
RapID Platform Generation 2 (RPG2) Reference Manual**INTRODUCTION**

The RapID Platform Generation 2 (RPG2) ADIN2299 is a network interface solution that enables connectivity to devices that do not have this capability. The ADIN2299 supports multiple industrial Ethernet protocols with the same host processor software platform. The user simply interfaces with the ADIN2299 over a universal asynchronous receiver transmitter (UART) interface, a serial peripheral interface (SPI), or an Ethernet interface (raw socket connections with no need for a TCP/IP stack). The ADIN2299 can be used as a modular solution (see the [ADIN2299](#) data sheet) or as an embedded design version. The same software architecture outlined in this reference manual works for both the ADIN2299 as a module and as an embedded design. Any differences are noted in this reference manual.

TABLE OF CONTENTS

Introduction.....	1	RPG2 I/O Configuration Tool User Guide.....	82
RPG2 RapID Platform Generation 2 User Guide.....	4	General Description.....	82
Evaluation Kit Options.....	4	RPG2 I/O Configuration Tool Functionality and Installation.....	82
Evaluation Kit Contents.....	4	Configuration Objects.....	82
Additional Hardware Needed.....	4	Using the RPG2 I/O Configuration Tool.....	90
Documents Needed.....	4	Example Configurations.....	95
General Description.....	4	PROFINET GSDML Files.....	109
Evaluation Kit Photograph.....	5	Ethernet/IP EDS Files.....	110
Evaluation Board Setup.....	5	EtherCAT ESI Files.....	115
Protocol Specific Quickstart Guides.....	7	POWERLINK XDD Files.....	129
RPG2 EtherCAT Quickstart Guide.....	8	RPG2 Programming User Guide.....	132
Features.....	8	Features.....	132
Evaluation Kit Contents.....	8	Equipment Needed.....	132
Equipment Needed.....	8	Documents Needed.....	132
Software Needed.....	8	Software Needed.....	132
General Description.....	8	General Description.....	132
EtherCAT Example Application Setup.....	8	Communications Controller Applications and Images.....	133
Network Interface Example Application Suite With PC Running TwinCAT.....	10	Needed Images for the RPG2 Solution.....	134
Next Step: The Design Phase.....	33	Programming by Means of the API.....	136
RPG2 Profinet Quickstart Guide.....	34	Programming by Means of the Embedded Web Server.....	138
Features.....	34	Programming by Means of JTAG.....	140
Equipment Needed.....	34	Programming Methods by Use Case.....	142
Software Needed.....	34	Programming Methods by State of Design.....	143
General Description.....	34	RPG2 Hardware Design Integration Guide.....	145
Evaluation Kit Setup for PROFINET.....	34	Features.....	145
Network Interface Application Suite With a PC Running the Siemens TIA Portal.....	35	Evaluation Kit Contents.....	145
Next Step: The Design Phase.....	49	Equipment Needed.....	145
RPG2 EtherNet IP Quickstart Guide.....	50	Documents Needed.....	145
Features.....	50	Software Needed.....	145
Equipment Needed.....	50	General Description.....	145
Software Needed.....	50	Development Overview.....	145
General Description.....	50	Introduction.....	146
Evaluation Kit Setup for EtherNet/IP.....	50	Hardware Integration.....	148
Network Interface Application Suite With a PC Running RS Logix.....	52	Software Integration.....	151
Next Step: Design Phase.....	56	Considerations for Production and Maintenance.....	152
RPG2 Unified Interface User Guide.....	58	Connecting Timers for RPG2 Embedded Reference Design.....	153
Introduction.....	58	LED Behavior.....	154
Block Diagram for Embedded Design and Module Users.....	58	RPG2 Web Server User Guide.....	158
Unified Interface Background.....	58	Features.....	158
Design Flow Using the Unified Interface.....	59	Equipment Needed.....	158
Application Processor Link Porting Layer.....	59	Documents Needed.....	158
Porting and Customization.....	61	Software Needed.....	158
Unified Interface API.....	69	General Description.....	158
Application Processor Link Type.....	80	Introduction.....	158

TABLE OF CONTENTS

RPG2 Web Server Security.....	159	Example RPG2 Web Server Content.....	169
CI Item, Name Reference Syntax.....	161	Firmware Upgrade.....	177
NonCI Item, Name Syntax.....	163	MbedTLS.....	178
SSI Directives.....	164	Loading RPG2 Web Server Files.....	178
CGI Functions.....	165	Notes.....	180

REVISION HISTORY

2/2024—Rev. 0 to Rev. A

Changes to Features Section.....	34
Changes to Software Needed Section.....	34
Changed Network Interface Application Suite With a PC Running the TIA Portal Section to Network Interface Application Suite With a PC Running the Siemens TIA Portal Section.....	35
Changes to Network Interface Application Suite With a PC Running the Siemens TIA Portal Section.....	35
Changes to Installing the General Description File Section.....	41
Added Configuring the Optional PROFINET IRT Section.....	46
Added Figure 90 to Figure 92; Renumbered Sequentially.....	47
Changes to Table 14.....	83
Added Current IRT Example PROFINET Configuration Section and Figure 129 to Figure 131.....	98
Changes to PROFINET GSDML Files Section.....	109
Changes to Table 27.....	110
Change to Table 45.....	150

7/2023—Revision 0: Initial Version

RPG2 RAPID PLATFORM GENERATION 2 USER GUIDE**EVALUATION KIT OPTIONS**

- ▶ EtherCAT: EV-RPG2-ECZ
- ▶ EtherNet/IP: EV-RPG2-ENZ
- ▶ Profinet: EV-RPG2-PNZ
- ▶ Powerlink: EV-RPG2-PLZ
- ▶ Modbus: EV-RPG2-MBZ

EVALUATION KIT CONTENTS

- ▶ Baseboard
 - ▶ RapID Generation 2 module installed and preloaded with protocol specific software
- ▶ Wall mount AC adapter, 90 V ac to 264 V ac to 12 V dc, 12 W, 1 A
- ▶ 4 power supply plug adapters (A, C, G, and I types)
- ▶ 1 USB A male to USB micro B male cable
- ▶ Ethernet cable

ADDITIONAL HARDWARE NEEDED

- ▶ PC with Windows® 10 operating system

DOCUMENTS NEEDED

- ▶ Protocol specific user guides are available in this reference manual as follows:
 - ▶ [RPG2 EtherCAT Quickstart Guide](#) section
 - ▶ [RPG2 EtherNet IP Quickstart Guide](#) section
 - ▶ [RPG2 Profinet Quickstart Guide](#) section

GENERAL DESCRIPTION

This user guide describes how to set up the provided hardware of the evaluation kit and establish the link type specific host connection and connect the evaluation board to an industrial Ethernet network.

The operation of the EV-RPG2 boards vary by Industrial Ethernet Protocol, and separate user guides are available for each protocol on this reference manual.

The EV-RPG2 evaluation kits provide an end to end evaluation of the communication path from the host processor to the programmable logic controller (PLC) or PC-based tool.

The EV-RPG2 evaluation kits can completely verify the communication path between the host processor and a PLC before integrating the network interface into the end field device.

RPG2 RAPID PLATFORM GENERATION 2 USER GUIDE

EVALUATION KIT PHOTOGRAPH

Figure 1 shows the contents of the evaluation kit.

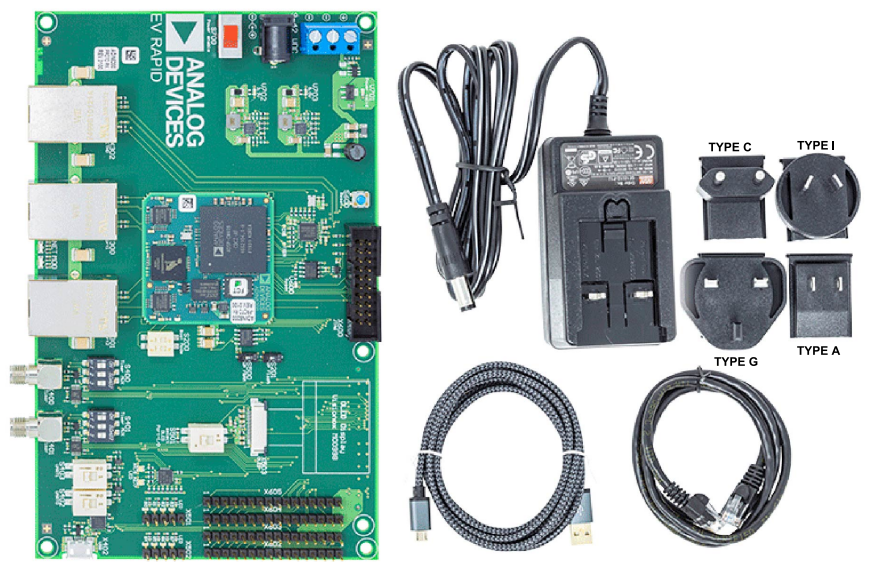


Figure 1. EV-RPG2 Evaluation Kit Example

EVALUATION BOARD SETUP

Electrostatic Sensitive Device (ESD) Warning

Handle the hardware in this evaluation kit in accordance with proper ESD device handling techniques. Use a grounding wrist strap when using this evaluation board to prevent accidental damage to the hardware.

Power Supply and Grounding

A wall adapter power supply is included with the evaluation kit. This power supply is 12 V, 12 W, 1 A, and attaches to the barrel connector on the evaluation board. A screw terminal on the evaluation board allows attachment of a lab benchtop power supply. For either type of power supply connection, the input voltage to the board must be in a voltage range of 9 V to 42 V.

Hardware Setup

The following steps detail the hardware setup process. The evaluation kit comes preinstalled jumpers and preset switches. To set up the evaluation kit, take the following steps.

1. Plug the provided AC adapter into the wall socket, and connect the barrel connector of the AC adapter to the baseboard, as shown in [Figure 2](#). The Power Valid LED illuminates blue. At this point, the evaluation board is powered.

RPG2 RAPID PLATFORM GENERATION 2 USER GUIDE

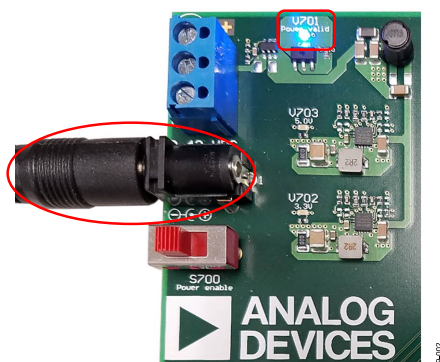


Figure 2. Power Cable Connection and Power Valid LED

2. Switch the Power Enable switch on the baseboard to the **ON** position, as shown in Figure 3. The 3.3 V LED (V702) and the 5.0 V LED (V703) illuminate blue.

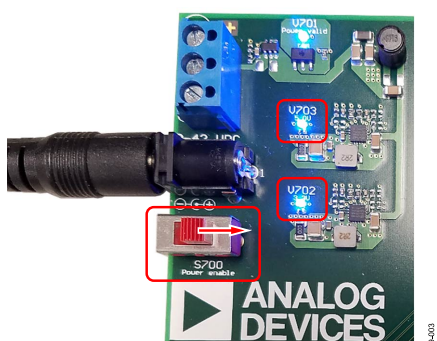


Figure 3. Power Switch in ON position and 5 V and 3.3 V LEDs

3. If it is necessary to reset the evaluation board, press the **Reset** switch, which illuminates two red LEDs, as shown in Figure 4.

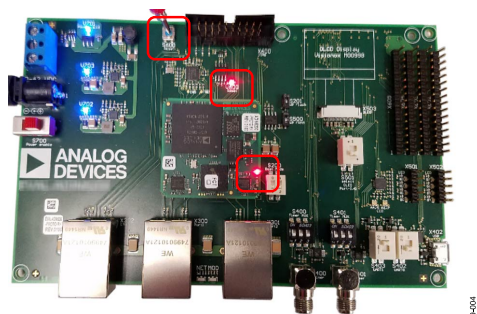


Figure 4. Reset Switch Location and Reset LEDs

RPG2 RAPID PLATFORM GENERATION 2 USER GUIDE

Connecting the Ethernet Host Interface

Install the provided Ethernet cable between the host PC and the host port of the evaluation board as shown in [Figure 5](#).

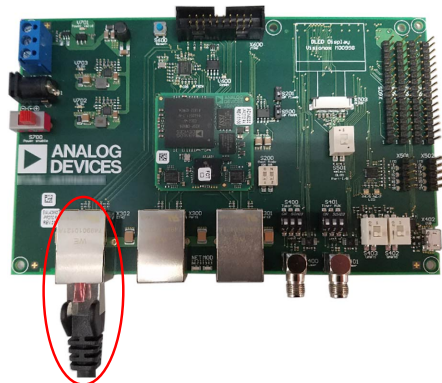


Figure 5. Ethernet Cable Connection

Connecting to the Industrial Ethernet

Use the two industrial Ethernet ports on the baseboard to connect the evaluation board to an industrial Ethernet network, as shown in [Figure 6](#). The two industrial Ethernet ports are labeled Port 1 and Port 2 on the evaluation board.

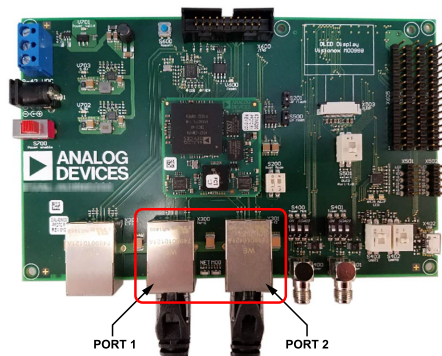


Figure 6. Network Interface Port Location

PROTOCOL SPECIFIC QUICKSTART GUIDES

After running the application example, the user can refer to the protocol specific user guide for an example of the network communication. The evaluation kit and protocol specific user guide must be used in conjunction with the [RPG2 Hardware Design Integration Guide](#) section to develop field device products. These user guides are available in this reference manual.

RPG2 ETHERCAT QUICKSTART GUIDE

FEATURES

- ▶ Cyclic input data transfer up to 1440 bytes
- ▶ Cyclic output data transfer up to 1440 bytes
- ▶ Cycle time down to 1 ms
- ▶ 8 SyncManager entities
- ▶ 8 FMMU entities
- ▶ Distributed clock support
- ▶ Supports the modular device profile (MDP) and the CAN application protocol over EtherCAT (CoE)

EVALUATION KIT CONTENTS

- ▶ 1 baseboard with the RPG2 module installed with the protocol specific software preloaded
- ▶ 1 wall mount AC adapter, 90 V_{AC} to 264 V_{AC} to 12 V_{DC}, 12 W, 1 A
- ▶ 4 power supply plug adapters (Type A, Type C, Type G, and Type I)
- ▶ 1 USB A male to USB Micro B male cable
- ▶ 1 Ethernet cable

EQUIPMENT NEEDED

- ▶ EV-RPG2-ECZ evaluation kit
- ▶ 1 PC (2 PCs recommended for simplified use of the evaluation board)

SOFTWARE NEEDED

- ▶ [Network Interface Example Application Suite](#)
- ▶ Beckhoff TwinCAT Version 4024.20 or higher version for best results
- ▶ WinPCap

GENERAL DESCRIPTION

The RapID Platform Generation 2 (RPG2) module is a pretested, industrial network interface designed to manage industrial protocols and network traffic. The RPG2 module supports PROFINET®, PROFINET isochronous real-time (IRT), Ethernet/IP®, Ethernet/IP with device level ring (DLR), EtherCAT®, and Modbus/TCP. The RPG2 module uses the Unified Interface to communicate with different protocols.

The Unified Interface is a custom protocol by Analog Devices, Inc., that allows interaction between an application processor and the RPG2 module. The Unified Interface is agnostic of the industrial protocol.

The Unified Interface ensures that the application processor hardware and software interface is not required to change when switching or updating protocols. The RPG2 module connects to an application processor via a universal asynchronous receiver transmitter (UART), Ethernet, or serial peripheral interface (SPI).

The EV-RPG2-ECZ evaluation kit provides end to end evaluation of the communication path from the application processor to the programmable logic controller (PLC) over the industrial Ethernet interface by using the [Network Interface Example Application Suite](#). This user guide describes how to use the EV-RPG2-ECZ evaluation kit to set up and run a PLC example application.

For the example described in this user guide, the application processor is a PC and communicates to the RPG2 module via an Ethernet network interface card (NIC).

ETHERCAT EXAMPLE APPLICATION SETUP

Refer to the [RPG2 RapID Platform Generation 2 User Guide](#) section to set up the hardware for the EV-RPG2-ECZ. The default link type is Ethernet. [Figure 7](#) and [Figure 8](#) show the hardware setup using the default link type, depending on whether one or two PCs are in use. If users want to change the link type to UART, refer to the [Link Configuration File](#) section for instructions on how to reprogram the EV-RPG2-ECZ.

The user can choose to run the TwinCAT application and [Network Interface Example Application Suite](#) on a single PC. However, it is ideal to use two PCs so that each application is clearly distinguished on each PC.

This user guide describes the two PC configuration to set up the TwinCAT application and run the [Network Interface Example Application Suite](#).

RPG2 ETHERCAT QUICKSTART GUIDE

Note that in [Figure 7](#) and [Figure 8](#), the A6/7 LED is green, which indicates that the evaluation board is preloaded with EtherCAT and that the startup process is complete.

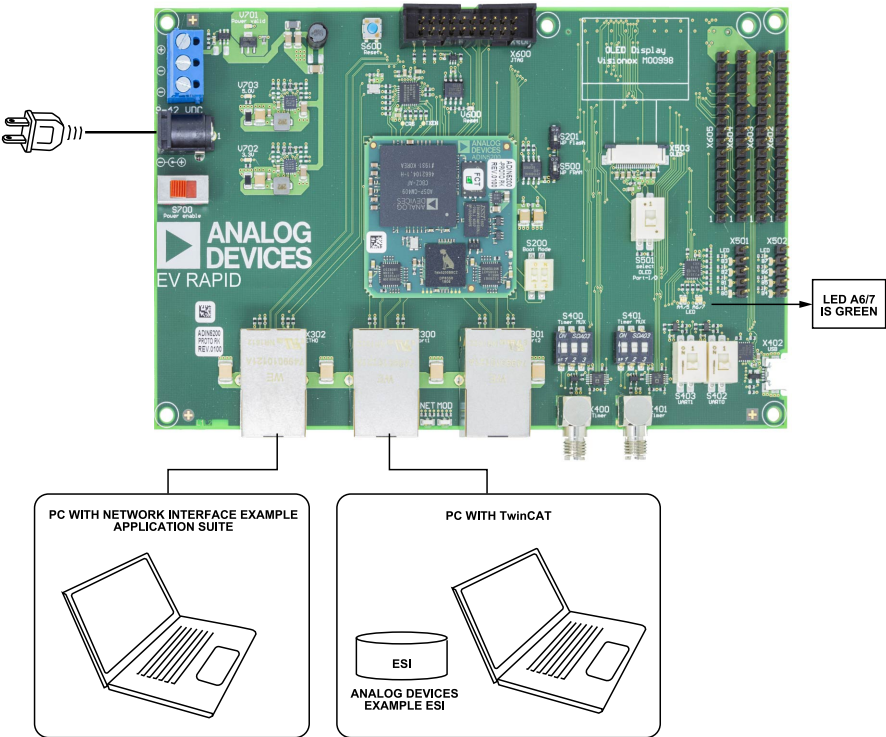


Figure 7. EtherCAT Example Application Setup, Ethernet (Default) Link with Two PCs (ESI Is EtherCAT Slave Information)

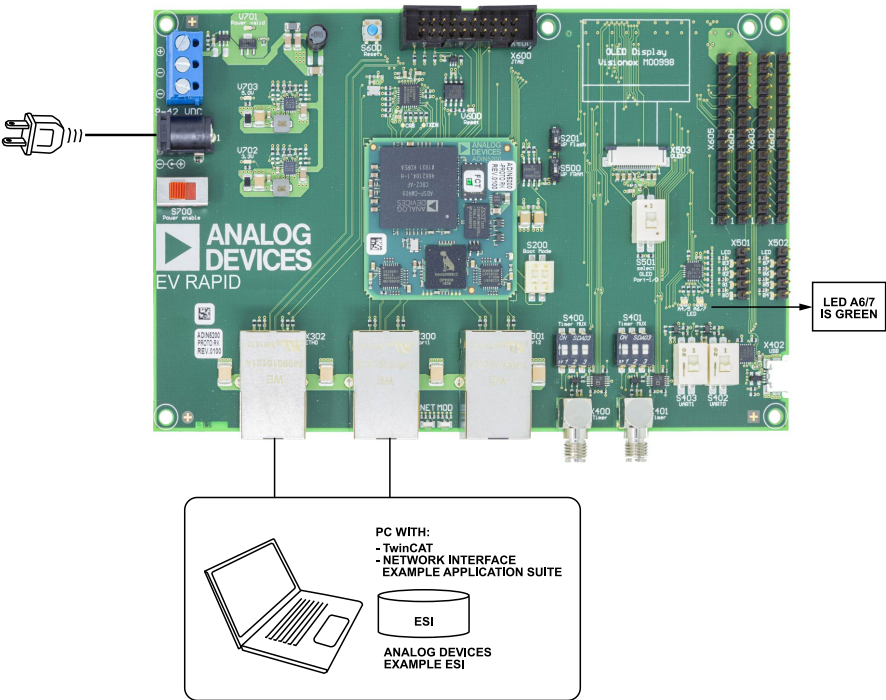


Figure 8. EtherCAT Example Application Setup, Ethernet (Default) Link with One PC

RPG2 ETHERCAT QUICKSTART GUIDE

NETWORK INTERFACE EXAMPLE APPLICATION SUITE WITH PC RUNNING TWINCAT

This TwinCAT-based PLC example application is used with a PC. However, this application can run with a regular PLC, which is the recommended method for real-time systems.

