.80 V selected (Vdet0_3).
4: 2.85 V selected (Vdet0_2).
3: 2.35 V selected (Vdet0_1).
2: 1.90 V selected (Vdet0_0).
1: NOT OK.

Properties

Prototyped in file iec_tests.c
Implemented in file iec_tests.h

Resources used

- Voltage Detection Circuit.

API execution time

7.0 us.

Protection

This function has associated registers that are protected from unintentional writes by the MCU's protection function.

Comments

The demo includes a function to have the MCU reset when the voltage drops below Vdet0. This is called Voltage Monitor 0 Reset. Power on resets will occur when Vcc falls below Vdet0 if the LVDAS-bit "Voltage detection 0 circuit start bit" in the OFS-register is set to 0.

Figure 8 shows how the MCU will reset when external power drops below Vdet0.

See Application note in 7(5), *rej05b1023_r8cap.pdf* for more details on Voltage Monitor 0 Reset.

Do not use the .X30 file with HEW to program the device standalone as the OFSREG data is not included.

API_SetPinSRtoReadIO

Detect actual voltage level of an IO pin.

Set up the MCU to detect actual high or low voltage level of an IO pin.

Format

```
rslt_t API_SetPinSRtoReadIO(void);
```

Parameters

None.

Return Values

0: OK.
1: NOT OK.

Properties

Prototyped in file iec_tests.c
Implemented in file iec_tests.h

Resources used

None

API execution time

3.4 us.

Comments

This function sets up the MCU (currently only R8C/35a) to use the Output Level Detection function which detects the actual voltage high or low of an IO pin. The port status can be read regardless of direction mode.

To set up the MCU to read the actual IO pin voltage (1 or 0), set the PINSR (I/O Function Pin Select) register's IOINSEL bit to 1.

API_StartWDTwithSlowOCO

Using the Watchdog

The watchdog is typically used as 'Time Slot Monitor' and resets the MCU when its timer is not serviced regularly.

Format

```
rslt_t API_StartWDTwithSlowOCO(void);
```

Return Values

0: OK.
1: NOT OK.

Properties

Prototyped in file iec_tests.c
Implemented in file iec_tests.h

Resources used

- Watchdog Timer circuit.
- On Chip Oscillator

API execution time

9.2 us.

Protection

This function has associated registers that are protected from unintentional writes by the MCU's protection function. See chapter Register Protection.

Comments

The watchdog is typically used as the 'Time Slot Monitor' in the IEC standard.

Either of following can be selected to run the watchdog:

- Count starts automatically after a reset. This is determined by the OFSREG register.
- Count starts by writing to a watchdog register.

Do not use the .X30 file with HEW to program the device standalone as the OFSREG data is not included

Count Source Protection Mode

The watchdog is used in Count Source Protection Mode Enabled. This means the watchdog will be clocked with the 125 kHz On-Chip Oscillator.

If the watchdog is initialized by writing to the wdts or wdtr register, the watchdog timer will reset the MCU if it's counter underflows. To avoid watchdog resets, the software must write to the watchdog timer to reset it regularly.

Starting the watchdog

The watchdog can be started in two ways.

1. Automatically at reset by setting the registers OFSREG and OFSREG2. Since the watchdog will start counting from reset, it must be refreshed earlier on as opposed to starting it manually as in 2.
2. From application source code. Start the watchdog by writing to the Watchdog Start or Reset register. By using this method the watchdog interrupt can be used instead of having the MCU reset. To use the

watchdog start and refresh code already provided, define WDT to e.g. ‘1’ from within HEW. This is easiest as it will be defined globally for all files and will enable watchdog refresh code.

In the example code a 100% write window is used, meaning the watchdog can be refreshed anytime in its count cycle. To have a 100% write cycle OFS2REG set the must be set to 0xFF. (It is 0 by default!) See the source code for how to set this register.

(API_TestWDT)

User test of Watchdog

This is not a part of resident test code. It is just a manual demo test function for developers to check whether the Watchdog is running

Format

```
rslt_t API_TestWDT(void);
```

Return Values

0: OK. Watchdog was not on.
CPU RESET: Watchdog was running.

Properties

Prototyped in file iec_tests.c
Implemented in file iec_tests.h

Resources used

- Watchdog Timer circuit.
- On Chip Oscillator

API execution time

NA

Protection

NA

Comments

If the CPU resets. The function goes into a delay-loop, occupying the CPU. The watchdog is not serviced and so it will time out and reset the CPU if it is running.

6. More functions

Other functions which are not explicitly part of Class B software testing requirements that can be added to testing are

7. References and Bibliography

Other documents and presentations on testing according to IEC 60730.

(1) **Renesas Guide to IEC 60730-1 with regards to integrated controls**

Written with the designers of products (e.g. white goods) in mind. A general investigation, guide, and condensation of the relevant parts of the standard for MCU-controlled class B products. Written by RTA.

(2) **Self Test Sample Code for Renesas Microcontroller Families. Application Note REG05B0016-0400.**

(3) **Walking, marching and galloping patterns for memory tests**

RAM testing algorithms. Defines the fault types; Stuck-a-faults, Coupling faults, Transition faults etc. A term paper for ELEC 7250 submitted by Arvind Raghuraman.

(4) **March Test for Word oriented Memories**

A research paper on software marching tests. Another term paper for ELEC 7250 submitted by Arvind Raghuraman.

(5) **Power-On Reset Function and Voltage Monitor 0 Reset**

Renesas document REJ05B1023. This document contains voltage level diagrams and behavior for Vdet0.

(6) **The R8C/35A Group Hardware Manual**

The R8C/35A HW manual REJ09B0407-0010 was used when writing this document.

