



# Intel® 82802 Firmware Hub: Random Number Generator

Programmer's Reference Manual

---

*December 1999*

Order Number: 298029-001





Information in this document is provided in connection with Intel products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in Intel's Terms and Conditions of Sale for such products, Intel assumes no liability whatsoever, and Intel disclaims any express or implied warranty, relating to sale and/or use of Intel products, including liability or warranties relating to fitness for a particular purpose, merchantability or infringement of any patent, copyright or other intellectual property right. Intel products are not intended for use in medical, lifesaving, or life-sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

The Intel® 82802 Firmware Hub's Random Number Generator Interface may contain design defects or errors known as errata that may cause the product to deviate from published specifications. Current characterized errata are available upon request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

I<sup>2</sup>C is a 2-wire communications bus/protocol developed by Philips\*. SMBus, a subset of the I<sup>2</sup>C bus/protocol, was developed by Intel. Implementations of the I<sup>2</sup>C bus/protocol may require licenses from various entities, including Philips Electronics N.V. and North American Philips Corporation.

Alert on LAN is a result of the Intel-IBM Advanced Manageability Alliance and is a trademark of IBM\*.

Copies of documents that have an order number and are referenced in this document or other Intel literature may be obtained from:

Intel Corporation

URL: [www.intel.com](http://www.intel.com)

Phone: 1-800-548-4725

\*Third-party brands and names are the property of their respective owners.

Copyright © Intel Corporation 1999



# Contents

1.	Introduction .....	5
1.1.	Purpose and Scope .....	5
1.2.	Intended Audience.....	5
1.3.	Document Organization.....	5
2.	Random Number Generator Interface .....	7
2.1.	Hardware Status.....	7
2.2.	RNG Status .....	7
2.3.	RNG Data .....	8
3.	Random Number Generator Usage.....	9
3.1.	Detecting and Initializing the RNG Device.....	9
3.1.1.	Detecting the RNG Device.....	9
3.1.2.	Initializing the RNG Device .....	9
3.1.3.	Pseudo-Code Example of Detection and Initialization .....	10
3.2.	Retrieving a Random Number.....	11
3.2.1.	Pseudo-Code Example of Retrieving a Random Number .....	12



## Revision History

Rev.	Draft/Changes	Date
-001	Initial Release	December 1999

# 1. Introduction

---

This document describes the interface to the Intel® 82802 Firmware Hub Device Random Number Generator (RNG). The RNG is dedicated hardware that harnesses system thermal noise to generate random and indeterministic values.

## 1.1. Purpose and Scope

This document describes how to use the RNG hardware interface. The scope of the document is limited to the minimum information necessary to enable a user to detect and initialize the RNG and to retrieve random numbers. The functions and locations of relevant I/O registers are described, and a usage model is provided for each of the relevant operations.

This document is not intended to provide guidance on how the RNG functionality should be exposed to an operating system or other software application.

## 1.2. Intended Audience

This document is intended for use by driver or operating system developers who wish to access the RNG device. It is assumed that the reader is highly technical and is familiar with the general methods of accessing and controlling hardware, such as the Firmware Hub, in their particular system environment.

## 1.3. Document Organization

This document is divided into the following sections:

- **Section 2, *Random Number Generator Interface***, provides a description of the RNG I/O registers and their relevant values. This section provides a reference for those already familiar with the RNG usage model.
- **Section 3, *Random Number Generator Usage***, provides a description of how to use the I/O registers to detect and initialize the RNG device and to retrieve a random number.



Page Intentionally Left Blank

## 2. Random Number Generator Interface

---

This section describes the location and relevant values of the RNG registers. (Section 3 describes how to use these registers in order to detect and initialize the RNG device and to retrieve a random number.)

### 2.1. Hardware Status

The Hardware Status register is used to determine whether or not an RNG device is present on the Firmware Hub and, if so, whether it is enabled for use.