Communication between the application processor and the RPG2 module on the baseboard is enabled by using an Ethernet NIC. In addition to the evaluation kit, a PC running the [Network Interface Example Application Suite \(ni-example-app.exe\)](#) and a PC running the TwinCAT application is required.

To see the syntax to start up the example application using any link type, type **ni-example-app.exe -h**. Note that this user guide covers the Ethernet link type only (see [Figure 9](#)). Ensure that the identifier for the NIC (local NET device) is set to use. Run **ni-example-app.exe -l ETH** to see a list of local NET devices.

Obtaining the Network Interface Example Application Suite

To obtain the [Network Interface Example Application Suite](#) from the ADIN2299 product page, take the following steps:

1. Go to the ADIN2299 [main product page](#).
2. Click the **RPG2 Network Interface Example Application Suite (ZIP)** link.
3. Review the **Terms and Conditions** and accept these conditions to proceed.
4. Extract the contents of the .zip file. This file includes the **ni-example-app.exe** executable file that must run from the command line (see [Figure 9](#)).

When using the software, if a message appears saying **FATAL Bad Memory Block** while running the [Network Interface Example Application Suite](#), ensure that the most recent version of WinPcap is on the PC. Note that WinPcap must be installed on the PC executing the [Network Interface Example Application Suite](#). WinPcap is a packet capturing library that is used by the [Network Interface Example Application Suite](#) to examine, capture, and transmit network packets. Go to the WinPcap website and follow the instructions to download the appropriate version. The NIC identifier begins with **DeviceNPF_**, as shown in [Figure 9](#).

```

Administrator: C:\WINDOWS\system32\cmd.exe - ni-example-app.exe -l ETH -n DeviceNPF_{F348E065-828C-4F40-8D0B-2D0861DAE9C8}

C:\RPG2 Network Interface Example Application Suite\ni-example-app\project\vs\ni-example-app\release>ni-example-app.exe -l ETH -n
DeviceNPF_{F348E065-828C-4F40-8D0B-2D0861DAE9C8}
[...\\src\\main.c:75] INFO: Welcome to ni-example-app!

[...\\src\\main.c:78] INFO: Processing arguments...
[...\\src\\main.c:86] INFO: DONE
[...\\src\\main.c:90] INFO: Performing system startup...
[...\\src\\main.c:96] INFO: DONE
[...\\src\\main.c:100] INFO: Start time update process...
[...\\src\\main.c:106] INFO: DONE
[...\\src\\main.c:110] INFO: Starting up example application...
[...\\src\\main.c:137] INFO: DONE
[...\\src\\main.c:140] INFO: Link init...
[...\\src\\main.c:146] INFO: DONE
[...\\src\\main.c:150] INFO: Unified Interface stack init...
[...\\src\\main.c:159] INFO: DONE
[...\\src\\main.c:164] INFO: Set response timeout...
[...\\src\\main.c:170] INFO: DONE
[...\\src\\main.c:175] INFO: Find Network Interface...
[...\\src\\main.c:186] INFO: DONE
[...\\src\\main.c:190] INFO: Set transmit modes...
[...\\src\\main.c:204] INFO: DONE
[...\\src\\main.c:208] INFO: Get installed protocol...
[...\\src\\main.c:220] INFO: DONE (EtherCAT)
[...\\src\\main.c:231] INFO: Set device 400...
[...\\src\\main.c:292] INFO: DONE
[...\\src\\main.c:296] INFO: Add item 500 to location 1...
[...\\src\\main.c:307] INFO: DONE
[...\\src\\main.c:311] INFO: Add item 501 to location 2...
[...\\src\\main.c:322] INFO: DONE
[...\\src\\main.c:326] INFO: Add item 502 to location 3...
[...\\src\\main.c:337] INFO: DONE
[...\\src\\main.c:342] INFO: Finalize configuration...
[...\\src\\main.c:353] INFO: DONE
  
```

Figure 9. Network Interface Example Application Suite

Configuring TwinCAT as the PLC

This section provides instructions for setting up and using the RPG2 module with the link type selected as Ethernet. The RPG2 module is connected to a PC running the [Network Interface Example Application Suite](#) (which is the leader network program) and TwinCAT to evaluate the EV-RPG2-ECZ and observe the flow of the packets.

The examples described in this user guide make use of a PC running the TwinCAT application with a real-time capable NIC. This PLC example application assumes the RPG2 module is configured as detailed in [Table 1](#).

RPG2 ETHERCAT QUICKSTART GUIDE

This configuration defines three items with different input and output types. Item Number 500 defines 16 digital inputs and 16 digital outputs (2 bytes of input and output data each), Item Number 501 defines two analog inputs and two analog outputs (2 bytes per channel for a total of 4 bytes of input and output data), and Item Number 502 represents 2 bytes of control data.

It is recommended to use TwinCAT Version 4024.20 for optimum system performance.

Table 1. RPG2 Module Input and Output Configurations

Item Number	Input Size (Bytes)	Output Size (Bytes)	Description
500	2	2	Digital input/output
501	4	4	Analog input/output
502	0	2	Control

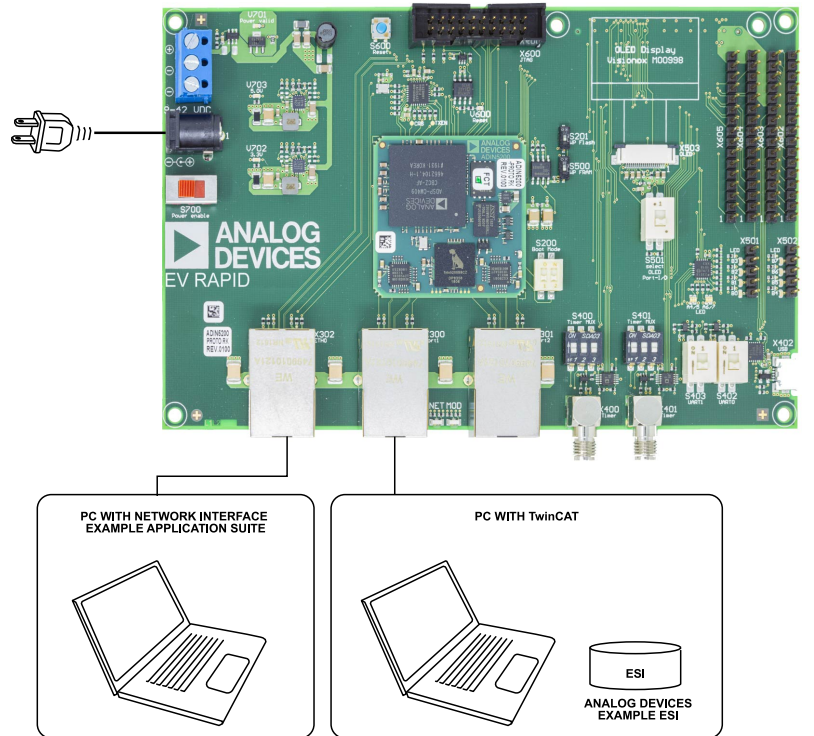


Figure 10. Test Network Setup with Two PCs

The remainder of this user guide describes the setup shown in [Figure 10](#). For details regarding the setup of the test network, see the [EtherCAT Example Application Setup](#) section.

Take the following steps to configure TwinCAT as the PLC:

1. Start the TwinCAT application.
2. Go to **File** and select **New Project** from the dropdown menu. Rename the project as desired. For this example, name the project **EtherCAT Quickstart**, and click **OK** to display the New Project screen (see [Figure 11](#)).
3. Ensure the .esi file is installed. The default path is **C:\TwinCAT3.1\ConfigloEtherCAT**. Place the example ESI file (**Analog-Devices-RapID-Example.xml**) in this location.
4. Go to **TwinCAT > EtherCAT Devices** and select **Reload Device Descriptions** (see [Figure 12](#)).
5. Select **I/O** within the **Solution Explorer** pane if it is not already expanded and right-click **Devices** under **I/O** (see [Figure 13](#)).
6. Select **Scan** in the **Devices** pulldown menu (see [Figure 13](#)) and the scan warning message shown in [Figure 14](#) appears.
7. Click **OK** and the NIC selection list shown in [Figure 15](#) (or a similar message, depending on the NIC used) appears. Choose the desired NIC.

In the event that the **No new I/O devices found** message appears, ensure that the following is done:

RPG2 ETHERCAT QUICKSTART GUIDE

- ▶ Locate the **Network Connections** option in the **Windows Control** panel. Right-click the TwinCAT NIC and ensure that the **TwinCAT Ethernet Protocol** option is checked. Uncheck all other items for this connection.
- ▶ In the TwinCAT application window, click **TwinCAT** in the menu bar and click **Show Realtime Ethernet Compatible Devices**. If the TwinCAT NIC is under the **Installed and ready to use devices (real-time capable)** section then the real-time driver is installed correctly.

If the TwinCat NIC is not installed correctly, it appears under **Compatible Devices**. To install the driver, highlight the TwinCAT NIC and click **Update List**. If you are installing the TwinCAT NIC for the first time, you may have to repeat this step to confirm that the driver is installed.

1. Click **OK** to add an EtherCAT device to the **Devices** node with the EtherCAT icon, which is the red icon next to **Device 5 (EtherCAT)** within the **Solution Explorer** pane (see Figure 16). The message in Figure 16 then appears asking the user to **Scan for boxes**.
2. Click **Yes**. The box descriptor is then identified as **Box 1 (Rapid Platform Network Interface)**, see Figure 17.
3. Click **Yes to Activate Free Run**, as shown in Figure 18.
4. EEPROM errors will appear in the **Error List** section (see Figure 19) and the **MOD LED** on the baseboard flashes red.
5. Program the EEPROM as described in the [Imaging the EEPROM in TwinCAT](#) section to resolve these errors. Note that this step must be executed every time the device descriptions (see Step 4) change and load.
6. Follow the steps in the [Imaging the EEPROM in TwinCAT](#) section to test the EtherCAT network.

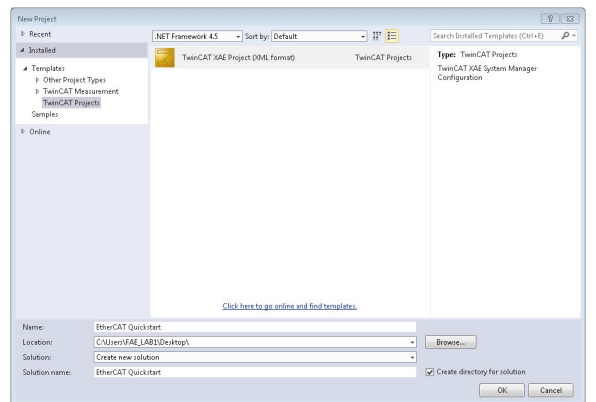


Figure 11. New Project Window

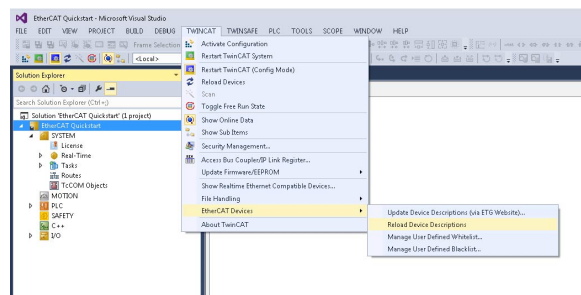


Figure 12. Reload Device Descriptions Pulldown Menu

RPG2 ETHERCAT QUICKSTART GUIDE

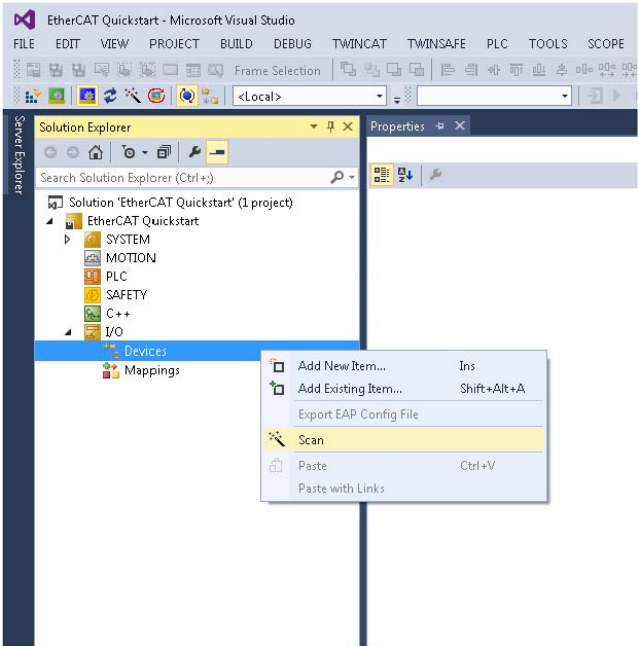


Figure 13. Scan Device

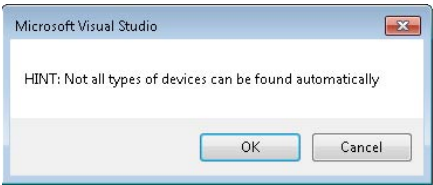


Figure 14. Scan Warning

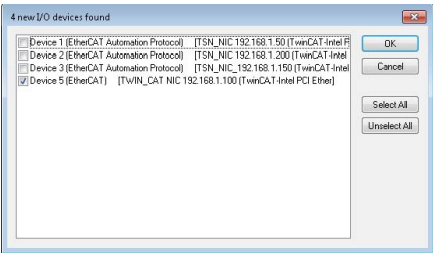


Figure 15. NIC Selection

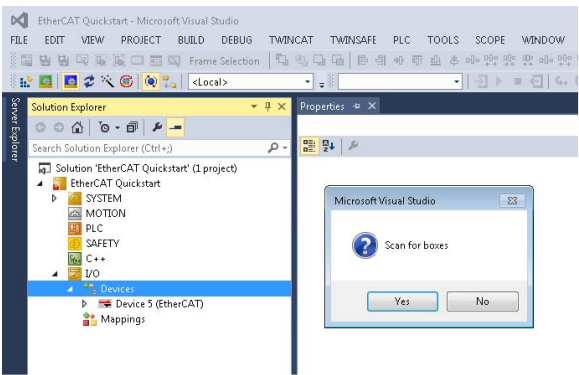


Figure 16. Scan for Boxes Dialog Box

RPG2 ETHERCAT QUICKSTART GUIDE

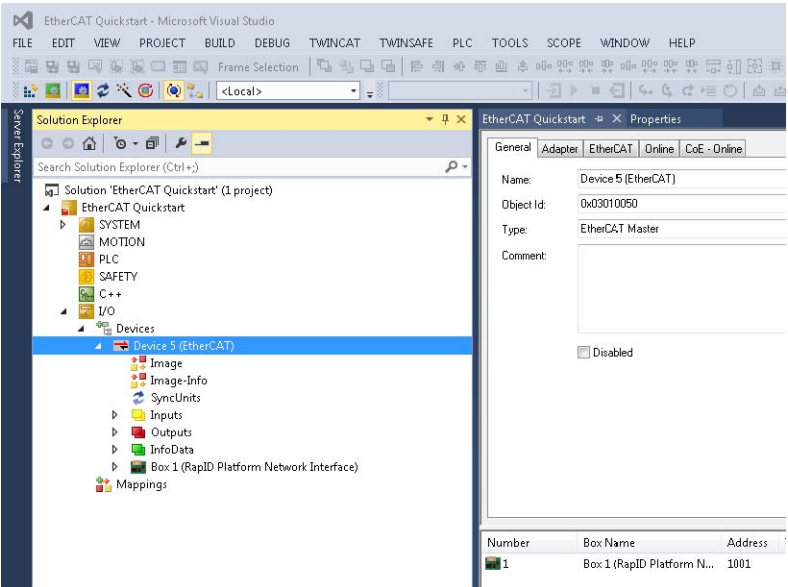


Figure 17. TwinCAT with Rapid Found



Figure 18. Activate Free Run Dialog Box

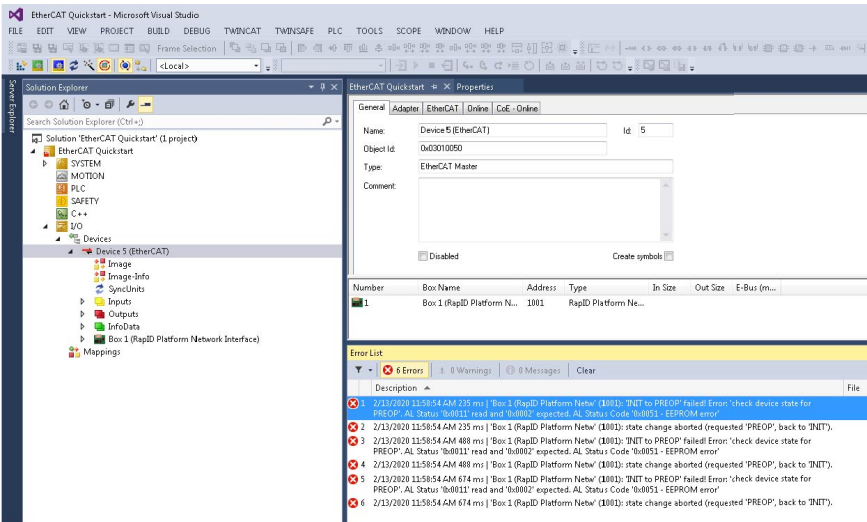


Figure 19. EEPROM Error List Section

RPG2 ETHERCAT QUICKSTART GUIDE

Imaging the EEPROM in TwinCAT

If the device has never been used in an EtherCAT network, image the EEPROM for the RPG2 module with TwinCAT.

Note that the EEPROM of the system is emulated. The evaluation kit does not have an EEPROM chip, rather, the EEPROM data is stored in a serial flash chip on the RPG2 module.

To image the EEPROM data, take the following steps:

1. Under the **Device 5 (EtherCAT)** node in the **Solution Explorer** pane, click the **Box 1 (RapID Platform Network Interface)** expandable list and select the **EtherCAT** tab.
2. Click **Advanced Settings** (see Figure 20).
3. In the **Advanced Settings** window, navigate to **ESC Access > E²PROM > Hex Editor** and click **Download from List** as shown in Figure 21.
4. Highlight the **RapID Platform Network Interface (-238005760/65537)** icon and click **OK** (see Figure 22).
5. Wait until the status in the pane at the bottom of TwinCAT changes from **Writing** (see Figure 23) to **Ready** (see Figure 24). Close the **Advanced Settings** dialog box.
6. Power cycle the RPG2 module and restart the [Network Interface Example Application Suite](#).
7. In the **Solution Explorer** pane, delete the **Box 1 (RapID Platform Network Interface)** node, right-click **Device 5 (EtherCAT)**, and select **Scan** (see Figure 25) to go back to the TwinCAT home screen (see Figure 26), where the **Activate Free Run** message now appears (see Figure 18).
8. Click **Yes** and then navigate to the **Online** tab. Notice the **Current State** and **Requested State** boxes are both set to **OP** (see Figure 27). A ladder logic program can now be created that toggles a bit in Item 500.

