RTP Self-Test Library

Example Code Introduction for 64/32-bit NuMicro® Family

Document Information

Application	This example code is a self-test library for Real Time Processor (RTP) of MA35D1 series microprocessor. This library implements the self-test function of the chip to meet safety requirements required by the market. When errors occur in the chip, they can be detected in real-time and functional safety of the system can be maintained.
BSP Version	MA35D1 Series RTP BSP CMSIS V3.00.001
Hardware	NuMaker-HMI-MA35D1-S1

The information described in this document is the exclusive intellectual property of Hugoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. Overview

This example code is a Self-Test Library (STL) for the Real Time Processor (RTP) of MA35D1 series microprocessor. This library provides basic system detection functions, such as self-test of CPU register, program counter, clock, Flash, and SRAM to detect errors in real-time and maintain functional safety of the system when errors occur in the chip. Additionally, users can modify or add this example code to existing applications to meet system safety requirements, and accelerate user development and safety certification processes.

This example code implements methods and techniques used by the STL, providing user with secure application functionality. However, final verification at the application system level must be verified by a certification agency.

1.1 Principle

The software process is shown in Figure 1-1. The code is divided into two parts for self-test: Startup Time Check and Run Time Check. Both parts will call the self-test function in the STL to perform corresponding detection items.

The Startup Time Check is only executed once when the system is powered on to ensure the integrity of the code. Then, in the Main Loop, the Run Time Check will be continuously executed for self-test to ensure that the chip runs normally during applications running and maintain system safety.

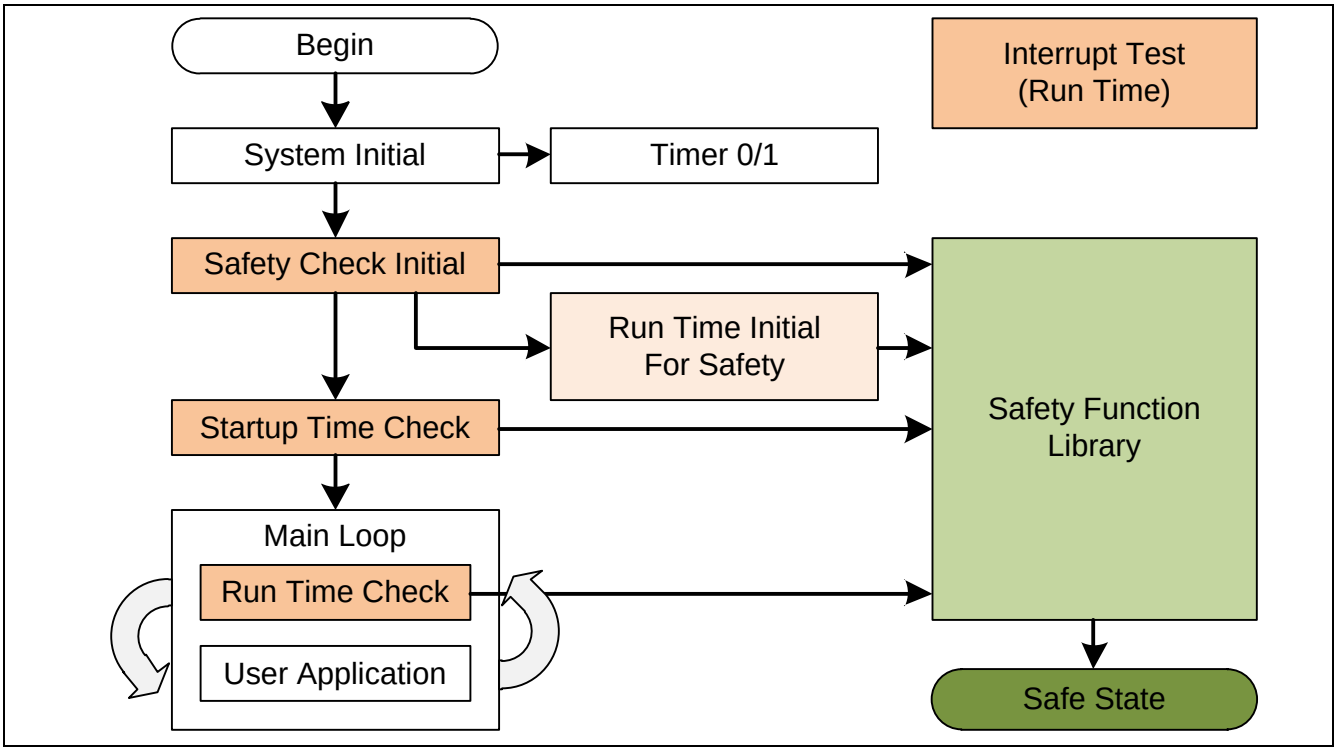

Figure 1-1 Software Flowchart

1.2 Demo Result

When the program is executed normally, it will display "Pass" for each detection item:

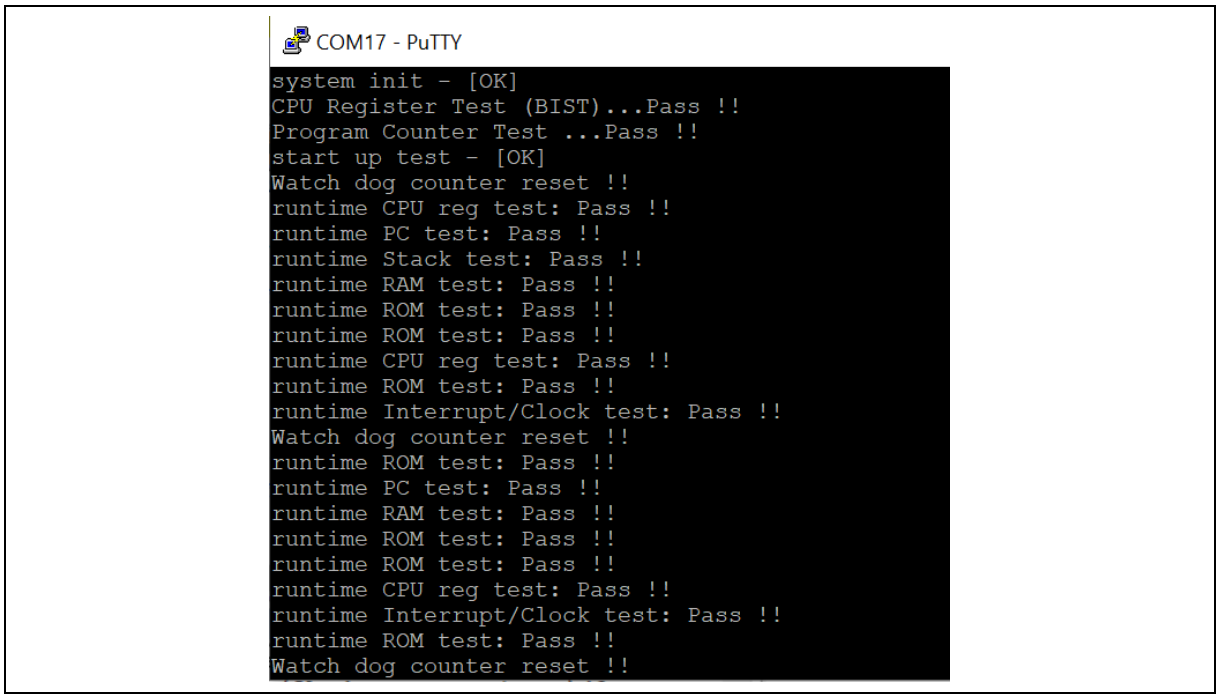

Figure 1-2 Demo Result

When there is a problem with a detection item, the corresponding "fail code" will be displayed. User can identify the item with the error based on the fail code:

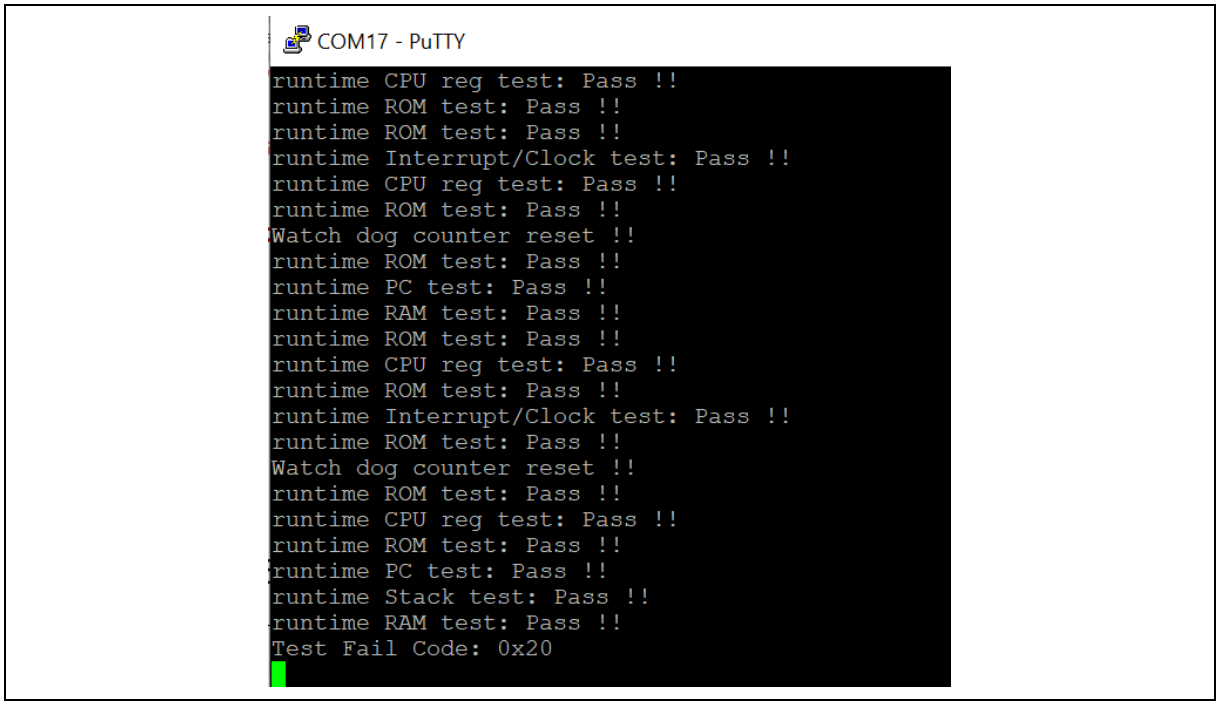

Figure 1-3 Test Fail Case

The table of fail code is shown below:

Test Functions Fails	Fail Code
CPU Register Test Fail	0x10
Program Counter Test Fail	0x14
Stack Test Fail	0x26
Interrupt Test Fail	0x18
Clock Test Fail	0x1C
Flash Test Fail	0x20
RAM Test Fail	0x25

Table 1-1 Fail Code

2. Code Description

The following code implements the architecture described in section 1.1.

```
int main(void)
{
    if(system_init())
        printf("system init - [FAIL]\n");
    else
        printf("system init - [OK]\n");

    /* To check if system has been reset by WDT time-out reset or not */
    if(WDT_GET_RESET_FLAG())
    {
        /* Clear WDT reset flag */
        WDT_CLEAR_RESET_FLAG();
        printf("*** System has been reset by WDT time-out event ***\n\n");
        while(1)
            WDT_RESET_COUNTER();
    }

    /* Test Lib Init */
    TESTLIB_INIT(UserErrorHandling, WatchDogReset);

    if(STARTUP_TESTS() != 1)
        printf("start up test - [FAIL]\n");
    else
        printf("start up test - [OK]\n");

    while (1){
        /* Run Time Execution Check */
        CHECK_RUNTIME_TESTS_EXECUTION();
        /* Run Time Test mainloop */
        RUNTIME_TESTS();
    }
}
```

The detection items that will be called in the Startup Test are as follows:

```
uint8_t STARTUP_TESTS(void)
{
    uint8_t u8TestResult;
    /* CPU Registers Test */
    rt_printf("CPU Register Test (BIST)...");
    u8TestResult = Registers_Test(STARTUP);
    if(u8TestResult)
        rt_printf("Pass !!\n");
    else
        SAFE_STATE(CPU_TEST_FAIL);

    /* Program Counter Test */
    rt_printf("Program Counter Test ...");
    u8TestResult = ProgramCounter_Test();
```

```
    if(u8TestResult)
        rt_printf("Pass !!\n");
    else
        SAFE_STATE(PC_TEST_FAIL);

    return u8TestResult;
}
```