**Physical Address:** 0xFFBC015F (*read/write*)

**Values:**

Bit Mask	Description
0x40	<b>(Bit 6 – read only) RNG Present:</b> If this bit is read as 1, an RNG is present on the Firmware Hub device. If it is read as 0, the RNG is absent.
0x01	<b>(Bit 0 – read/write) RNG Enabled:</b> This bit indicates whether or not the RNG is enabled. If the bit is set to 1, the RNG will generate random data. If this bit is 0, the RNG is disabled.

### 2.2. RNG Status

The RNG Status register is used to determine whether or not the RNG Data register (described next) contains valid random data. The RNG Data register should never be read until the RNG Status register indicates that the RNG Data register is valid. Each time the RNG Data register is read, the RNG Status bit is cleared, and it remains cleared until the RNG Data register is filled with new random data.

**Physical Address:** 0xFFBC0160 (*read only*)

**Values:**

Bit Mask	Description
0x01	<b>(Bit 0 – read only) Data Available:</b> If this bit is read as 1, the RNG Data register contains valid random data. If this bit is read as 0, the data in the RNG data register is not valid.



## 2.3. RNG Data

The RNG Data register contains a byte of random data. The data contained in this register should be used only if (1) the RNG Present and RNG Enabled bits of the Hardware Status register are set and (2) the Data Available bit of the RNG Status register is set.

**Physical Address:** 0xFFBC0161 (read only)

**Value:** One byte of random data or zero if the RNG is not present or not enabled.



## 3. Random Number Generator Usage

---

This section describes how the RNG interface registers should be used in order to detect and initialize the RNG device and to retrieve a random number.

### 3.1. Detecting and Initializing the RNG Device

Before any process attempts to read random data directly from the Firmware Hub RNG device, it should execute a process to verify that a supported RNG device is available for use, enable the device, and verify the correct functionality. This initialization process is described in a following subsection.

#### 3.1.1. Detecting the RNG Device

The Manufacturer Code and Hardware Status registers are used to determine whether a supported RNG device is available on the system.

**Step 1:** From the system BIOS or using the Read Identifier Codes command, as specified in the Intel® 82802AB/82802AC Firmware Hub (FWH) datasheet, verify the Intel 82802 manufacturer code.

**Step 2:** If a valid Intel 82802 FWH is found, then the RNG Present bit (bit 6) of the Hardware Status register should be checked in order to verify that an RNG device is available.

**Note:** There is a chance that, even if no RNG device is present, the physical memory locations described above may coincidentally match the values expected for an RNG device. For this reason, before random data is sent to an application, the device should be exercised to verify that it is indeed an RNG. This can be accomplished by enabling the device and running an initial test (e.g., FIPS (Federal Information Processing Standard) 140-1) before use.

#### 3.1.2. Initializing the RNG Device

Once the RNG device is detected, it must be enabled and should be tested before use.

**Step 1:** The RNG Enabled bit (bit 0) of the Hardware Status register must be set to enable the RNG device.

**Step 2:** Once the RNG is enabled, an initial read of the RNG Data register should be made to clear any preexisting data from the register.

**Step 3:** A test (e.g., FIPS 140-1) should be run on the RNG Device. This test will ensure that there was no error in detecting the device and that the device is functioning properly.

### 3.1.3. Pseudo-Code Example of Detection and Initialization

The following pseudo-code demonstrates how to detect and initialize the RNG device:

```
//Pseudo-code example of detection and initialization of the
//Firmware Hub RNG Device.

//Save register locations - locations are defined here relative
//to a base feature space address
uint8* pFeatureSpace = 0xFFBC0000;
uint8* pHardwareStatus = pFeatureSpace + 0x015F;
uint8* pRNGStatus = pFeatureSpace + 0x0160;
uint8* pRNGData = pFeatureSpace + 0x0161;

//Check the Manufacturer ID for a 82802 Firmware Hub code.

//Check the Hardware Status for presence of RNG Device
if ( (*pHardwareStatus & 0x40) == 0)
then (FAIL - NO RNG PRESENT);

//Enable the RNG using the Hardware Status register
*pHardwareStatus = (*pHardwareStatus | 0x01);

//Burn initial random number by reading the RNG Data register.
//See the following section, "Retrieving a Random Number", for
//a description of how to read data from the RNG Data register
if ( GetRandomData() == FAIL)
then (FAIL - COULDN'T INITIALIZE RANDOM DATA BUFFER)

//Run a test on the RNG
if ( RunFipsTest() == FAIL)
then (FAIL - START-UP TEST FAIL)

//It is now possible to generate random numbers.
```

## 3.2. Retrieving a Random Number

Once the RNG device has been detected and initialized, it is ready to provide random data to applications. This section describes the recommended process for reading random data from the RNG.