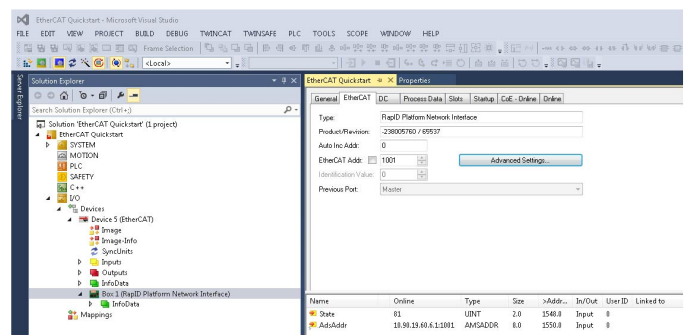


Figure 20. EtherCAT Tab

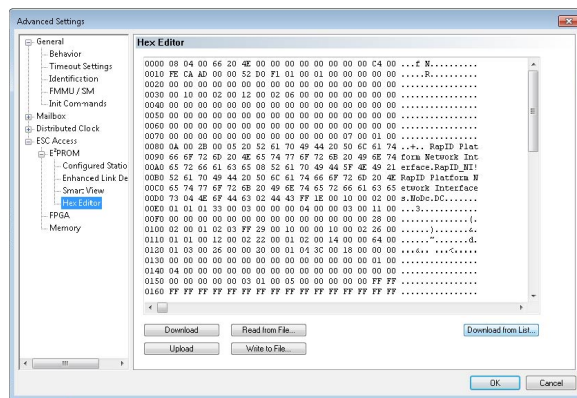


Figure 21. EtherCAT Options EEPROM Navigation

RPG2 ETHERCAT QUICKSTART GUIDE

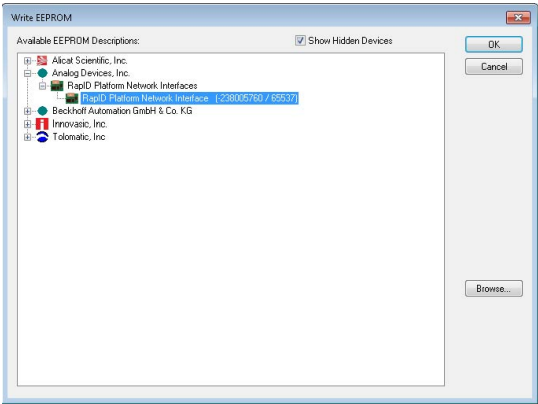


Figure 22. RapidID ESI File



Figure 23. Status Pane: Writing EEPROM



Figure 24. Status Pane: Ready

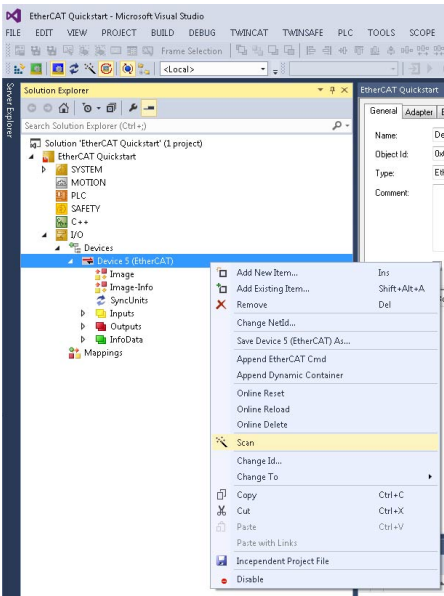


Figure 25. Scan After Imaging EEPROM

RPG2 ETHERCAT QUICKSTART GUIDE

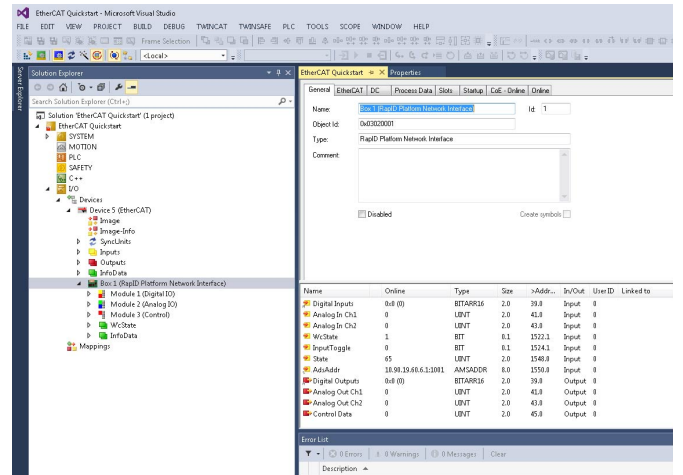


Figure 26. TwinCAT Home View

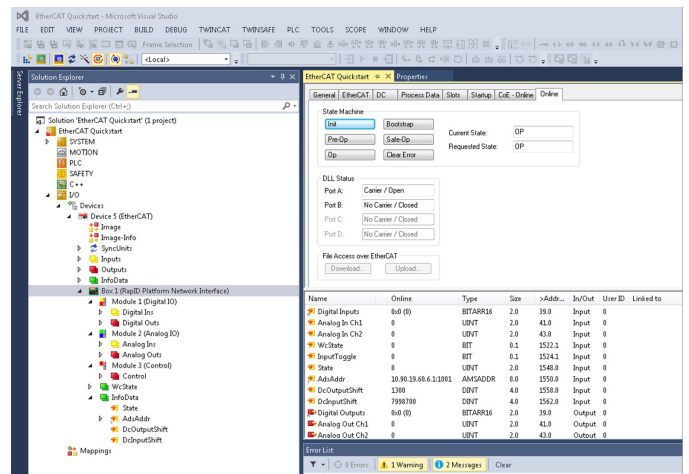


Figure 27. EtherCAT State Machine

Simple Data Flow Example

The following is a simple data flow example:

1. Expand the **Box 1 (RapID Platform Network Interface)** node and all three modules as shown in [Figure 28](#).
2. Click the **Box 1 (RapID Platform Network Interface)** node and select the **CoE-Online** tab. Expand the digital, analog, and control data indices as shown in [Figure 29](#).
3. Ensure the **Current State** is set to **OP** under the **Online** tab before writing a value to the outputs.
4. Right-click **Digital Outs** in the **Solution Explorer** pane (see [Figure 30](#)) or the **ATTRIBUTES PANE** (see [Figure 31](#)) and select **Online Write**.
5. The **Set Value Dialog** box (see [Figure 32](#)) opens, edit the value, and click **OK**. The corresponding **Digital Ins** value then updates to match the **Digital Outs** value.
6. Repeat Step 4 and Step 5 for **Analog Out Ch1**, **Analog Out Ch2**, and **Control Data** to see the corresponding input values change.
7. Verify that the inputs match the outputs under the **CoE-Online** tab (see [Figure 33](#)) and under the **Online** tab for the **ATTRIBUTES PANE** (see [Figure 31](#)).
8. The **Current Output Data** then updates in the [Network Interface Example Application Suite](#) (see [Figure 34](#)).

RPG2 ETHERCAT QUICKSTART GUIDE

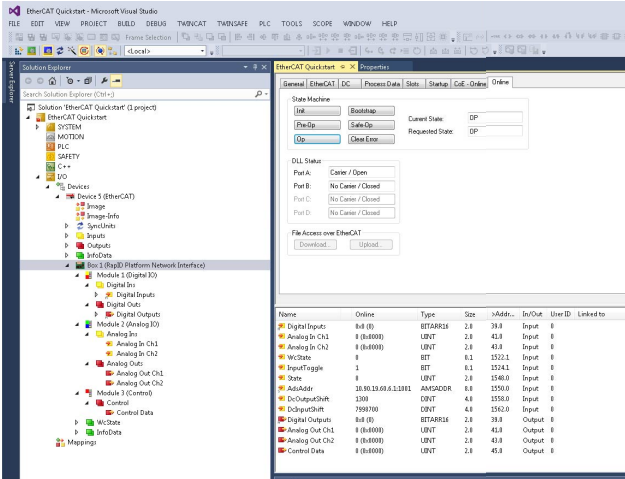


Figure 28. Expanded Modules

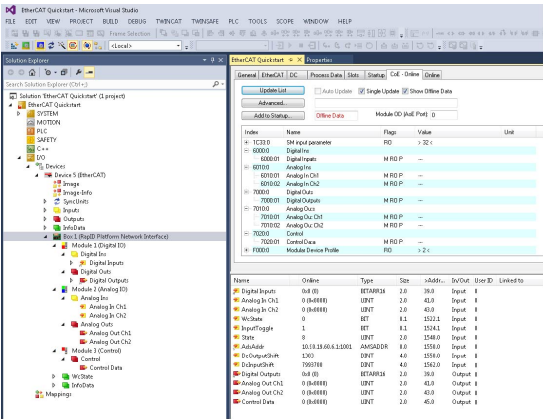


Figure 29. Expanded Indices Within CoE-Online Tab

RPG2 ETHERCAT QUICKSTART GUIDE

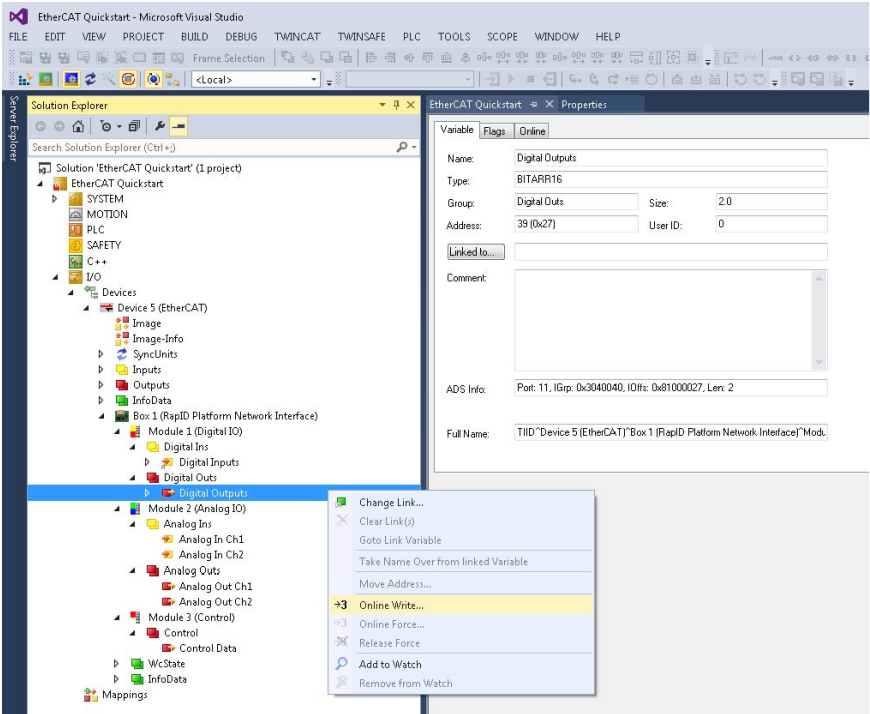


Figure 30. Online Write via Box 1 (RapID Platform Network Interface) Node < Module 1 (Digital IO) < Digital Outs < Digital Outputs within the Solution Explorer Pane

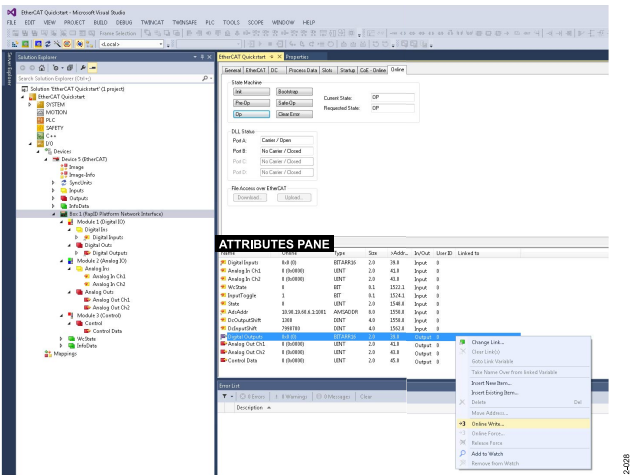


Figure 31. Online Write via Box 1 (RapID Platform Network Interface) Node Within the ATTRIBUTES PANE < Digital Outputs

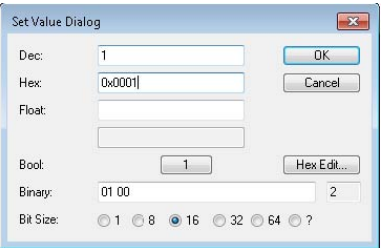


Figure 32. Set Value Dialog Box

RPG2 ETHERCAT QUICKSTART GUIDE

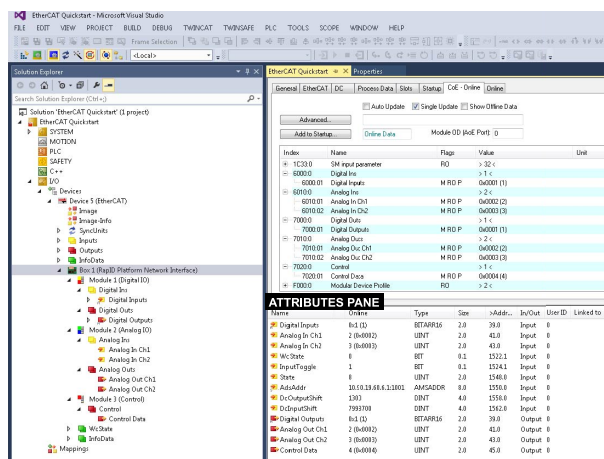


Figure 33. Verify Online Writes Were Successful in TwinCAT Within the CoE-Online Tab

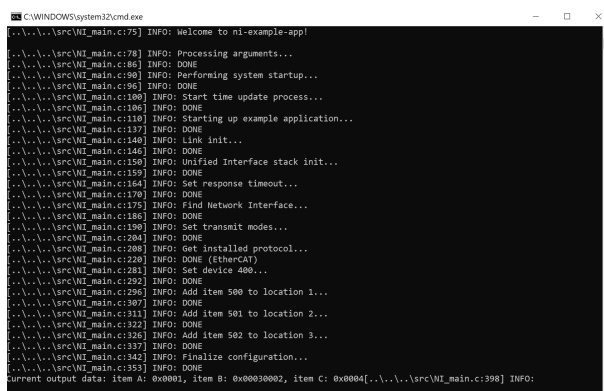


Figure 34. Verify Online Writes Successful in Network Interface Example Application Suite

Running the PLC Example Application

This section demonstrates how to automatically toggle the digital input and output data. For instructions on manually writing the outputs one by one to see the change in the corresponding inputs, see the [Simple Data Flow Example](#) section.

To automatically toggle the digital input and output data, take the followings steps:

1. Within the **Solution Explorer** pane, go to **Solution 'EtherCAT Quickstart' (1 project)**, right-click **PLC**, and select **Add New Item** (see [Figure 35](#)).
2. In the **PLC Project Creation** window that then appears (see [Figure 36](#)), select the **Standard PLC Project** and give the project a name. In this example, the item is named **EtherCAT_Quickstart_PLC**. After naming the project, click **Add**. The new project appears on the left-hand side of the **Solution Explorer** pane under the **Solution 'EtherCAT Quickstart' (1 project)** expandable list (see [Figure 37](#)).
3. Click **EtherCAT_Quickstart_PLC Project** for the **PLC Project Expanded View** to appear (see [Figure 38](#)).
4. Right-click **GVLs** and navigate to the **Add > Global Variable List** (see [Figure 39](#)).
5. Click **Global Variable List** and rename this variable as desired. In this example, the variable name is **RapidQSG** (see [Figure 40](#)), and click **Open**. Once the variable is named, the **Global Variable View** window appears (see [Figure 41](#)).

The input and output variables are declared in the **RapidQSG.TcGVL** file. Declare that the **RapidInput** and **RapidOutput** variables are **RapidInput AT %I* : UINT** and **RapidOutput AT %Q* : UINT**, and save the file (see [Figure 42](#)).

1. The global variables are now set up, and the user must add a ladder logic diagram. Go to the **POUs** pulldown menu and select **Add > POU**. Name the ladder logic diagram **Quickstart_Ladderlogic**. Ensure that the implementation language option is set to **Ladder Logic Diagram (LD)** and click **Open** as shown in [Figure 43](#).

RPG2 ETHERCAT QUICKSTART GUIDE

2. The screen now appears as shown in [Figure 44](#) with **Quickstart_Ladderlogic** on the left and a location for the ladder logic in a new window to right.
3. Open the **MAIN (PRG)** tab. On the first line, type **Quickstart_Ladderlogic()**; (see [Figure 45](#)).
4. Navigate back to **Quickstart_Ladderlogic (PRG)**. Click the pink highlighted area shown in [Figure 46](#) and click the **Insert Negated Contact** button shown in the upper left portion of the screen.
5. The **Negated Contact Inserted** window (see [Figure 47](#)) appears after selecting the negated contact.
6. With the network highlighted, click the **Insert Coil** button (see [Figure 48](#) and [Figure 49](#)). The input and output are assigned to the PLC program. There are spaces above the negated contacts with three question marks (? ? ?) above these contacts. Enter **RapIDQSG.RapidInput.0** in the left ? ? ? spot and **RapIDQSG.RapidOutput.0** in the right ? ? ? spot to assign the input and output variables to the negated contacts. The screen then resembles what is shown in [Figure 50](#).
7. Within the **Solution Explorer** pane, navigate to **I/O < Device 5 (EtherCAT) < Box 1 (RapID Platform Network Interface)**, if it is not already expanded (see [Figure 51](#)).
8. Within the TwinCAT window, go to **File > Save All** to save the project.
9. Within the TwinCAT window, go to **Build > Build Solution** to build the solution.
10. After building the solution, within the **Solution Explorer** pane, navigate to **Solution 'EtherCAT Quickstart' (1 project) < EtherCAT Quickstart < I/O < Devices < Device 5 (EtherCAT) < Box 1 (RapID Platform Network Interface)** to expand **Module 1 (Digital IO)** and expand both **Digital Ins** and **Digital Outs** (see [Figure 52](#)).
11. Under **Digital Ins**, right-click **Digital Inputs** and select **Change Link**. A message similar to what is shown in [Figure 53](#) will appear. Select **RapIDQSG.RapidInput** and click **OK**.
12. Under **Digital Outs**, right-click **Digital Outputs** and select **Change Link**. The message similar to what is shown in [Figure 54](#) will appear. Select **RapIDQSG.RapIDOutput** and click **OK**.
13. Ensure that both **Digital Inputs** and **Digital Outputs** are linked to the **RapIDQSG.RapidInput** and **RapIDQSG.RapidOutput**, respectively, by reviewing the **ATTRIBUTES PANE** shown in [Figure 55](#).
14. Click the **Activate Configuration** button (see [Figure 56](#)) and the **Activate Configuration Warning** message appears (see [Figure 57](#)).
15. Click **OK** and the **Restart TwinCAT System in Run Mode** message appears (see [Figure 58](#)).
16. Click **OK** and then click the green rectangular bracket to log in to the PLC (see [Figure 59](#)).
17. If the PLC program does not automatically start, select the green **Play** button arrow (see [Figure 60](#)) to start the PLC program. A screen similar to [Figure 61](#) then appears.
18. The highlighted red icon, shown in [Figure 61](#), indicates that the user is logged into the PLC and that the PLC example application is running.
19. The logic elements on the ladder logic screen are now flashing because the input and output data is sampled as shown in [Figure 62](#). In the [Network Interface Example Application Suite](#) window, the LSB of the digital output data toggles quickly between 0 and 1 as shown in [Figure 63](#). The toggling of the digital output data confirms that this data is flowing. The PLC example application creation is now complete.