(7) **The IEC60730-1 standard**

The governing standard for the E.U.

Appendix H lists "requirements for electronic controls" and specifies measures for insuring compliance. This is the main section of interest for software tests. Of special interest for Renesas and white goods applications is software **Class B** which 'control functions intended to **prevent unsafe operation** of the controlled equipment'.

Condensed version of table H.11.12.7 for Class B controllers

Component	Fault/error	Acceptable measures	Definitions	MCU Impacted
1. CPU				
1.1 Registers	Stuck at	Functional test, or	H.2.16.5	YES
		periodic self-test using either:	H.2.16.6	YES
		– static memory test, or	H.2.19.6	YES
		– word protection with single bit redundancy	H.2.19.8.2	YES
1.3 Programme counter	Stuck at	Functional test, or	H.2.16.5	YES
		periodic self-test, or	H.2.16.6	YES
		independant time slot monitoring, or	H.2.16.5	YES
		logical monitoring of the programme sequence	H2.18.10.2	YES
2. Interrupt handling & execution	No interrupt or too frequent interrupt	Functional test, or	H.2.16.5	YES
		time-slot monitoring	H.2.16.5	YES
3. Clock	Wrong frequency	Functional test, or	H.2.18.10.1	YES
		time-slot monitoring	H.2.18.10.4	YES
4. Memory				
4.1 Invariable Memory	All single bit faults	Periodic modified checksum; or mutliple checksum or word protection with single bit redundancy	H.2.19.3.1	YES
		mutliple checksum, or or word protection with single bit redundancy	H.2.19.3.2	YES
		word protection with single bit redundancy	H.2.19.8.2	YES
4.2 Variable Memory	DC fault	Periodic static memory test, or word protection with single bit redundancy	H.2.19.6	YES
		word protection with single bit redundancy	H.2.19.8.2	YES
4.3 Addressing (relevant to variable & invariable memory)	Stuck at	word protection with single bit redundancy including the address	H.2.19.18.2	YES
5. Internal data path	Stuck at	word protection with single bit redundancy	H.2.19.18.2	YES
5.2 Addressing	Wrong address	word protection with single bit redundancy including the address	H.2.19.18.2	YES
6. External communication	Hamming distance 3	word protection with multi-bit redundancy, or	H.2.19.18.1	NO
		CRC - single word, or	H.2.19.4.1	NO
		transfer redundancy, or	H.2.18.2.2	NO
		protocol test	H.2.18.14	NO
6.3 Timing	Wrong point in time	Time slot monitoring, or	H.2.18.10.4	NO
		scheduled transmission	H.2.18.18	NO
		Time slot and logical monitoring, or comparison of redundant communication channels by euther:	H2.2.18.10.3	NO
		- Reciproque comparison	H.2.18.15	NO
		- Independant hardware comparator	H.2.18.3	NO
	Wrong sequence	logical monitoring, or	H.2.18.10.2	NO
		Time slot monitoring, or	H.2.18.10.4	NO
		scheduled transmission	H.2.18.18	NO
7. Input/output periphery	Faults conditions specified in H.27	Plausibility check	H.2.18.13	YES
7.2 Analog I/O				
7.2.1 A/D & D/A convertor	Faults conditions specified in H.27	Plausibility check	H.2.18.13	YES
7.2.2 Analog MUX	Wrong addressing	Plausibility check	H.2.18.13	YES

Comments.

2. To test that periodical interrupts occur as designed, a counter in each ISR can be incremented when an interrupt occurs. For example, if the Serial Peripheral Interface (SPI) is configured to generate an interrupt every 10 ms, the SPI should generate at least 10 interrupts in 100 ms. This counter can then be verified by the another timer interrupt, for example the same interrupt as used in the frequency test.

4.3 and 5 do not contain the wording “memory test”. Item 4.3 concerns the RAM & Flash/ROM. It is possible to detect the "stuck at" fault by the self-test software if the routines are called periodically and after reset. Item 5 concerns the internal data path. Any errors are detected by the use of internal independent Watch Dog Timer. If there are any errors or faults in the addressing of the internal bus the program will not run correctly. The WDT is the right HW module to detect such errors.

6. External comm. Protocol test: This could be a function reading a static memory area being written to by an interrupt. The function would clear the data before exiting.

The Interrupt test implements the independent time slot monitoring H.2.18.10.4 defined by the IEC 60730 standard. It checks whether the number of interrupts that occurred is within the predefined range. The goal of the Interrupt test is to verify that interrupts occur regularly. The Interrupt test function can be invoked at specified time intervals. It is triggered by a timer or line frequency interrupt to monitor and verify the interrupt operation.

Website and Support

Renesas Electronics Website
http://www.renesas.com

Inquiries
http://www.renesas.com/inquiry

All trademarks and registered trademarks are the property of their respective owners.

Revision Record

Rev.	Date	Comments	
		Page	Summary
0.90	Aug 29 ‘08	—	First edition for review.
0.94	Sep 14 ’08		After review FP.
0.95	Sep 16 ‘08		Before group review RTA.
0.98	Oct 2 ‘08		Changes according to review Sep 19.
1.00	Oct 10 ‘08		Official release.
1.03	Mar 20 ‘09		Changes as per RSO feedback for RTE’s core tests.
1.04	Apr 2 ‘09		Changes for GSCE (A4, doc number added)
1.05	Jun 5 ‘09		Omitted all Hi-speed OCO references. Refer back to 1.04 workspace and application note when OCO is re-released.
1.06	Apr 12 ‘10		API_TestRamUserStack and API_TestWDT sections added. Chapter 3 tables also changed to include this API.

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan, R.O.C.
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

© 2010 Renesas Electronics Corporation. All rights reserved.

Colophon 1.0