R-IN32M3 Module (RY9012A0)

R30AN0452EJ0100
Rev.1.00
May.31.2024

DD Tool Guide

Introduction

This document describes the DD (Device Detection) tool for acquiring device information of the R-IN32M3 Module (RY9012A0).

Target Device

R-IN32M3 Module (RY9012A0)

Contents

List of Abbreviations and Acronyms

In this document, the terms below are defined as follows:

Terms	Description
DD	Device Detection
API	Application Programming Interface
GOAL	Generic Open Abstraction Layer See "R-IN32M3 Module User's Manual: Software API description” (R17US0002ED****)
HTTP	Hyper-Text Transfer Protocol

Related documents

Document Type	Document Title	Document No.
Data Sheet	R-IN32M3 Module Datasheet	R19DS0109ED****
User’s Manual	R-IN32M3 Module User’s Manual: Hardware	R19UH0122ED****
User’s Manual	R-IN32M3 Module User’s Manual: Software	R17US0002ED****
Quick Start Guide	R-IN32M3 Module Application Note: Quick Start Guide	R12QS0042ED****
Application Note	R-IN32M3 Module (RY9012A0) User's Implementation Guide	R30AN0386EJ****
User’s Manual	Adaptor Board with R-IN32M3 module YCONNECT-IT-I-RJ4501	R12UZ0094EJ****
Application Note	RA6M3/RA6M4 Sample application	R30AN0398EJ****
Application Note	Management Tool Instruction Guide	R30AN0390EJ****
Application Note	Software PLC Connection Guide TwinCAT	R30AN0380EJ****

1. Overview

1.1 Abstract

This document describes how to use the DD tool to obtain device information from R-IN32M3 Module (RY9012A0) Industrial Ethernet Communication Module by Renesas Electronics.
The DD tool is a tool that acquires and stores various device information stored in the R-IN32M3 Module over the network. It provides source code and libraries that run on each platform.

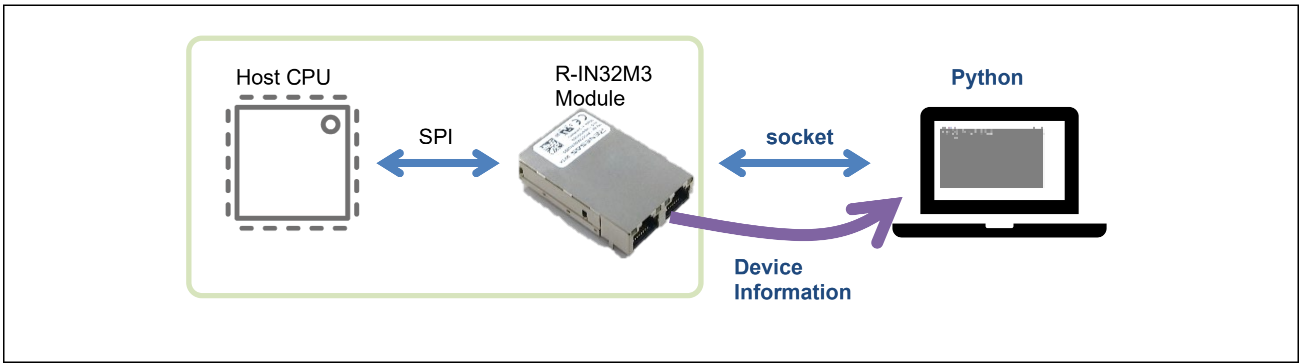

図 1-1 DD Tool

1.2 Operating Environment

This section describes the operating environment used by the DD tool.

1.2.1 Hardware Environment

The DD tool described in this document has been tested on the following hardware platforms:

(1) Combination of Adapter Board with R-IN32M3 Module and EK-RA6M3 or EK-RA6M4

(2) Combination of Adapter Board with R-IN32M3 Module and RL78/G14 (RTK5RLG140C00000BJ)

(3) RX66T CPU Card with R-IN32M3 Module

(1) R-IN32M3 Module + EK-RA6M4

(2) R-IN32M3 Module + RL78/G14 Board

(3) RX66T CPU Card with R-IN32M3 Module

Figure 1-1 R-IN32M3 Module Development Environment

Table 1-1 Hardware environments

Name	Type Name	Maker	Link	Note
Adapter Board with R-IN32M3 Module	YCONNECT-IT-I-RJ4501	Renesas Electronics Corporation	R-IN32M3-Module-Solution-Kit	
EK-RA6M3	RTK7EKA6M3S00001BU	Renesas Electronics Corporation	RA6M3 MCU Group Evaluation Board	
EK-RA6M4	RTK7EKA6M4S00001BE	Renesas Electronics Corporation	Evaluation Kit for RA6M4 MCU Group	
RL78/G14 Fast Prototyping Board	RTK5RLG140C00000BJ	Renesas Electronics Corporation	RL78/G14 Fast Prototyping Board	
RX66T CPU Card with R-IN32M3 Module	SEMB1320	SHIMAFUJI Electric Incorporated	https://www.renesas.com/SEMB1320	

1.2.2 Software environment

The operating environment for updating firmware is shown inTable 1-1.

Each software is confirmed in Windows 10 (64bit) environment.