RPG2 ETHERCAT QUICKSTART GUIDE

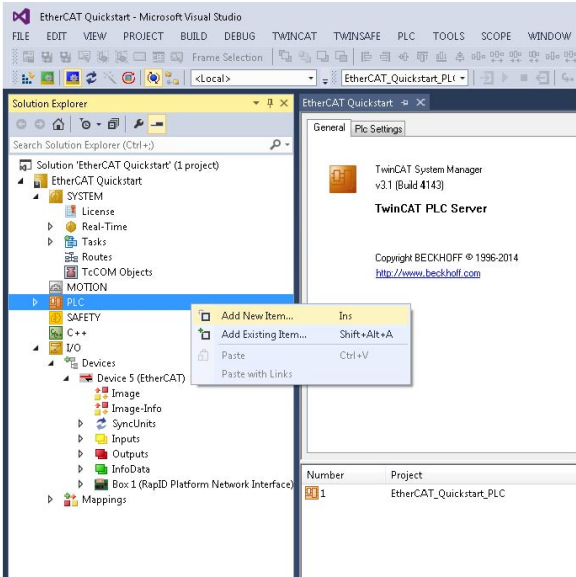


Figure 35. Add New Item, PLC

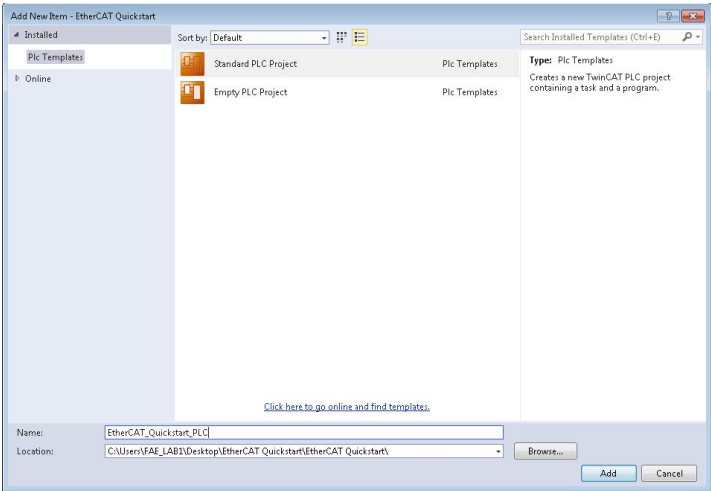


Figure 36. PLC Project Creation

RPG2 ETHERCAT QUICKSTART GUIDE

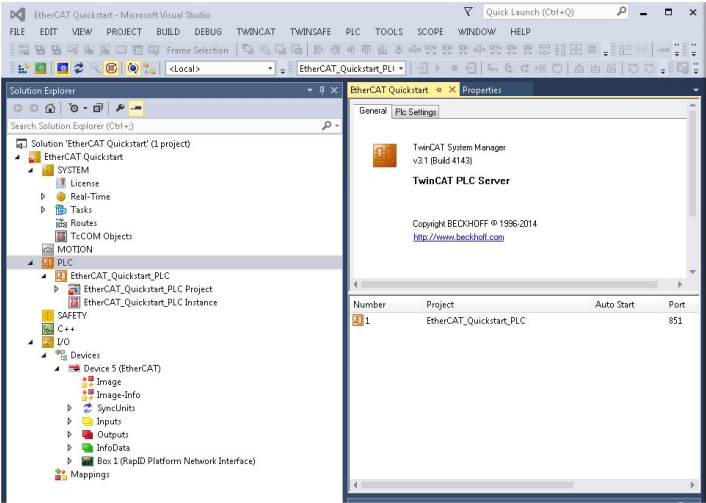


Figure 37. PLC Project Home

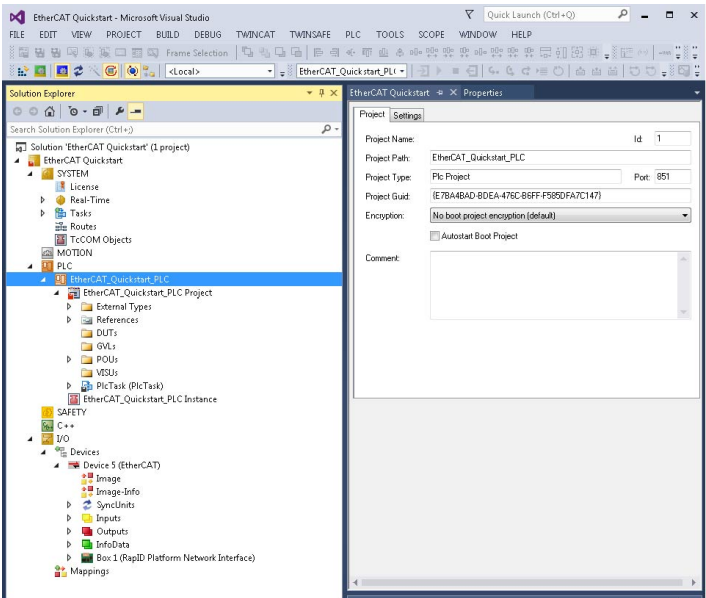


Figure 38. PLC Project Expanded View

RPG2 ETHERCAT QUICKSTART GUIDE

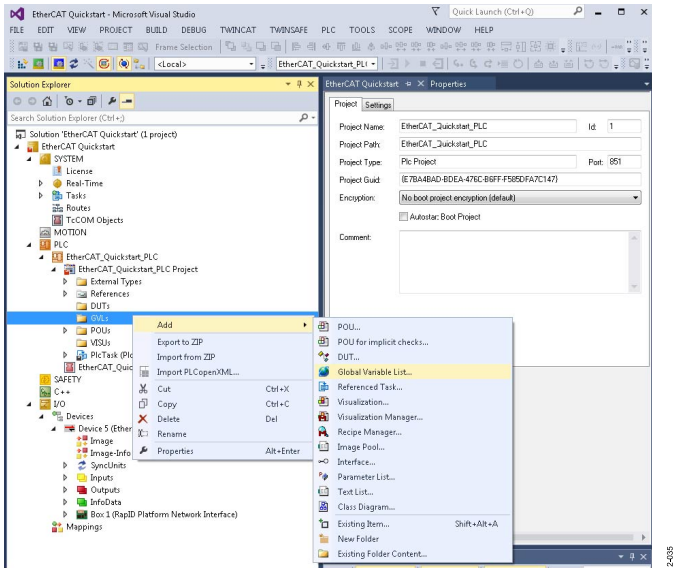


Figure 39. Add GVLs



Figure 40. Global Variable List

RPG2 ETHERCAT QUICKSTART GUIDE

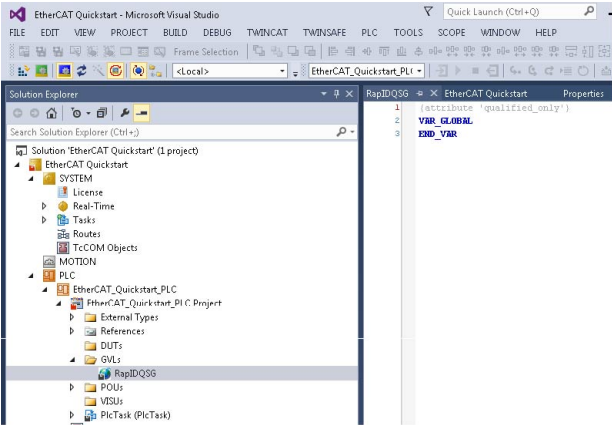


Figure 41. Global Variable View

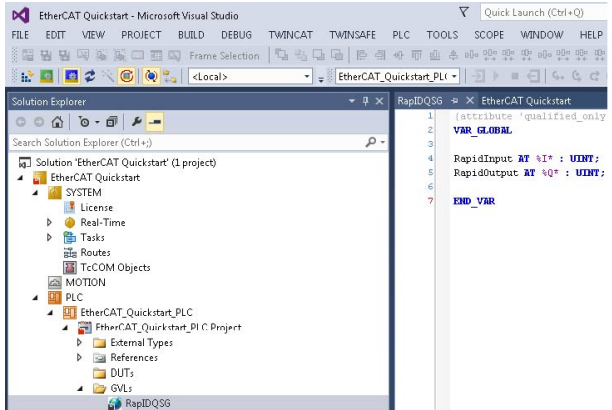


Figure 42. Global Variable Declarations

RPG2 ETHERCAT QUICKSTART GUIDE

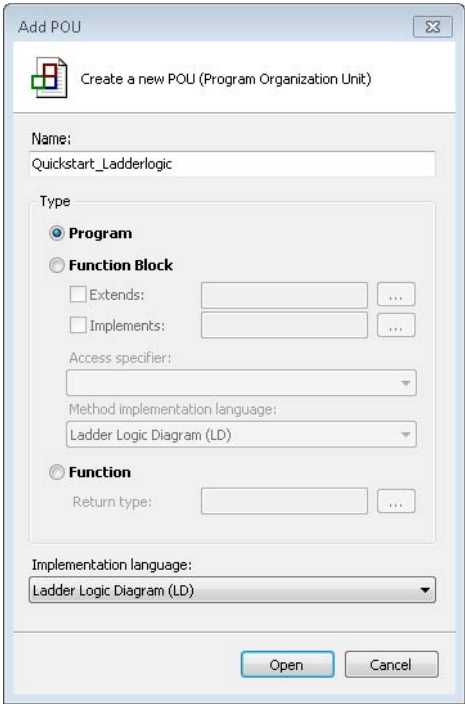


Figure 43. Add Program Organization Unit (POU)

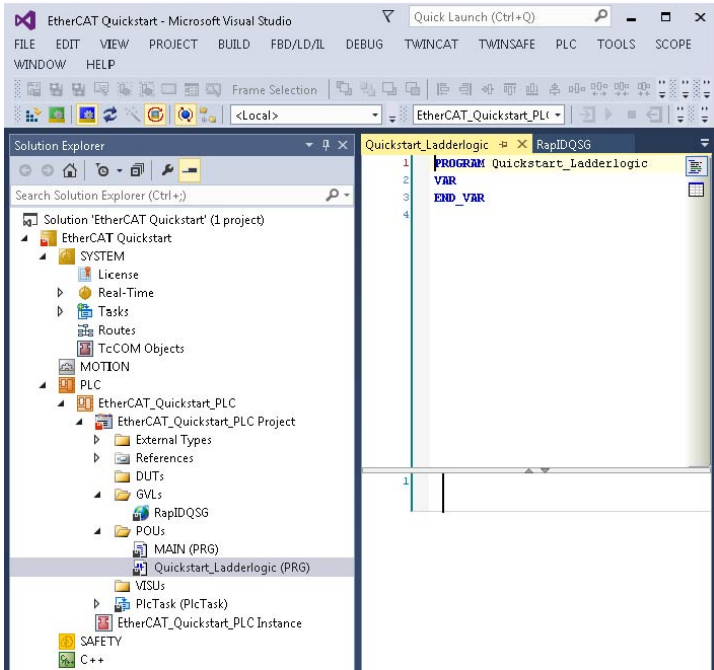


Figure 44. Ladder Logic File Part 1

RPG2 ETHERCAT QUICKSTART GUIDE

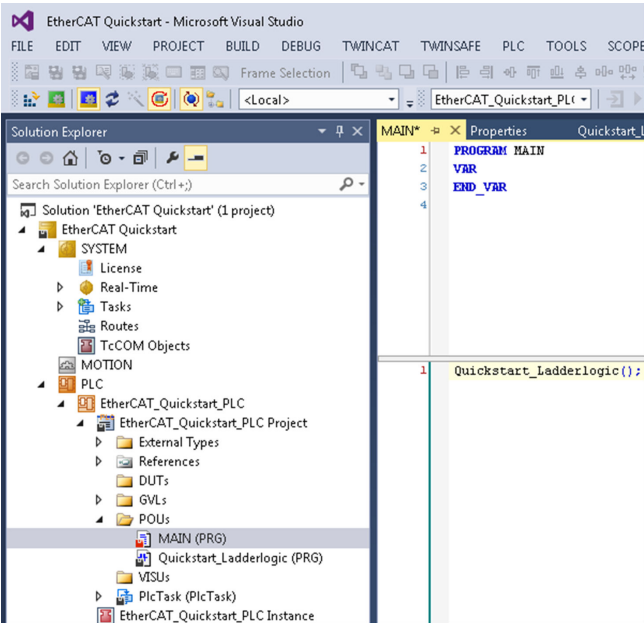


Figure 45. PLC Project Main

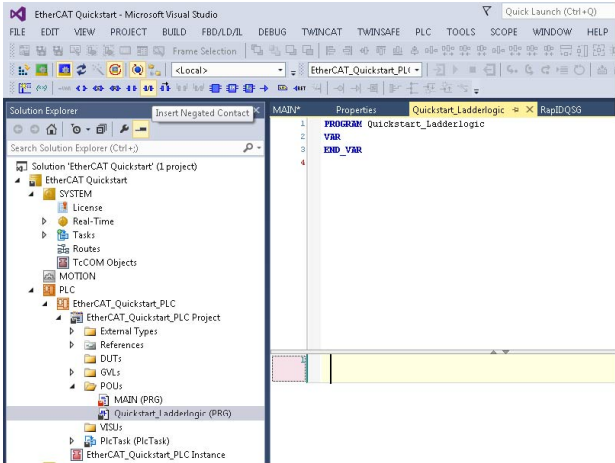


Figure 46. Ladder Logic File Part 2

RPG2 ETHERCAT QUICKSTART GUIDE

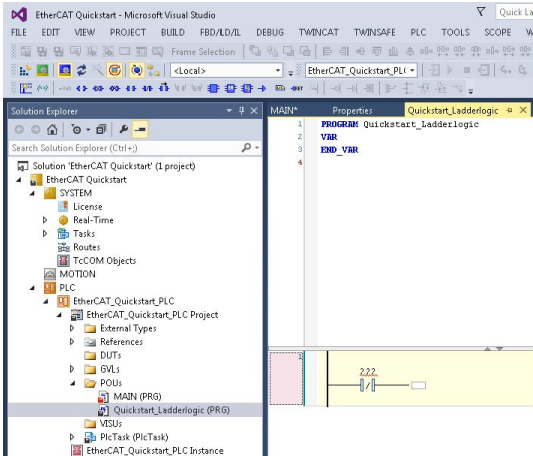


Figure 47. Negated Contact Inserted

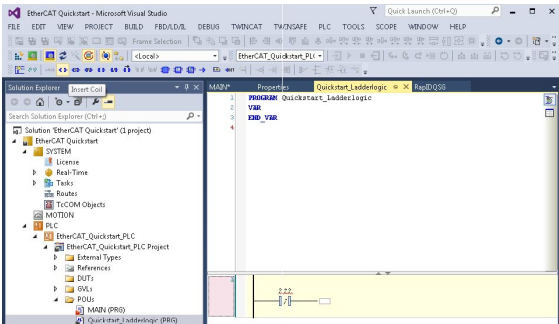


Figure 48. Insert Coil

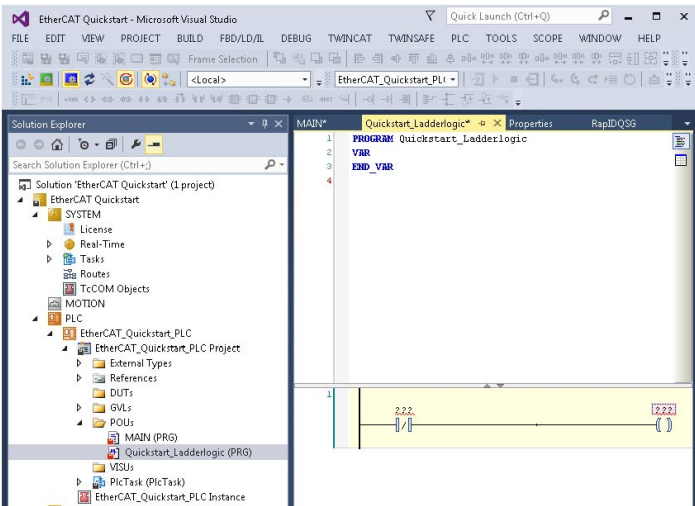


Figure 49. Output Energize Inserted

RPG2 ETHERCAT QUICKSTART GUIDE

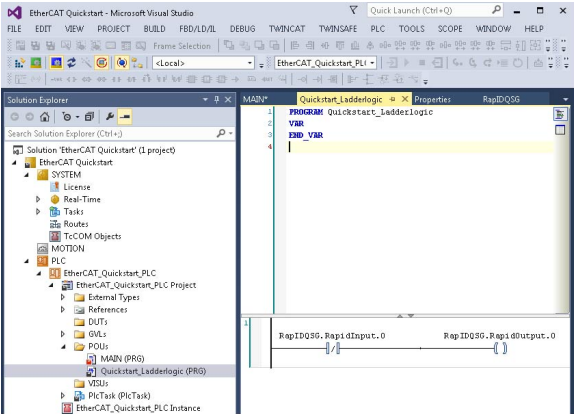


Figure 50. Input and Output Mapping

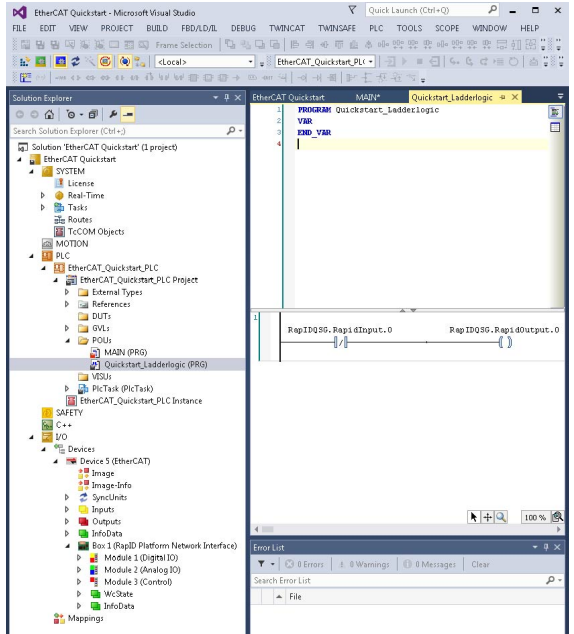


Figure 51. RapID Input and Output

RPG2 ETHERCAT QUICKSTART GUIDE

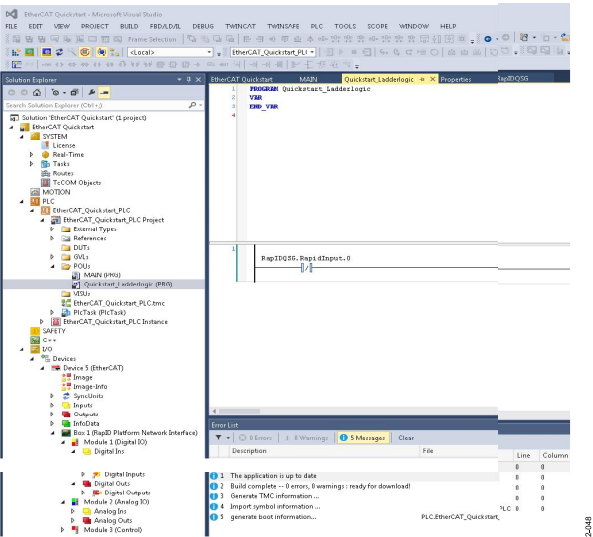


Figure 52. Build Complete and Rapid Input and Output Footprint

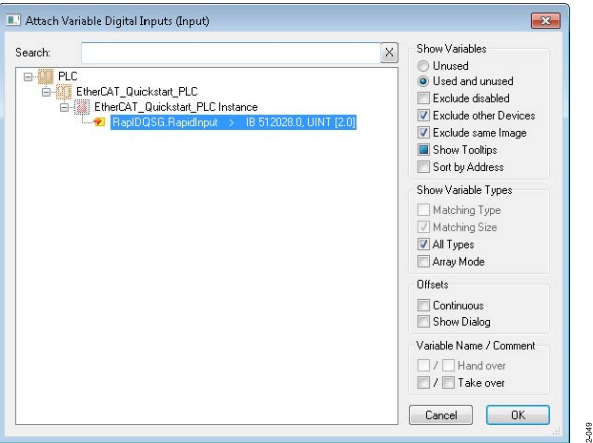


Figure 53. Rapid Digital Inputs

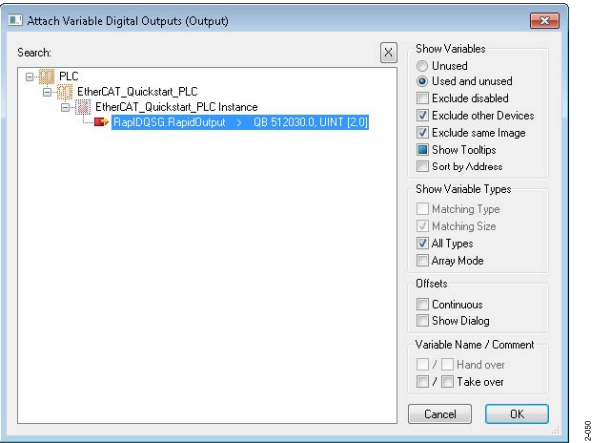


Figure 54. Rapid Digital Outputs

RPG2 ETHERCAT QUICKSTART GUIDE

ATTRIBUTES PANE

Name	Online	Type	Size	Address	In/Out	User ID	Linked to
Digital Inputs	X	0x1 (Q)	BITARR16	2.0	38.0	Input	RapIDQSG.RapidInput - PLC Task Inputs - EtherCAT_Quickstart_PLC Instance - EtherCAT_Quickstart_PLC
Analog In Ch1		UNINT	7.0	41.0	Input		
Analog In Ch2		UNINT	2.0	43.0	Input		
WzData		BIT	8.0	1512.1	Input		
InputFraggle		BIT	8.0	1524.1	Input		
State		UNINT	2.0	1540.0	Input		
AddrAddr		AMASAD...	8.0	1550.0	Input		
DiOutputShift		UNINT	4.0	1550.0	Input		
DiOutputShift		UNINT	4.0	1562.0	Input		
Digital Outputs	X	0x1 (Q)	BITARR16	2.0	38.0	Output	RapIDQSG.RapidOutput - PLC Task Outputs - EtherCAT_Quickstart_PLC Instance - EtherCAT_Quickstart_PLC
Analog Out Ch1		UNINT	2.0	43.0	Output		
Analog Out Ch2		UNINT	2.0	43.0	Output		
Control Data		UNINT	2.0	45.0	Output		