The detection items that will be called in Runtime Test are as follows:

```
void RUNTIME_TESTS(void)
{
    uint8_t u8TestResult;
    /* reset wdt */
    if(RunTimeTestExecution.WATCHDOG_RESET == 1) {
        u8TestResult = WatchDog_Reset();
        if(u8TestResult)
            rt_printf("Watch dog counter reset !!\n");
    }

    /* CPU Registers Test */
    if(RunTimeTestExecution.REGISTER_TEST == 1)
    {
        u8TestResult = Registers_Test(RUNTIME);
        if(u8TestResult)
            rt_printf("runtime CPU reg test: Pass !!\n");
        else
            SAFE_STATE(CPU_TEST_FAIL);
    }

    /* Program Counter Test */
    if(RunTimeTestExecution.PROGRAMCOUNTER_TEST == 1) {
        u8TestResult = ProgramCounter_Test();
        if(u8TestResult)
            rt_printf("runtime PC test: Pass !!\n");
        else
            SAFE_STATE(PC_TEST_FAIL);
    }

    /* Stack Overrun Test */
    if(RunTimeTestExecution.STACK_TEST == 1) {
        u8TestResult = Stack_Test();
        if(u8TestResult)
            rt_printf("runtime Stack test: Pass !!\n");
        else
            SAFE_STATE(STACK_TEST_FAIL);
    }

    /* RAM Test */
    if(RunTimeTestExecution.RAM_TEST == 1) {
        u8TestResult = RAM_Test(RUNTIME);
        if(u8TestResult)
            rt_printf("runtime RAM test: Pass !!\n");
        else
            SAFE_STATE(RAM_TEST_FAIL);
```

```
    }

    /* ROM Test */
    if(RunTimeTestExecution.ROM_TEST == 1) {
        u8TestResult = Flash_Test(RUNTIME);
        if(u8TestResult)
            rt_printf("runtime ROM test: Pass !!\n");
        else
            SAFE_STATE(FLASH_TEST_FAIL);
    }

    /* Interrupt/Clock test */
    if(RunTimeTestExecution.INTERRUPT_TEST) {
        u8TestResult = Interrupt_Clock_Test();
        if(u8TestResult)
            rt_printf("runtime Interrupt/Clock test: Pass !!\n");
        else
            SAFE_STATE(INTERRUPT_TEST_FAIL);
    }
}
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - MA35D1 Series RTP BSP CMSIS V3.00.001
- IDE version
  - Keil uVersion 5.37

3.2 Hardware Requirements

- NuMaker-HMI-MA35D1-S1
- Pin Connection

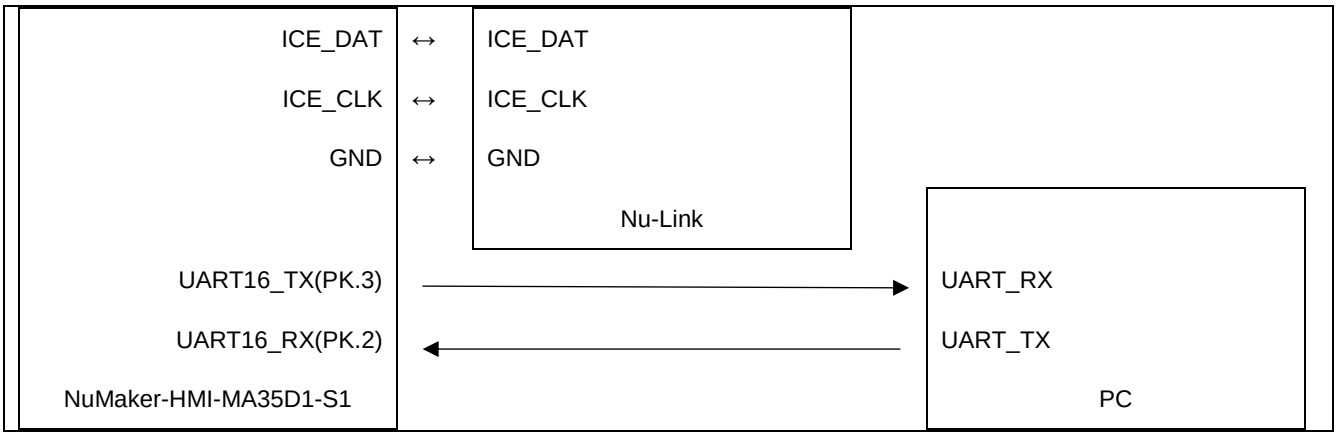

Figure 3-1 Pin Connection

4. Directory Information

The directory structure is shown below.

EC_MA35D1_RTP_Self_Test_Library_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	Source file of example code

Figure 4-1 Directory Structure

5. Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *RTP_Self_Test_Library.uvproj*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

For details, please refer to *RTP User Manual*. The download link is as follows:

https://www.nuvoton.com/resource-download.jsp?tp_GUID=UG132022120809384399

6. Revision History

Date	Revision	Description
2023.2.28	1.00	Initial version.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.