Table 1-2 Operating Environments

Category	Name	Version	Link	Remarks
DD Tool (PyDD)	Python	V3.8.3 or later	https://www.python.org/	

2. DD Tool

This chapter describes the DD tool for obtaining device information of the R-IN32M3 Module in each platform. The following five device information can be obtained from R-IN32M3 Module with this tool.

・IP address

・Subnet Mask

・Gateway address

・DNS Server address（x2）

・DHCP enable/disable

2.1 PyDD

PyDD is a Python module in which the device detection protocol of R-IN32M3 Module is implemented. This allows device configuration by other applications such as web-based management software.

(1) Device detection in the network

To discover devices in the network, instantiate a DeviceDetectionProtocol that provides a file containing a database of GOAL variables.

```
from pydd import GoalDb, DeviceDetectionProtocol

# File containing the variable info data
goal_db_file = 'goal_db.json'

goal_db = GoalDb(goal_db_file)
proto = DeviceDetectionProtocol(goal_db)
```

The device can be detected by calling the scan function of the DeviceDetectionProtocol class, which provides the local IP to be used as the source address for sending UDP packets and a timeout. The following code example searches for all devices in the network and outputs the IP address and MAC address of each device found.

```
from pydd import GoalDb, DeviceDetectionProtocol, Device

# The local IP address
localip = "192.168.0.200"
# Default timeout in ms
timeout = 2000

devices = proto.scan(localip, timeout)

idx = 1
for device in devices:
    print("============ Device {id} ============".format(id = idx))
    print("IP:  " + device.ip)
    print("MAC: " + device.mac.hex(':').upper())
    idx += 1
    print()
```

As shown in the code example above, scan returns a list containing all devices found in the network expressed by the device instance.

(2) Read network parameters

The network parameters of the device can be read by calling readnetworkparams of the DeviceDetectionProtocol class. This function returns a tuple containing the IP settings of the device.

```
from pydd import GoalDb, DeviceDetectionProtocol, Device

# The local IP address
localip = "192.168.0.200"
# Default timeout in ms
timeout = 2000

print("============ Network Params ============")

(ip, netmask, gateway, dns0, dns1, dhcp_enabled) = proto.readnetworkparams(device, localip, timeout)

print("IP:.......... " + ip)
print("Netmask:..... " + netmask)
print("Gateway:..... " + gateway)
print("DNS0:........ " + dns0)
print("DNS1:........ " + dns1)
print("DHCP Enabled: " + ("Yes" if dhcp_enabled >0 else "No"))
```

This function returns the IP address, netmask, gateway, two DNS, and whether DHCP is enabled.

(3) Configuration of network parameters

Device network parameters can be set by calling writenetworkparams of the DeviceDetectionProtocol class. This function returns true if the operation is successful and false otherwise.

The following code example sets the IP address of each device in the list with a contiguous IP address.

```
from pydd import GoalDb, DeviceDetectionProtocol, Device

# The local IP address
localip = "192.168.0.200"
# Default timeout in ms
timeout = 2000

...

idx = 1
ip_start = "192.168.0."
ip_end = 101
netmask = "255.255.255.0"
gw = "192.168.0.1"
dns0 = "0.0.0.0"
dns1 = "0.0.0.0"
dhcp_enabled = False
activate = True
permanent = True
for device in devices:
    ip = ip_start + str(ip_end)
    ipdata = (ip, netmask, gw, dns0, dns1, dhcp_enabled)
    print("============ Network Params Device {id} ============".format(id = idx))

    res = proto.writenetworkparams(device, localip, timeout, ipdata, activate, permanent)

    print("Result: " + ("Success" if res else "Failed"))
    print()
    idx += 1
    ip_end += 1
```

As shown in the code example above, writenetworkparams requires the following parameters to be set.

- device : Device instance to which IP data is set
- localip : IP address to be used as the source address, specified in dot-delimited string notation (e.g., "192.168.0.1").
- timeout : timeout (msec)
- ipdata : Data including IP address, subnet mask, gateway, dns0, dns1, and whether DHCP is enabled or not. IP address is specified in dot-delimited string notation. dhcp_enabled is specified by a boolean value.
- activate : Set to True if the IP address setting will take effect immediately. Otherwise, the setting will be applied after the next reboot.
- permanent : Set to True if the IP address settings will be saved in flash; if False, the settings will be lost after rebooting.

(4) Execution example

A sample code, pydd_example.py, is included. An execution example is shown below.

```
C:¥Work¥temp¥pydd-v1.0.0>python pydd_example.py
============ Device 1 ============
IP:  192.168.0.100
MAC: [redacted]

============ Network Params Device 1 ============
IP:.......... 192.168.0.100
Netmask:..... 255.255.255.0
Gateway:..... 0.0.0.0
DNS0:........ 0.0.0.0
DNS1:........ 0.0.0.0
DHCP Enabled: No
```

Revision History

Rev.	Date	Description	
		Page	**Summary**
1.00	May/31/2024	-	First Edition

Trademark

- Ethernet is a registered trademark of Fuji Xerox Co., Ltd.
- Additionally, all product names and service names in this document are a trademark or a registered trademark which belongs to the respective owners.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance 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 the 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.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.