Figure 55. Linking Digital Input and Output Data

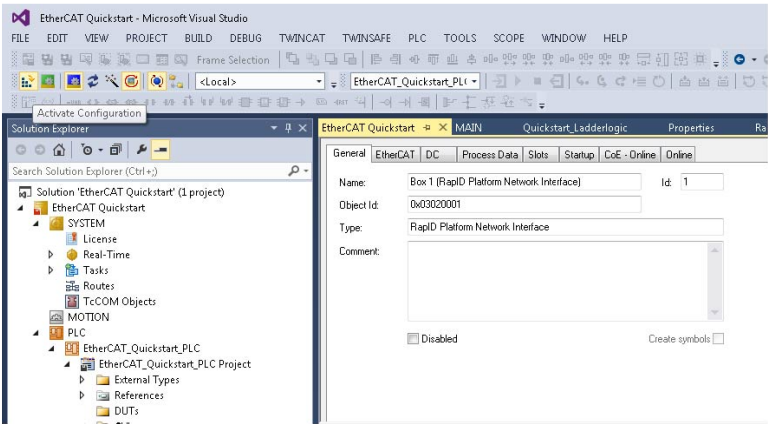


Figure 56. Activate Configuration

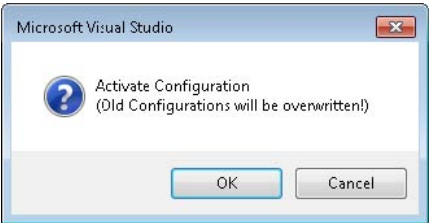


Figure 57. Activate Configuration Warning



Figure 58. Restart in Run Mode

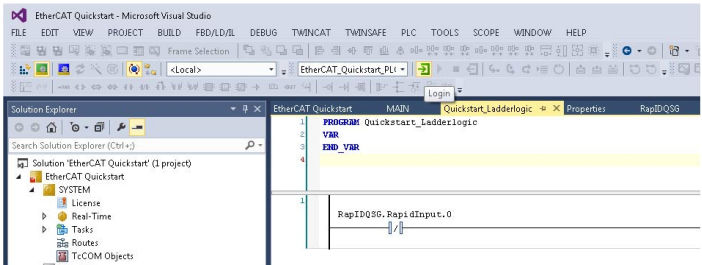


Figure 59. Logging Into the PLC

RPG2 ETHERCAT QUICKSTART GUIDE

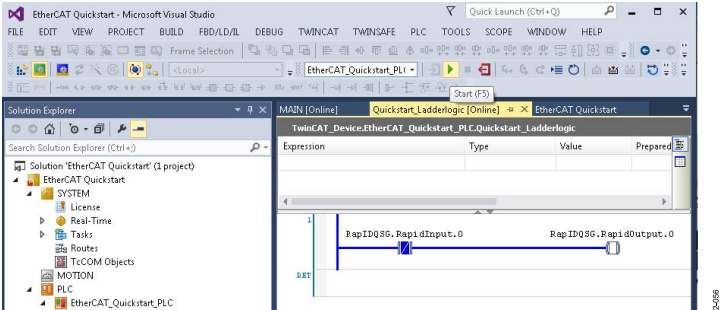


Figure 60. Start the PLC Program (Play Button)

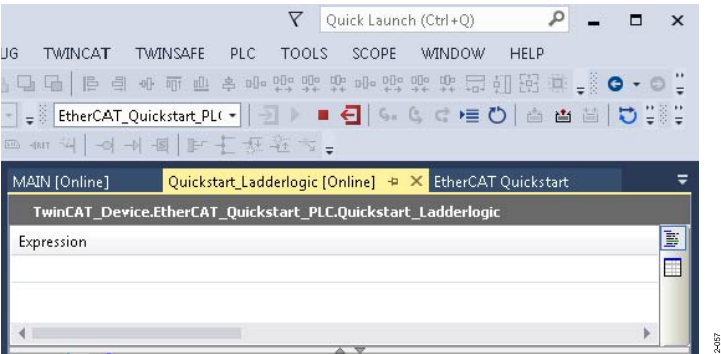


Figure 61. Logged into the PLC Program

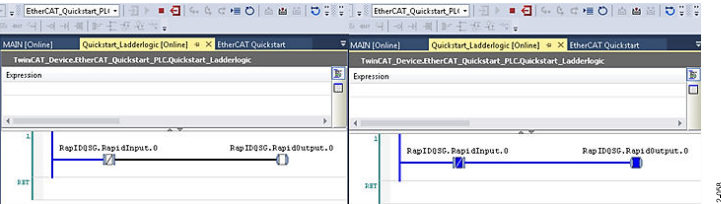


Figure 62. TwinCAT Ladderlogic Input and Output Toggling Between State 0 (Left) and State 1 (Right)

```
C:\WINDOWS\system32\cmd.exe
.\..\..\src\VM1_main.c:90 INFO: Performing system startup...
.\..\..\src\VM1_main.c:96 INFO: DONE
.\..\..\src\VM1_main.c:100 INFO: Start time update process...
.\..\..\src\VM1_main.c:106 INFO: DONE
.\..\..\src\VM1_main.c:110 INFO: Starting up example application...
.\..\..\src\VM1_main.c:137 INFO: DONE
.\..\..\src\VM1_main.c:140 INFO: Link init...
.\..\..\src\VM1_main.c:146 INFO: DONE
.\..\..\src\VM1_main.c:150 INFO: Unified Interface stack init...
.\..\..\src\VM1_main.c:159 INFO: DONE
.\..\..\src\VM1_main.c:164 INFO: Set response timeout...
.\..\..\src\VM1_main.c:170 INFO: DONE
.\..\..\src\VM1_main.c:175 INFO: Find Network Interface...
.\..\..\src\VM1_main.c:186 INFO: DONE
.\..\..\src\VM1_main.c:190 INFO: Set transmit modes...
.\..\..\src\VM1_main.c:204 INFO: DONE
.\..\..\src\VM1_main.c:208 INFO: Get installed protocol...
.\..\..\src\VM1_main.c:220 INFO: DONE (ethernet)
.\..\..\src\VM1_main.c:231 INFO: Set device 400...
.\..\..\src\VM1_main.c:292 INFO: DONE
.\..\..\src\VM1_main.c:296 INFO: Add item 500 to location 1...
.\..\..\src\VM1_main.c:307 INFO: DONE
.\..\..\src\VM1_main.c:311 INFO: Add item 501 to location 2...
.\..\..\src\VM1_main.c:322 INFO: DONE
.\..\..\src\VM1_main.c:326 INFO: Add item 502 to location 3...
.\..\..\src\VM1_main.c:337 INFO: DONE
.\..\..\src\VM1_main.c:342 INFO: Finalize configuration...
.\..\..\src\VM1_main.c:353 INFO: DONE
Current output data: Item A: 0x0000, Item B: 0x00000000, Item C: 0x0000[.\..\..\src\VM1_main.c:390] INFO:

C:\WINDOWS\system32\cmd.exe
.\..\..\src\VM1_main.c:90 INFO: Performing system startup...
.\..\..\src\VM1_main.c:96 INFO: DONE
.\..\..\src\VM1_main.c:100 INFO: Start time update process...
.\..\..\src\VM1_main.c:106 INFO: DONE
.\..\..\src\VM1_main.c:110 INFO: Starting up example application...
.\..\..\src\VM1_main.c:137 INFO: DONE
.\..\..\src\VM1_main.c:140 INFO: Link init...
.\..\..\src\VM1_main.c:146 INFO: DONE
.\..\..\src\VM1_main.c:150 INFO: Unified Interface stack init...
.\..\..\src\VM1_main.c:159 INFO: DONE
.\..\..\src\VM1_main.c:164 INFO: Set response timeout...
.\..\..\src\VM1_main.c:170 INFO: DONE
.\..\..\src\VM1_main.c:175 INFO: Find Network Interface...
.\..\..\src\VM1_main.c:186 INFO: DONE
.\..\..\src\VM1_main.c:190 INFO: Set transmit modes...
.\..\..\src\VM1_main.c:204 INFO: DONE
.\..\..\src\VM1_main.c:208 INFO: Get installed protocol...
.\..\..\src\VM1_main.c:220 INFO: DONE (ethernet)
.\..\..\src\VM1_main.c:231 INFO: Set device 400...
.\..\..\src\VM1_main.c:292 INFO: DONE
.\..\..\src\VM1_main.c:296 INFO: Add item 500 to location 1...
.\..\..\src\VM1_main.c:307 INFO: DONE
.\..\..\src\VM1_main.c:311 INFO: Add item 501 to location 2...
.\..\..\src\VM1_main.c:322 INFO: DONE
.\..\..\src\VM1_main.c:326 INFO: Add item 502 to location 3...
.\..\..\src\VM1_main.c:337 INFO: DONE
.\..\..\src\VM1_main.c:342 INFO: Finalize configuration...
.\..\..\src\VM1_main.c:353 INFO: DONE
Current output data: Item A: 0x0000, Item B: 0x00000000, Item C: 0x0000[.\..\..\src\VM1_main.c:390] INFO:
```

NEXT STEP: THE DESIGN PHASE

- ▶ The [RPG2 Hardware Design Integration Guide](#) section has details of how to interface with and integrate the required hardware.
- ▶ The [RPG2 Unified Interface User Guide](#) section has details on how to develop, customize, and integrate the software.
- ▶ The [RPG2 Programming User Guide](#) section has information about reprogramming or loading binary files onto the board.
- ▶ The [RPG2 I/O Configuration Tool User Guide](#) section has information about how to create a customized input and output footprint for the system.
- ▶ To evaluate another protocol, continue to the quickstart guide of that protocol.

analog.com

RPG2 PROFINET QUICKSTART GUIDE

FEATURES

- ▶ Cyclic input data transfer up to 1440 bytes
- ▶ Cyclic output data transfer up to 1440 bytes
- ▶ Cycle time down to 1 ms (Ethernet application processor interface with an input data of 6 bytes and an output data of 8 bytes)
- ▶ Media redundancy protocol
- ▶ Support for diagnostics (Siemens and manufacturer specific)

EQUIPMENT NEEDED

- ▶ Siemens PLC
- ▶ EV-RPG2-PNZ evaluation board
- ▶ 1 PC, using 2 PCs may be easier

SOFTWARE NEEDED

- ▶ [Network interface example application suite \(ni-example-app.exe\)](#)
- ▶ Siemens SIMATIC TIA portal
- ▶ Windows WinPcap
- ▶ [Analog Devices, Inc., general station description markup language \(GSDML\) file](#)

GENERAL DESCRIPTION

The RapID Platform Generation 2 (RPG2) module is a pretested industrial network interface designed to manage industrial protocols and network traffic. It supports PROFINET®, PROFINET isochronous real-time (IRT), Ethernet/IP®, Ethernet/IP with device level ring (DLR), EtherCAT®, and Modbus/TCP. This RPG2 module uses a Unified Interface to communicate with different protocols.

This Unified Interface is a custom protocol by Analog Devices that allows interaction between an application processor and the RPG2 module. The Unified Interface is agnostic of the industrial protocol.

The Unified Interface ensures that the application processor hardware and software interface does not need to change when switching or updating protocols. The RPG2 module connects to an application processor via a universal asynchronous receiver transmitter (UART), Ethernet, or serial peripheral interface (SPI).

The EV-RPG2-PNZ evaluation kit provides end to end evaluation of the communication path from the application processor to the programmable logic controller (PLC) over the Industrial Ethernet interface (using the [network interface example application suite \(ni-example-app.exe\)](#)). This user guide describes how to use the kit to set up and run a PLC example application.

For the example described in this quickstart guide, the application processor is a PC, and the PC communicates to the RPG2 module via an Ethernet network interface card (NIC).

EVALUATION KIT SETUP FOR PROFINET

Refer to the [RPG2 RapID Platform Generation 2 User Guide](#) section to set up the hardware. Note that the default link type is Ethernet. When a change of link type to UART is required, refer to the [Link Configuration File](#) section of this document. See [Figure 64](#) and [Figure 65](#) for the setup for running the PROFINET application example with the default link type for one PC or two PCs, respectively.

Use one PC if having both the total integrated automation (TIA) portal application and the [network interface example application suite \(ni-example-app.exe\)](#) on one PC is acceptable. Use two PCs if there is a requirement to clearly distinguish between the two applications.

For the rest of this user guide, the two PC setup is used for setting up the TIA portal PLC application and for running the [network interface example application suite \(ni-example-app.exe\)](#).

When LED A6/A7 is green (see [Figure 64](#) and [Figure 65](#)), this color indicates that the board is preloaded with PROFINET, and the startup process is complete.

RPG2 PROFINET QUICKSTART GUIDE

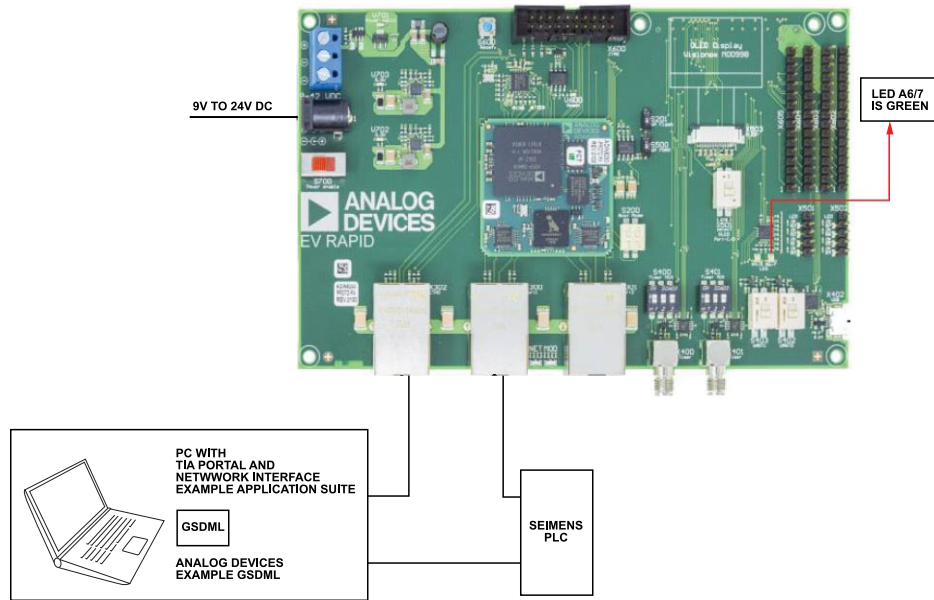


Figure 64. Setup for Running the PROFINET Application Example with the Default Link Type and One PC

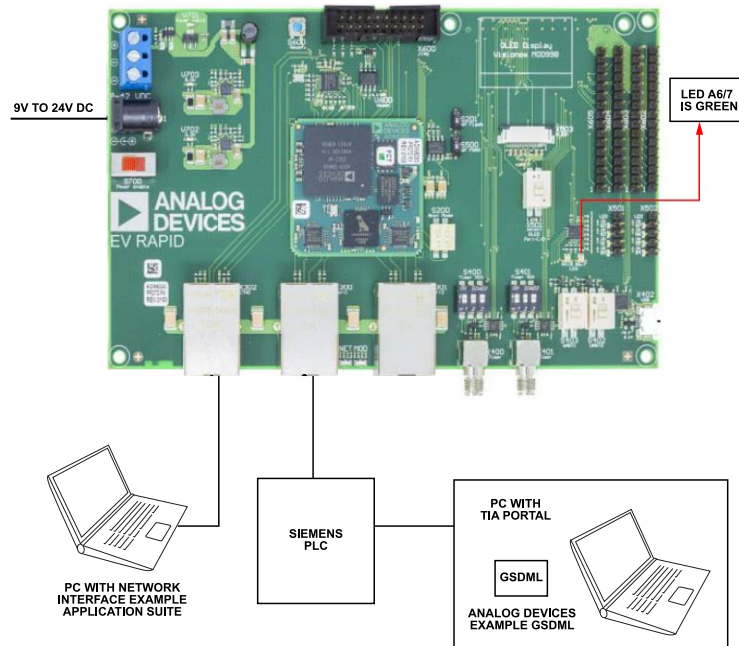


Figure 65. Setup for Running the PROFINET Application Example with the Default Link Type and Two PCs

NETWORK INTERFACE APPLICATION SUITE WITH A PC RUNNING THE SIEMENS TIA PORTAL

Communication between the application processor and the RPG2 module on the baseboard is enabled using an Ethernet RJ-45 connector on the baseboard. In addition to the evaluation kit, the following items (at a minimum) are required:

- ▶ A PC running the [ADIN2299](#) host processor simulator (**ni-example-app**).
- ▶ A PC running the Siemens TIA portal
- ▶ A Siemens PLC

Note that this user guide is not a general introduction into the TIA portal or the configuration and programming of Siemens PLCs. It is assumed that the user is comfortable using relevant Siemens SIMATIC technologies.

RPG2 PROFINET QUICKSTART GUIDE

Application Processor Simulator Software SetUp

Obtain the network interface example application suite by taking the following steps:

1. Go to the ADIN2299 main product page at www.analog.com/adin2299.
2. Select the [ADIN2299 ni-example-application](#) tab.
3. Review the **Terms and Conditions** and accept.
4. Extract the contents of the .zip file. In the extracted folder is the **ni-example-app.exe** file that must run from the command line of your PC.
5. To see the available Ethernet devices from the command line prompt, enter **ni-example-app.exe -l ETH -n** and a list of available Ethernet NICs will display. Running this command gives the network device numbers in the following format: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX. Use this list along with the **Network and Sharing Center** name in Windows® to determine which of the listed numbers to use. An example command is shown on [Figure 66](#).
6. Once this number is known, enter the following command: **ni-example-app.exe -l ETH -n XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX** and it will run the proper initialization and application processor functionality (see [Figure 67](#)).

Note that if you see **FATAL Bad Memory Block** while running the network example application suite, ensure that you have the most updated version of Windows WinPcap on your PC. The NIC identifier begins with **DeviceNPF_**.