Follow these steps to retrieve a byte of random data from the RNG:

**Step 1:** Look at the zero bit of the RNG Status register. If the bit is set, then the RNG Data register contains valid data, so proceed to Step 2. If the bit is not set, then the RNG Device is still busy generating random data, so wait a short period of time and repeat this step.

**Note:** Because the RNG is based on a natural noise source, varying lengths of time may be required for a byte of random data to become available. A timeout of 4.5 milliseconds is recommended for polling the RNG Status register.

**Step 2:** Read a byte of random data from the RNG Data register. The data in the RNG Data register is valid for one read only. Do not read the register a second time without first checking the status, as described in Step 1.

**Note for multithreaded applications:** It is important that multithreaded applications protect the RNG Data register from access by multiple threads. Steps 1 and 2 above should be executed as a critical section of code, in order to prevent one thread from reading the RNG Status register while another is reading the RNG Data register.

### 3.2.1. Pseudo-Code Example of Retrieving a Random Number

The following pseudo-code demonstrates how to retrieve random data from the RNG device:

```
//Pseudo-code example of retrieving a random number from the RNG
//device. This example assumes that the RNG device has already
//been detected and enabled. For clarity, no timeout test is
//included.

//Retrieve 32 bits of random data, and save the data in this
//this variable
uint32 dwRandomData = 0;

//Save register locations - locations are defined here relative
//to a base feature space address
uint8* pFeatureSpace = 0xFFBC0000;
uint8* pRNGStatus = pFeatureSpace + 0x0160;
uint8* pRNGData = pFeatureSpace + 0x0161;

//Declare a pointer to a byte of the 32-bit data variable
uint8* pRandomDataPointer = &dwRandomData;

//Read one byte at a time, until all 32 bits are read.
for (int i = 0; i < 4; i++)
{
    *START_CRITICAL_SECTION*
    while ( !(pRNGStatus & 0x01))
    {
        //Data not available yet. Keep looping until bit 0 of
        //RNG Status register is set.
    }

    //RNG Status bit is set - read the RNG Data register
    (*pRandomDataPointer)++ = *pRNGData;
    *END_CRITICAL_SECTION*
}
```

## **Intel around the world**

### **United States and Canada**

Intel Corporation  
Robert Noyce Building  
2200 Mission College Boulevard  
P.O. Box 58119  
Santa Clara, CA 95052-8119  
USA  
Phone: (800) 628-8686

### **Europe**

Intel Corporation (UK) Ltd.  
Pipers Way  
Swindon  
Wiltshire SN3 1RJ  
UK

Phone:  
England (44) 1793 403 000  
Germany (49) 89 99143 0  
France (33) 1 4571 7171  
Italy (39) 2 575 441  
Israel (972) 2 589 7111  
Netherlands (31) 10 286 6111  
Sweden (46) 8 705 5600

### **Asia-Pacific**

Intel Semiconductor Ltd.  
32/F Two Pacific Place  
88 Queensway, Central  
Hong Kong, SAR  
Phone: (852) 2844 4555

### **Japan**

Intel Kabushiki Kaisha  
P.O. Box 115 Tsukuba-gakuen  
5-6 Tokodai, Tsukuba-shi  
Ibaraki-ken 305  
Japan  
Phone: (81) 298 47 8522

### **South America**

Intel Semicondutores do Brazil  
Rue Florida, 1703-2 and CJ22  
CEP 04565-001 Sao Paulo-SP  
Brazil  
Phone: (55) 11 5505 2296

### **For more information**

To learn more about Intel Corporation, visit our site  
on the World Wide Web at [www.intel.com](http://www.intel.com)