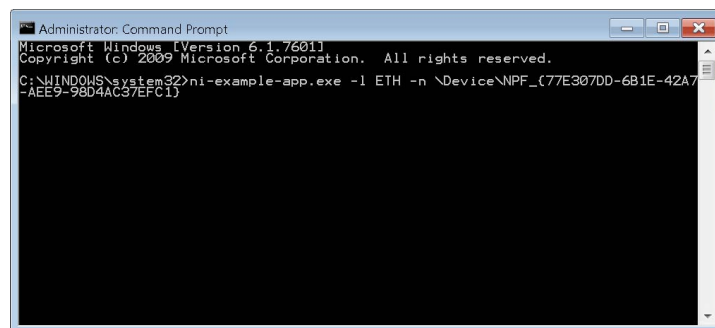
See [Table 2](#) for the RPG2 input and output modules for the PROFINET sample configuration.

```

..\..\src\NI_main.c:75] INFO: Welcome to ni-example-app!
..\..\src\NI_main.c:78] INFO: Processing arguments...
..\..\src\NI_main.c:86] INFO: DONE
..\..\src\NI_main.c:90] INFO: Performing system startup...
..\..\src\NI_main.c:96] INFO: DONE
..\..\src\NI_main.c:109] INFO: Start time update process...
..\..\src\NI_main.c:106] INFO: DONE
..\..\src\NI_main.c:110] INFO: Starting up example application...
..\..\src\NI_main.c:137] INFO: DONE
..\..\src\NI_main.c:140] INFO: Link init...
..\..\src\NI_main.c:146] INFO: DONE
..\..\src\NI_main.c:150] INFO: Unified Interface stack init...
..\..\src\NI_main.c:159] INFO: DONE
..\..\src\NI_main.c:164] INFO: Set response timeout...
..\..\src\NI_main.c:170] INFO: DONE
..\..\src\NI_main.c:175] INFO: Find Network Interface...
..\..\src\NI_main.c:180] INFO: DONE
..\..\src\NI_main.c:190] INFO: Set transmit modes...
..\..\src\NI_main.c:204] INFO: DONE
..\..\src\NI_main.c:208] INFO: Get installed protocol...
..\..\src\NI_main.c:220] INFO: DONE (PROFINET)
..\..\src\NI_main.c:281] INFO: Set device 400...
..\..\src\NI_main.c:292] INFO: DONE
..\..\src\NI_main.c:296] INFO: Add item 500 to location 1...
..\..\src\NI_main.c:307] INFO: DONE
..\..\src\NI_main.c:311] INFO: Add item 501 to location 2...
..\..\src\NI_main.c:322] INFO: DONE
..\..\src\NI_main.c:326] INFO: Add item 502 to location 3...
..\..\src\NI_main.c:337] INFO: DONE
..\..\src\NI_main.c:342] INFO: Finalize configuration...
..\..\src\NI_main.c:353] INFO: DONE
Current output data: item A: 0x0000, item B: 0x00000000, item C: 0x0000[..\..\src\NI_main.c:398] INFO:
..\..\src\NI_main.c:398] INFO:

```

Figure 66. Network Interface Example Application Suite Command Log



```

Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32\ni-example-app.exe -l ETH -n \Device\NPF_{77E307DD-6B1E-42A7-AEE9-98D4AC37EFC1}

```

Figure 67. Network Interface Example Application Suite

Setting up the PROFINET Network With the Siemens TIA Portal

This user guide describes an example of how to use a PC running the Siemens TIA portal with a Siemens PLC. This PLC example application assumes the RPG2 module is configured as detailed in [Table 2](#). This configuration defines three items with different types of input and output (I/O). Item 500 defines 2 bytes of input and output data, Item 501 defines 4 bytes of analog input and analog output data, and Item 502

RPG2 PROFINET QUICKSTART GUIDE

represents 2 bytes of control data (digital data). For more information on these items, refer to the [RPG2 I/O Configuration Tool User Guide](#) section.

This section provides instructions for setting up and using the RPG2 module with the link type set to Ethernet. Note that the RPG2 module must connect to a PC running the network interface example application suite (which is the leader network program) and a Siemens PLC that is configured by another PC.

[Figure 68](#) shows the setup used by the following sections. See the [Evaluation Kit Setup for PROFINET](#) section for details on how to set up the test network.

Table 2. RPG2 Input and Output Modules for the PROFINET Sample Configuration

Item Number	Item Type	Input Size (Bytes)	Output Size (Bytes)	Module ID	Submodule
500 (Digital Inputs and Outputs)	Cyclic	2	2	0x10400000	0x10440001
501 (Analog Inputs and Outputs)	Cyclic	4	4	0x10500000	0x10550001
502 (Control Register)	Cyclic	0	2	0x10600000	0x10660001

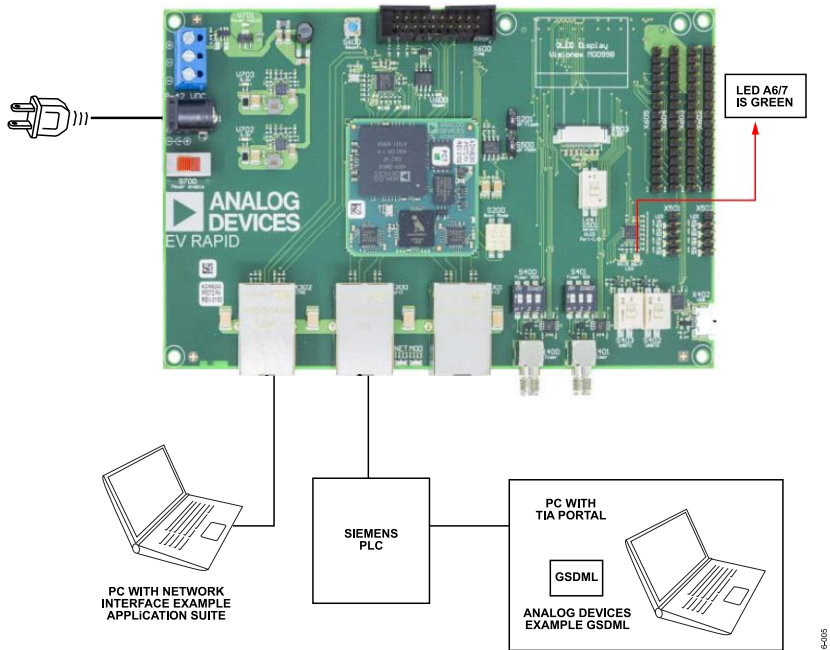


Figure 68. Sample Application Hardware Setup

Configuring the PLC Setup in the Siemens TIA Portal

Take the following steps to use the Siemens TIA portal to configure the PROFINET network:

1. Double-click to start the TIA portal to display the new project wizard window (see [Figure 69](#)).
2. Click **Create New project** to display the naming the new project window (see [Figure 70](#)).
3. Rename the project as it makes sense for your application.
4. Click **Create** to open the open project setup view window (see [Figure 71](#)).
5. Click **Project view > Open the project view**. Note that loading this view can take a while (see [Figure 71](#)).
6. When loading finishes, the main test project window displays (see [Figure 72](#)).
7. Click **Add new device** and the window shown in [Figure 73](#) appears.
8. Select the Siemens PLC in use and click **OK** (see [Figure 74](#)) to display the PLC slot window (see [Figure 75](#)).
9. Select the **Network view** tab to display the PROFINET network with the PLC added (see [Figure 76](#)).

If a user is going to use the Siemens TIA portal to run this example configuration, it is assumed the user is familiar with how to use Siemens TIA portal.

RPG2 PROFINET QUICKSTART GUIDE

Note that this section varies from user to user because of the different PLCs a user may have. While the following example uses the Siemens TIA portal, there are other vendors who produce PROFINET leader software. Use this user guide to walk through the application. However, the PROFINET specific portions may differ depending on the software chosen.

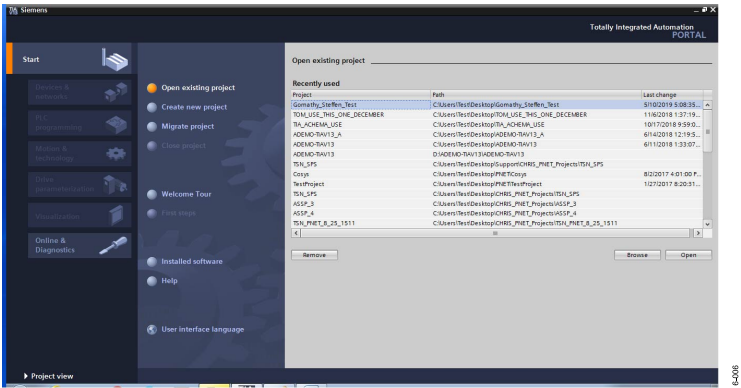


Figure 69. New Project Wizard Window

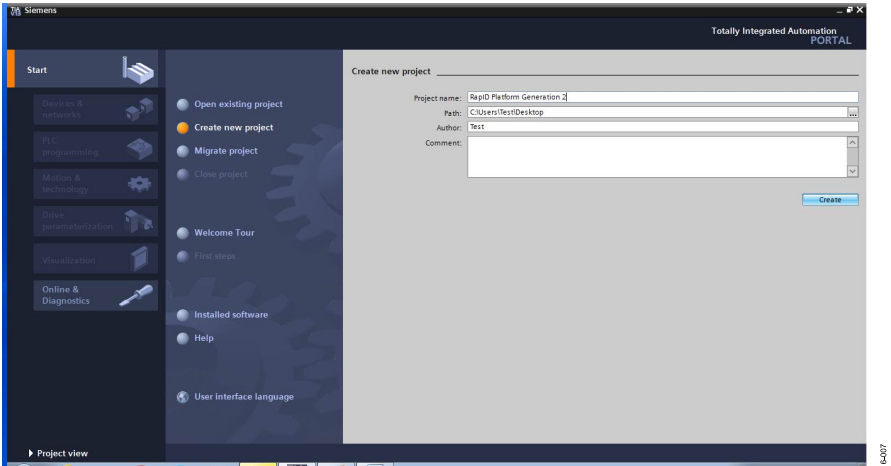


Figure 70. Naming New Project Window

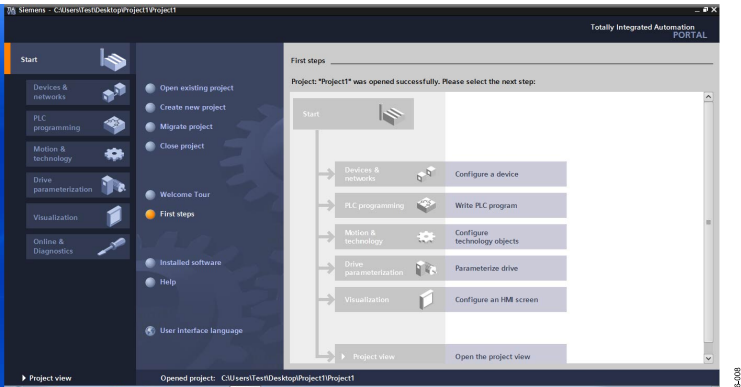


Figure 71. Project Setup View Window

RPG2 PROFINET QUICKSTART GUIDE

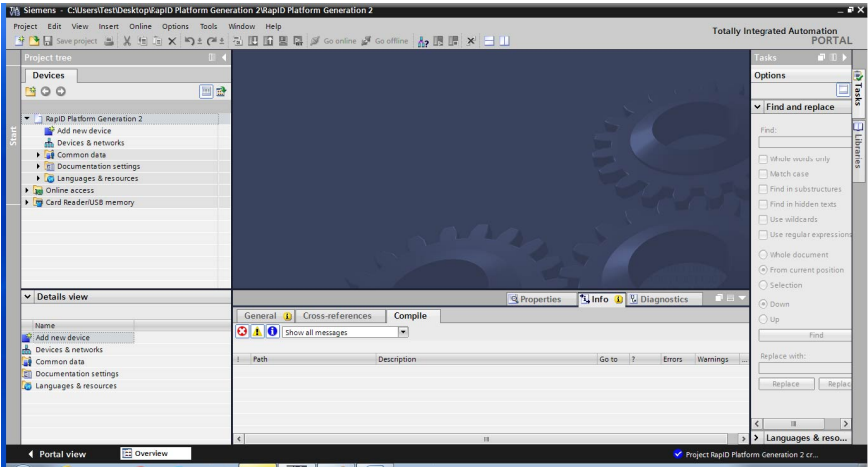


Figure 72. Main Test Project Window

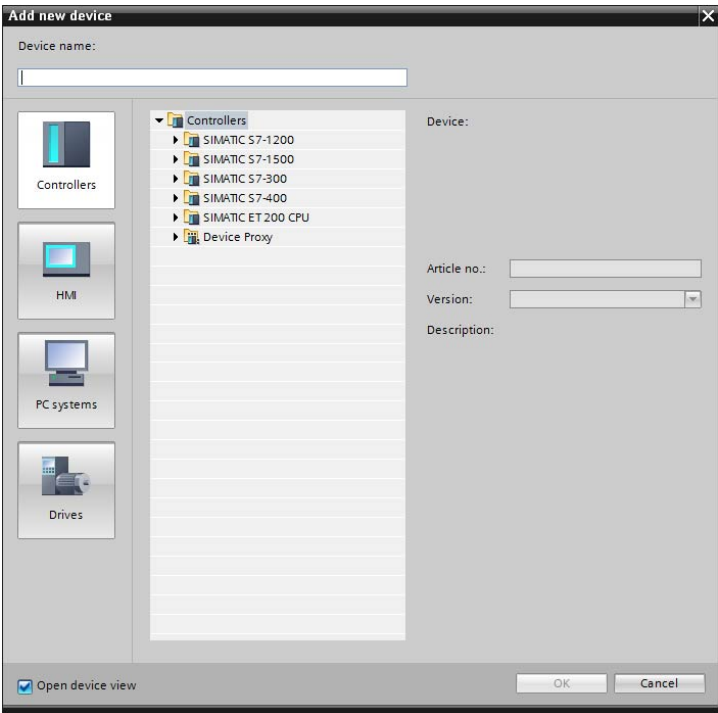


Figure 73. PLC Selection Window

RPG2 PROFINET QUICKSTART GUIDE

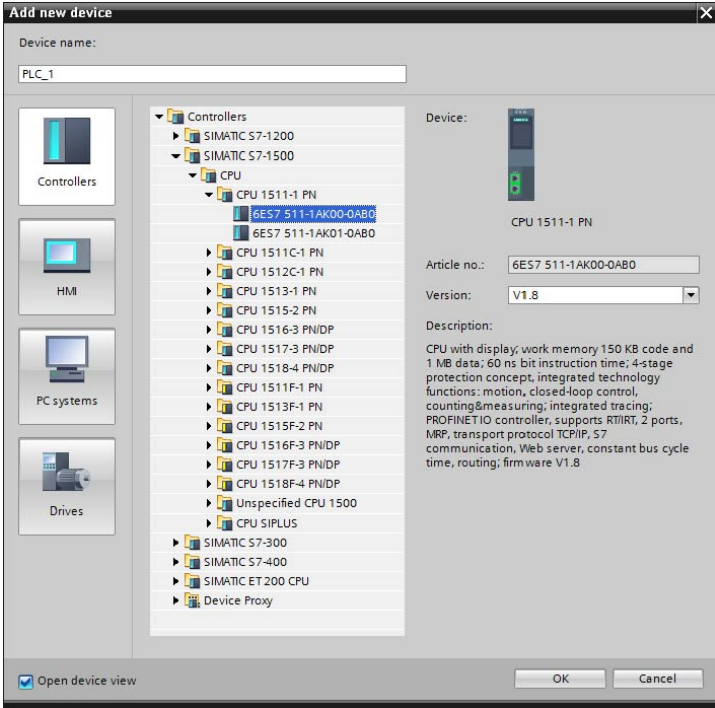


Figure 74. Selecting the PLC in Use

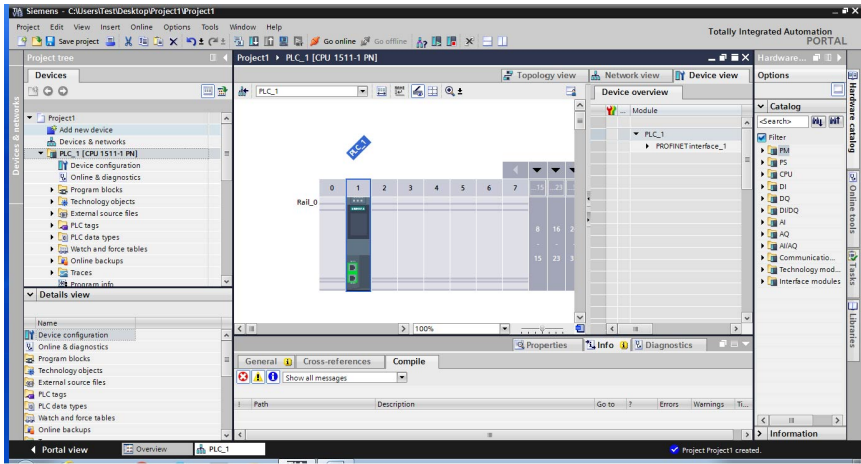


Figure 75. PLC Slot Window

RPG2 PROFINET QUICKSTART GUIDE

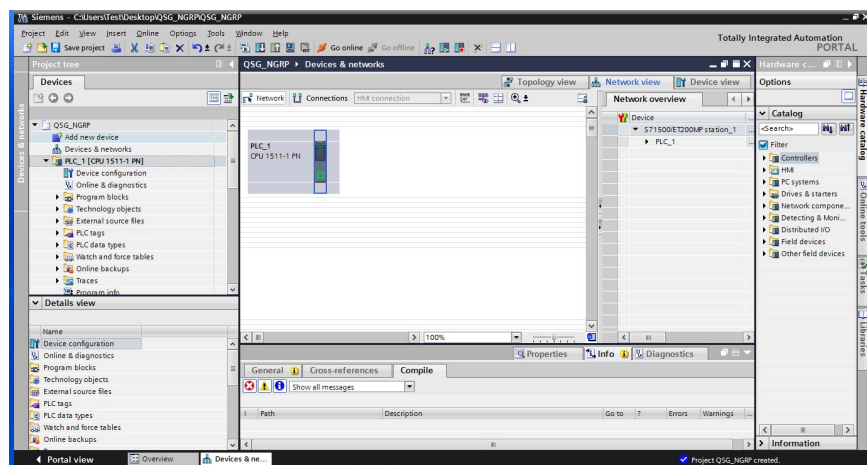


Figure 76. PROFINET Network with PLC Added

Installing the General Description File

Take the following steps to install the general description file:

1. Go to the **Options** dropdown menu (see Figure 77) and select **Manage general station description files (GSD)**.
2. The **Manage general station description files** window then displays (see Figure 78) your flash drive or hard drive directory location for the Rapid stock GSDML file (**GSDML-V2.34-Analog-Devices-Rapid-Example-20200226-020800.xml**). Note that the user must have the most up to date GSDML file from the Profinet Software at www.analog.com/adin2299.
3. Select the GSDML file box and click **Install** to install the GSDML file (see Figure 79).
4. Click **Close**. The **Updating the hardware catalog** window then displays (see Figure 80).

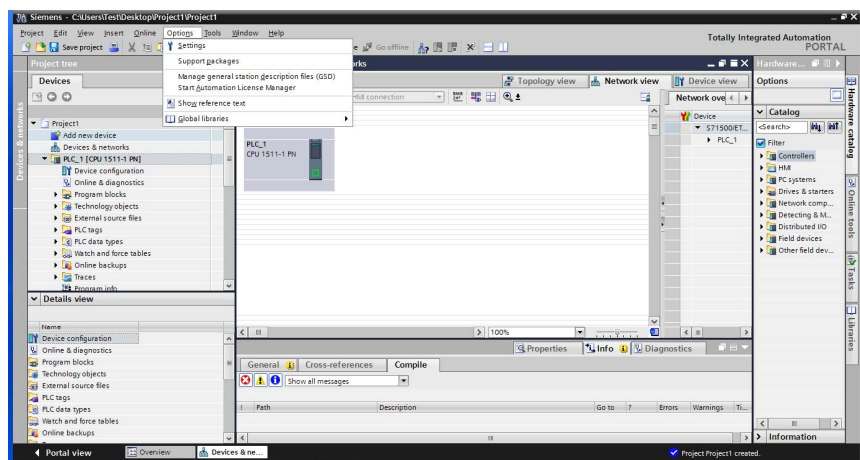


Figure 77. Main Test Project Window

RPG2 PROFINET QUICKSTART GUIDE

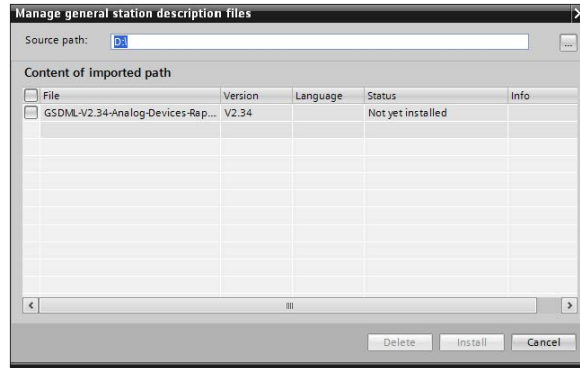


Figure 78. Add GSD File

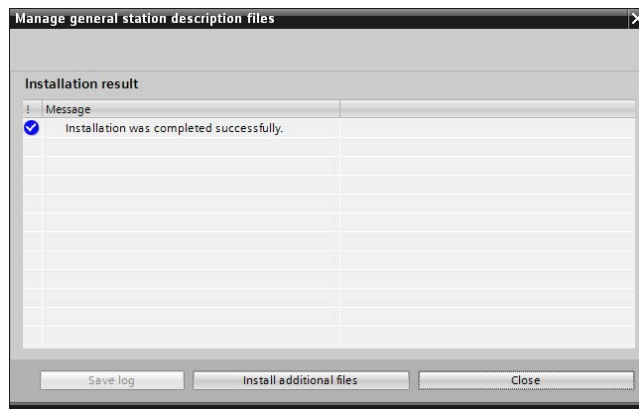


Figure 79. Successful Installation of the GSDML File

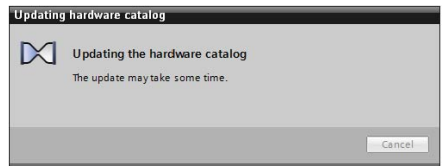


Figure 80. Updating the Hardware Catalog

Creating the PROFINET Network

Take the following steps to create the PROFINET network within the Siemens TIA portal:

1. Within the main test project window, click **Devices & networks** (see [Figure 81](#)).
2. Expand the **Hardware catalog** pane on the right-side of the window and select **PROFINET FIDO5000 REM** (see [Figure 81](#)).
3. Drag the **PROFINET FIDO5000 REM** device into the **Network** area under the **Network view** tab (see [Figure 82](#)).
4. Click the **Not assigned** link (see [Figure 82](#)) within the **Network view** tab to bring up a yellow pop-up with the text appearing in the same format as a URL link, **PLC_1.PROFINET_interface_1** (see [Figure 83](#)). Note that the actual text that is seen may differ depending on the PLC of the user.
5. Click this yellow pop-up to connect the network by using the green line (see [Figure 84](#)).

RPG2 PROFINET QUICKSTART GUIDE

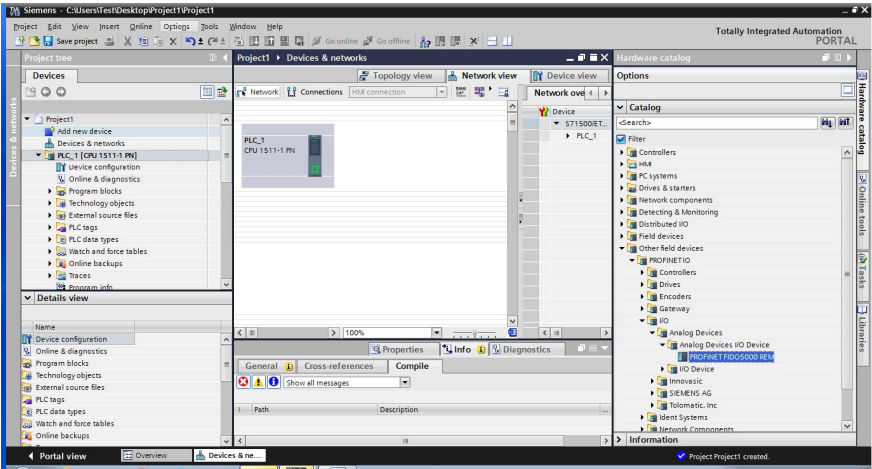


Figure 81. Main Test Project Window

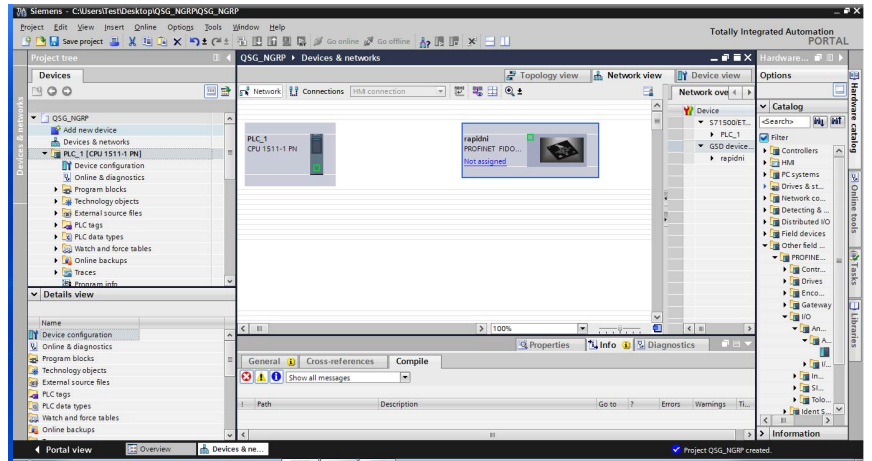


Figure 82. Rapid Device Installed onto the Network

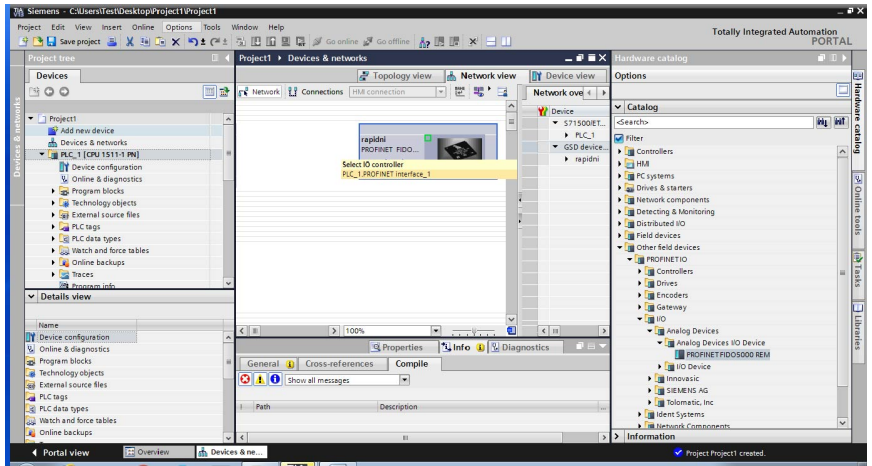


Figure 83. Assigning the Rapid Device to the PROFINET Network

RPG2 PROFINET QUICKSTART GUIDE

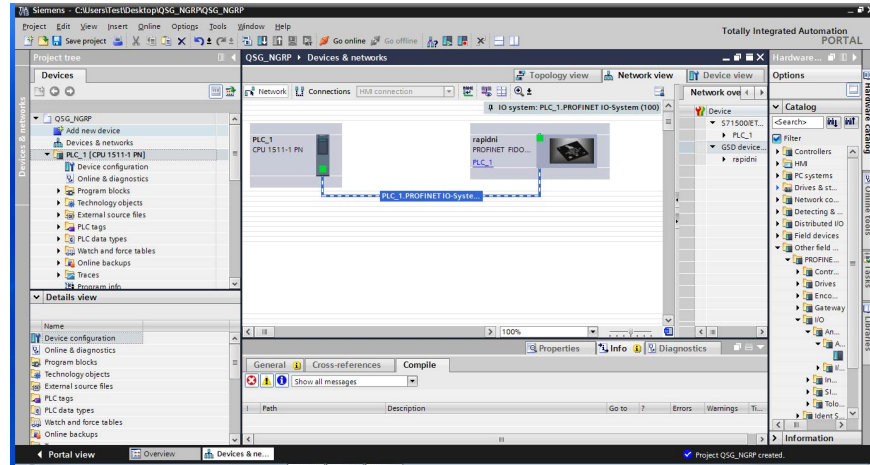


Figure 84. PROFINET Network Created

Getting Online With the PROFINET Network

Take the following steps to get online with the PROFINET network:

1. Double-click the **PLC_1** icon shown in Figure 84.
2. Select **PLC_1** and click **Download to Device** to select the network to download (see Figure 85).
3. Select the appropriate PLC in this window and click **Load** (see Figure 86).
4. In the **Load preview** window, click **Load** (see Figure 87).
5. Right-click the **PROFINET FIDO5000 REM** icon across from the PLC and click **Assign name** to display the **Assign PROFINET device name** window (see Figure 88).
6. Click the orange **Go online** button at the top of the window to go online with the PROFINET network (see Figure 89). Taking this step causes the outline of the screen to turn orange and green checkmarks to appear on all devices, which indicates that the system is actively communicating (see Figure 89). Communication between the PC of the user and the Siemens PLC is now good.
7. Click the blue **Go offline** button (see Figure 89) to complete this process.

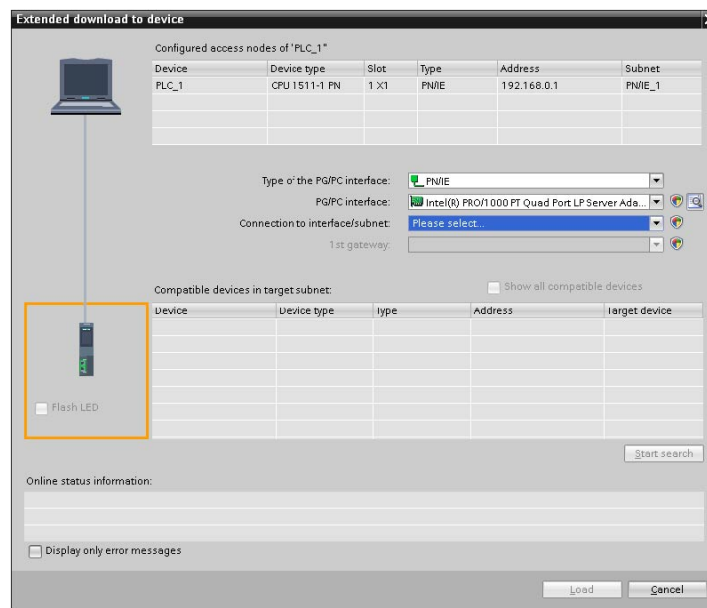


Figure 85. Network Download Selection

RPG2 PROFINET QUICKSTART GUIDE

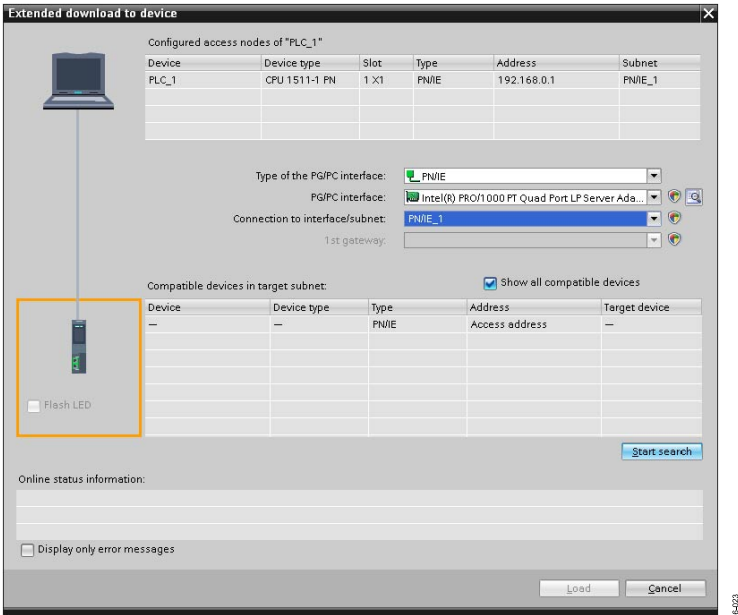


Figure 86. PLC Load Selection

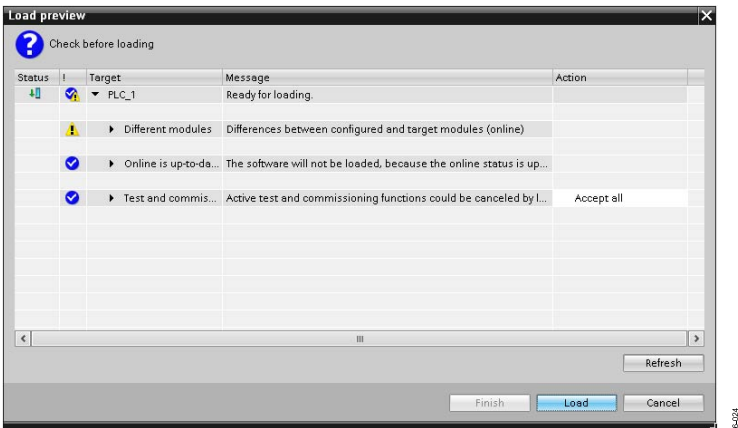


Figure 87. Load Preview Window

RPG2 PROFINET QUICKSTART GUIDE

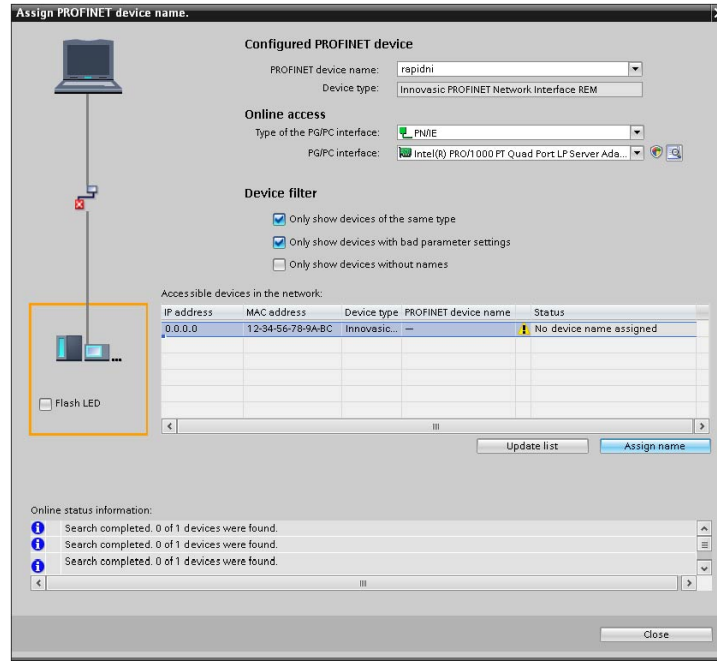


Figure 88. Main Test Project Name Assignment Window

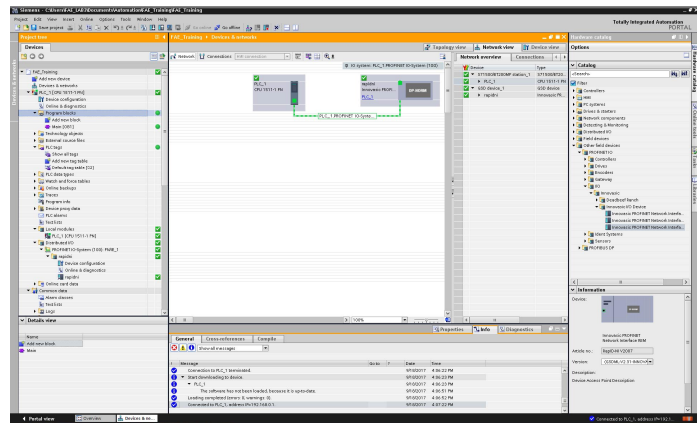


Figure 89. Main Test Project Window

Configuring the Optional PROFINET IRT

Take the following optional steps to activate IRT communication for your PROFINET network (make sure you use a PROFINET PLC that supports IRT communication):

1. In the **Devices & networks** window select the **Topology view** tab.
2. Draw a line from the connected PLC port (**PLC_1**) to the connected device port (**rapidni**) (see [Figure 90](#)).
3. Select the PLC (**PLC_1**) and open the **Properties** window.
4. In the **Properties** window, select **PROFINET interface [X1]**, **Advanced options**, **Real time settings**, and **Synchronization**, then set the **Synchronization role** dropdown menu to **Sync master** (see [Figure 91](#)).
5. Select the device (**rapidni**) and open the **Properties** window.
6. In the **Properties** window, select **PROFINET interface [X1]**, **Advanced options**, **Real time settings**, and **Synchronization**, then activate the **IRT radio** button (see [Figure 92](#)).
7. Check **Update time** of the PLC and device to ensure update times are identical.

RPG2 PROFINET QUICKSTART GUIDE

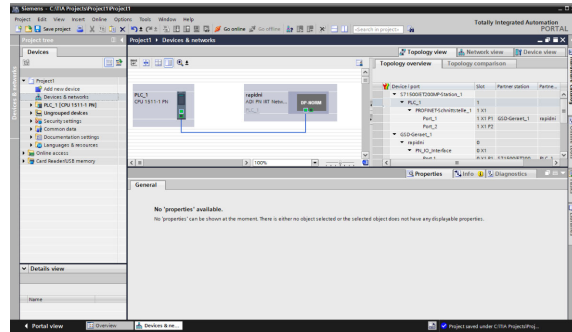


Figure 90. Topology View Window

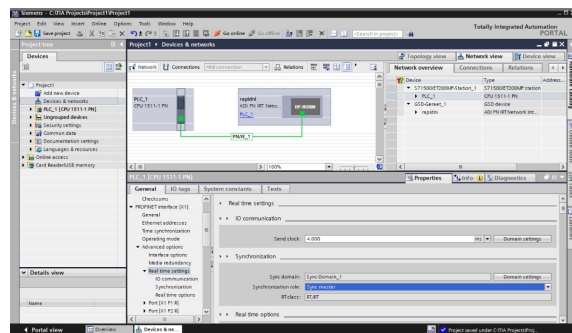


Figure 91. Synchronization Role PLC

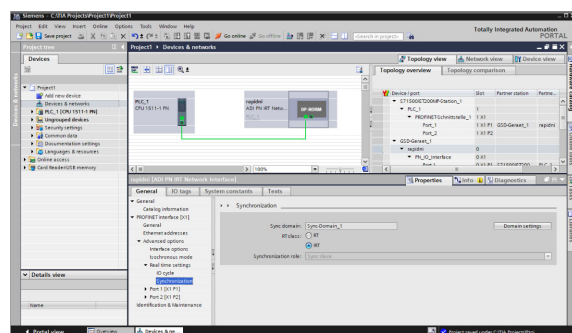


Figure 92. Synchronization Properties Device

Creating a Ladder Logic Application

Take the following steps to create a ladder logic application:

1. Within the **Network Overview** pane (see [Figure 84](#)), go to **GSD_device > rapidni** to display the **16-bit Digital I/O** module.
2. Drag the **16-bit Digital I/O** icon into the slot.
3. Go to **PLC tags < Show all tags** on the left-hand side of the **Project tree** window to display the **PLC tags** dropdown menu (see [Figure 93](#)).
4. Right-click to add two new tags and select **New Tag** for each new tag. The result of adding these tags brings up the screen shown in [Figure 94](#). For this example, the **Input_0** tag is **Address &I0.0** and the **Output_0** tag is **Address &Q0.0**. These correspond with the 0 index byte of the digital I/O item.
5. Within the **Project tree > Devices** pane (see [Figure 95](#)), navigate to **PLC-1[1511-1-PC] > Program blocks > Main [OB-1]** and type in the inputs and outputs over the rungs, for example, **Network 1** and **Network 2**.
6. Click the **Compile** button to build the program (see [Figure 96](#)). Once the program is built, it can be transferred to the PLC.
7. Click the **Download** button to transfer the program to the PLC (see [Figure 96](#)).

- Click **Go online** (see [Figure 96](#)). You will now see the ladders in the ladder logic toggling and the LSB in the **ni-example-app.exe** toggling at the same time (see [Figure 97](#)).

This completes the PROFINET Quickstart Guide Example. See the [Next Step: The Design Phase](#) section for more information on how to create your PROFINET device with customized parameters.



Figure 93. PLC Tags Menu



Figure 94. Adding New PLC Tags



Figure 95. PROFINET Network with Added Logic

RPG2 PROFINET QUICKSTART GUIDE



Figure 96. Compile and Download Button Locations

```

..\..\src\NI_main.c:78] INFO: Processing arguments...
..\..\src\NI_main.c:86] INFO: DONE
..\..\src\NI_main.c:90] INFO: Performing system startup...
..\..\src\NI_main.c:96] INFO: DONE
..\..\src\NI_main.c:100] INFO: Start time update process...
..\..\src\NI_main.c:106] INFO: DONE
..\..\src\NI_main.c:110] INFO: Starting up example application...
..\..\src\NI_main.c:137] INFO: DONE
..\..\src\NI_main.c:140] INFO: Link init...
..\..\src\NI_main.c:146] INFO: DONE
..\..\src\NI_main.c:150] INFO: Unified Interface stack init...
..\..\src\NI_main.c:159] INFO: DONE
..\..\src\NI_main.c:164] INFO: Set response timeout...
..\..\src\NI_main.c:170] INFO: DONE
..\..\src\NI_main.c:175] INFO: Find Network Interface...
..\..\src\NI_main.c:186] INFO: DONE
..\..\src\NI_main.c:190] INFO: Set transmit modes...
..\..\src\NI_main.c:204] INFO: DONE
..\..\src\NI_main.c:208] INFO: Get installed protocol...
..\..\src\NI_main.c:220] INFO: DONE (PROFINET)
..\..\src\NI_main.c:281] INFO: Set device 400...
..\..\src\NI_main.c:292] INFO: DONE
..\..\src\NI_main.c:296] INFO: Add item 500 to location 1...
..\..\src\NI_main.c:307] INFO: DONE
..\..\src\NI_main.c:311] INFO: Add item 501 to location 2...
..\..\src\NI_main.c:322] INFO: DONE
..\..\src\NI_main.c:326] INFO: Add item 502 to location 3...
..\..\src\NI_main.c:337] INFO: DONE
..\..\src\NI_main.c:342] INFO: Finalize configuration...
..\..\src\NI_main.c:353] INFO: DONE
current output data: item A: 0x0001, item B: 0x00000000, item C: 0x0000[..\..\src\NI_main.c:398] INFO:

```

Figure 97. Output Toggling for Bit 16 of Item A in the ni-example-app

NEXT STEP: THE DESIGN PHASE

To customize the example application based on your needs, consult the following user guides:

- ▶ The [RPG2 Hardware Design Integration Guide](#) section details how to embed and integrate the example software and other required hardware.
- ▶ The [RPG2 I/O Configuration Tool User Guide](#) section details how to create a customized I/O footprint for the RPG2 solution.
- ▶ The [RPG2 Unified Interface User Guide](#) section describes the language used by the **ni-example-app** application processor simulator to talk to the RPG2 solution. The Unified Interface is used with the [RPG2 I/O Configuration Tool User Guide](#) section to create a user application.
- ▶ To evaluate another protocol, continue to the quickstart guide of that protocol.

RPG2 ETHERNET IP QUICKSTART GUIDE

FEATURES

- ▶ Implicit input data transfer up to 1440 bytes
- ▶ Implicit output data transfer up to 1440 bytes
- ▶ Cycle time down to 2 ms (Ethernet application processor interface)
- ▶ DLR functionality
- ▶ Class 1 I/O connections
- ▶ Class 3 I/O connections

EQUIPMENT NEEDED

- ▶ EV-RPG2-EIP evaluation board
- ▶ Rockwell PLC
- ▶ 1 PC, using 2 PCs may be easier

SOFTWARE NEEDED

- ▶ [Network interface example application suite \(ni-example-app.exe\)](#)
- ▶ Rockwell Studio 5000 (RS Logix)

GENERAL DESCRIPTION

The RapID Platform Generation 2 (RPG2) module is a pretested industrial network interface designed to manage industrial protocols and network traffic. It supports PROFINET®, PROFINET isochronous real-time (IRT), Ethernet/IP®, Ethernet/IP with device level ring (DLR), EtherCAT®, and Modbus/TCP. The RPG2 module uses the Unified Interface to communicate with different protocols.

The Unified Interface is a custom protocol by Analog Devices, Inc., that allows interaction between an application processor and the RPG2 module. The Unified Interface is agnostic of the industrial protocol.

The Unified Interface ensures that the application processor hardware and software interface does not need to change when switching or updating protocols. The RPG2 module connects to an application processor via a universal asynchronous receiver transmitter (UART), Ethernet, or serial peripheral interface (SPI).

The EV-RPG2-EIP evaluation kit provides end to end evaluation of the communication path from the application processor to the programmable logic controller (PLC) over the Industrial Ethernet interface (using the [network interface example application suite](#)).

This user guide describes how to use the kit to set up and run a PLC example application.

For the example described in this quickstart guide, the application processor is a PC, and it communicates with the RPG2 module via an Ethernet port.

EVALUATION KIT SETUP FOR ETHERNET/IP

Refer to the [RPG2 RapID Platform Generation 2 User Guide](#) section to set up your hardware. Note that the default application processor link type is Ethernet. When a change to the link type to UART is required, refer to the [Link Configuration File](#) section of this document. See [Figure 98](#) through [Figure 101](#) for the setup for running the EtherNet/IP application example with the default link type for one or two PC(s). See [Figure 100](#) or [Figure 101](#) to set up the board with UART as the link type.

Use one PC if having both the Rockwell Studio 5000® RS Logix application and the [network interface example application suite](#) on a single PC is acceptable. Use two PCs if there is a requirement to clearly distinguish between the two applications.

For the rest of this user guide, two PCs are used to set up the Rockwell Studio 5000, RS Logix, and PLC application and to run the [network interface example application suite](#).

Note that LED A6/A7 is green in [Figure 98](#) through [Figure 101](#), which indicates that the board is preloaded with EtherNet/IP and that the startup process is complete.

RPG2 ETHERNET IP QUICKSTART GUIDE

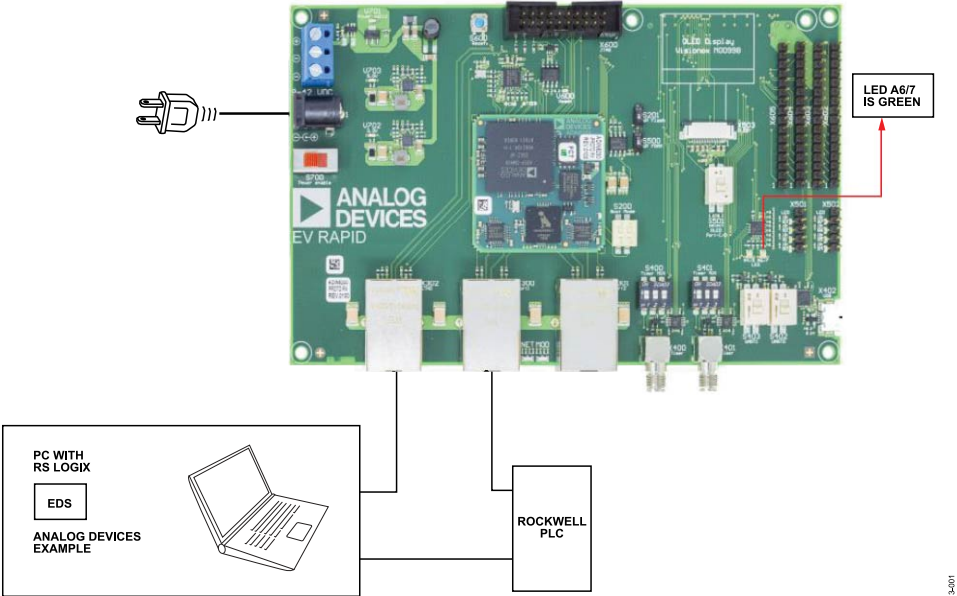


Figure 98. Setup for Running the EtherNet/IP Application Example with the Default Link Type and One PC

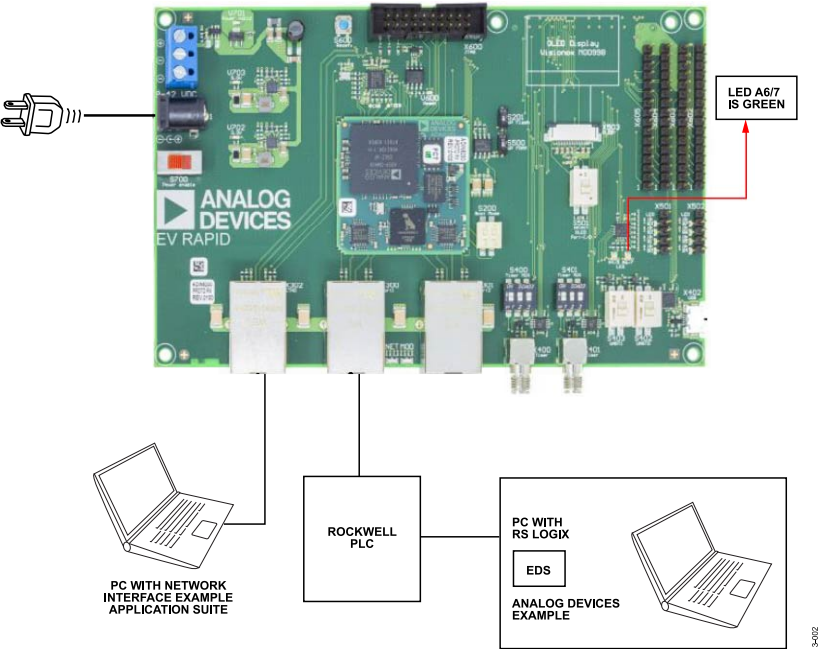


Figure 99. Setup for Running the EtherNet/IP Application Example with the Default Link Type and Two PCs

RPG2 ETHERNET IP QUICKSTART GUIDE

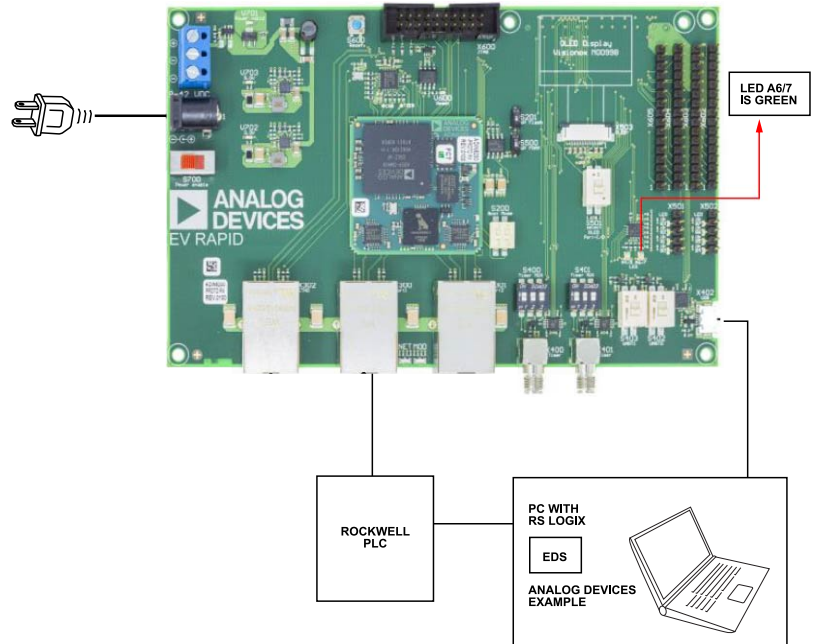


Figure 100. Setup for Running the EtherNet/IP Application Example with the UART Link Type and One PC

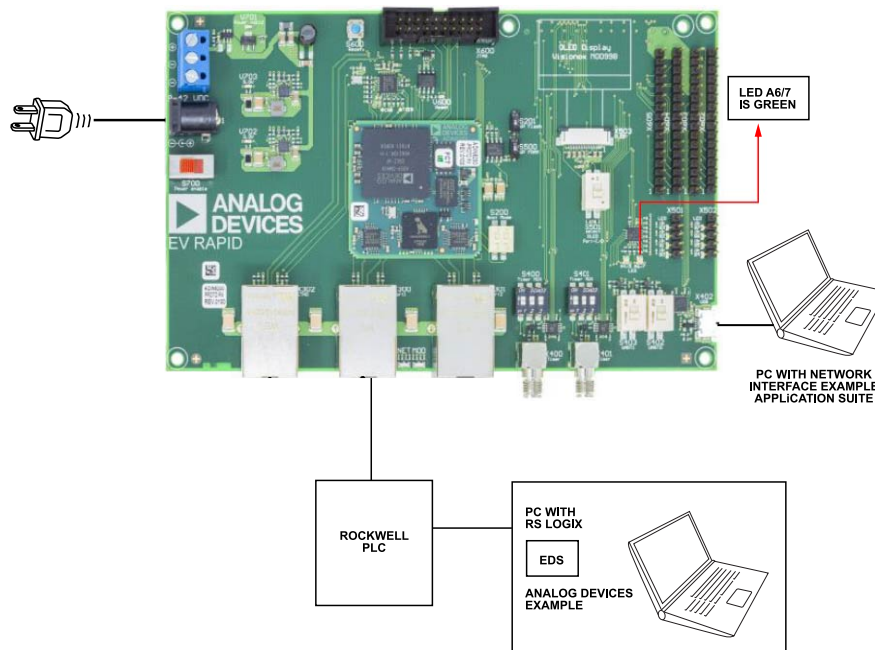


Figure 101. Setup for Running the EtherNet/IP Application Example with the UART Link Type and Two PCs

NETWORK INTERFACE APPLICATION SUITE WITH A PC RUNNING RS LOGIX

Enable communication between the application processor and the RPG2 module on the baseboard by using the USB virtual COM port on the baseboard. In addition to the RPG2 module evaluation kit, users must have the following items (at a minimum):

- ▶ A PC running the [network interface example application suite \(ni-example-app.exe\)](#)
- ▶ A PC running the Rockwell Studio 5000 RS Logix
- ▶ A Rockwell PLC

RPG2 ETHERNET IP QUICKSTART GUIDE

Setting Up the Host Processor Simulator Software

Obtain the network interface example application suite from the [main product page](#).

1. Go to the [main product page](#).
2. Click the **RPG2 Network Interface Example Application Suite (ZIP)** link.
3. Review the **Terms and Conditions** and accept.
4. Extract the contents of the .zip file. The extracted folder includes the **ni-example-app.exe** file that users must run from their command line.
5. To see the syntax to start up the example application using any link type, type **ni-example-app.exe -h** in the command line. To use the default link type, see [Figure 102](#) but ensure you change the identifier of the network interface card (NIC) to the local NET device to use.

Note that if you see **FATAL Bad Memory Block** while running the [network interface example application suite](#), ensure that you have the most updated version of Windows® WinPcap on your PC. The NIC identifier begins with **DeviceNPF_**.

```

.\..\src\NI_main.c:75] INFO: Welcome to ni-example-app!
.\..\src\NI_main.c:78] INFO: Processing arguments...
.\..\src\NI_main.c:86] INFO: DONE
.\..\src\NI_main.c:90] INFO: Performing system startup...
.\..\src\NI_main.c:96] INFO: DONE
.\..\src\NI_main.c:100] INFO: Start time update process...
.\..\src\NI_main.c:106] INFO: DONE
.\..\src\NI_main.c:110] INFO: Starting up example application...
.\..\src\NI_main.c:137] INFO: DONE
.\..\src\NI_main.c:140] INFO: Link init...
.\..\src\NI_main.c:146] INFO: DONE
.\..\src\NI_main.c:150] INFO: Unified Interface stack init...
.\..\src\NI_main.c:159] INFO: DONE
.\..\src\NI_main.c:164] INFO: Set response timeout...
.\..\src\NI_main.c:170] INFO: DONE
.\..\src\NI_main.c:175] INFO: Find Network Interface...
.\..\src\NI_main.c:186] INFO: DONE
.\..\src\NI_main.c:190] INFO: Set transmit modes...
.\..\src\NI_main.c:204] INFO: DONE
.\..\src\NI_main.c:208] INFO: Get installed protocol...
.\..\src\NI_main.c:220] INFO: DONE (Ethernet/IP)
.\..\src\NI_main.c:201] INFO: Set device 400...
.\..\src\NI_main.c:292] INFO: DONE
.\..\src\NI_main.c:296] INFO: Add item 500 to location 1...
.\..\src\NI_main.c:307] INFO: DONE
.\..\src\NI_main.c:311] INFO: Add item 501 to location 2...
.\..\src\NI_main.c:322] INFO: DONE
.\..\src\NI_main.c:326] INFO: Add item 502 to location 3...
.\..\src\NI_main.c:337] INFO: DONE
.\..\src\NI_main.c:342] INFO: Finalize configuration...
.\..\src\NI_main.c:353] INFO: DONE
Current output data: item A: 0x0000, item B: 0x00000000, item C: 0x0000[.\..\src\NI_main.c:398] INFO:
    
```

Figure 102. Network Interface Example Application Suite

EtherNet/IP Sample Configuration Setup

This user guide uses a PC running the Rockwell Studio 5000 RS Logix with a Rockwell PLC. This PLC example application assumes the module is configured as detailed in [Table 3](#).

This configuration defines three items with different input and output types. Item 500 defines 2 bytes of input and output data, Item 501 defines 4 bytes of analog input and analog output data, and Item 502 represents 2 bytes of control data (digital data). For more information on items, refer to the [RPG2 I/O Configuration Tool User Guide](#) section.

This section provides instructions for setting up and using the RPG2 module with the link type selected as Ethernet. Note that the RPG2 module must connect to a PC running the [network interface example application suite](#) (which is the leader network program) and a Rockwell PLC that is configured by another PC.

[Figure 103](#) shows the setup used in the following sections. See the [Evaluation Kit Setup for EtherNet/IP](#) section for details on how to set up the test network.

Table 3. EtherNet/IP Input and Output Sample Configuration

Item Number	Item Type	Consumer Size (Bytes)	Producer Size (Bytes)	Assembly Instance ID	
				Consumer	Producer
1 (Digital Inputs and Outputs)	Cyclic	2	2	100	101
2 (Analog Inputs and Outputs)	Cyclic	4	4	102	103
3 (Control Register)	Cyclic	0	2	Not applicable	104

RPG2 ETHERNET IP QUICKSTART GUIDE

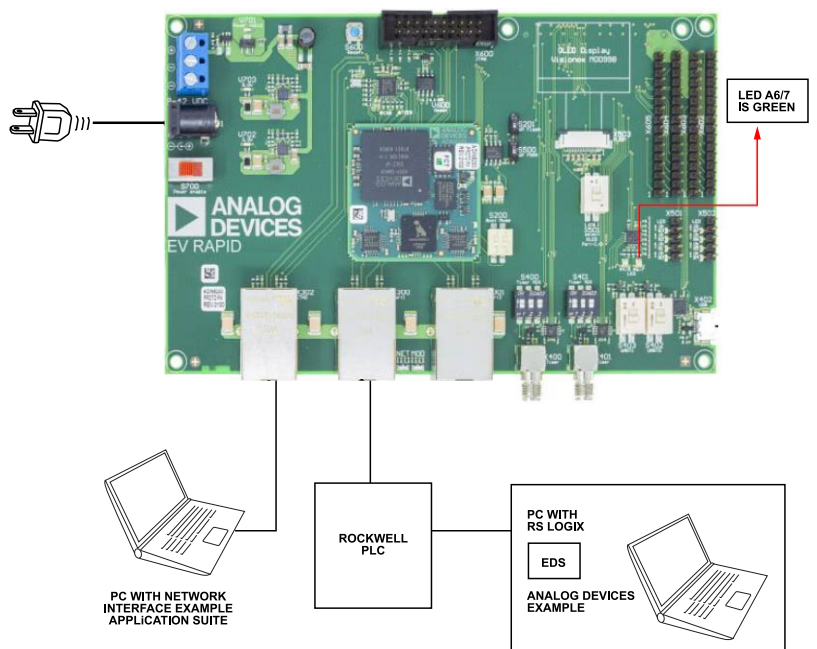


Figure 103. Sample Application Hardware Setup

Installing the EDS File

The following steps walk a user through using the Rockwell Studio 5000 RS Logix to register the electronic data sheet (EDS) file for the RPG2 module evaluation kit.

1. In the Rockwell Studio 5000 RS Logix main window, go to **Tools** and select the **EDS Hardware Installation Tool** in the dropdown menu to bring up the EDS installation options.
2. The **EDS Wizard** window displays two options: **Register a single file** or **Register a directory of EDS files**. This example only covers registering the default RPG2 EDS file as a single installation. In the **Named** box, navigate to where the EDS file is installed on the PC with the Rockwell Studio 5000 RS Logix and click **Next**.
3. The screen that appears next shows a green checkbox by the EDS file location, which indicates that the EDS file was found successfully.
4. Click **Next** to bring up the **Change Graphic Image** window. The **Basic REM Rapid Test Device** default image comes up as a suboption under the **Communications Adapter** folder.
5. Click **Next** to bring up the **Final Task Summary** window.
6. A dialogue box then appears asking **Do you want to register the Basic REM Rapid Test Device**.
7. Click **Next** and a window will appear confirming that you have successfully completed the EDS installation wizard.
8. Click **Finish** to complete this installation.

Note that the EDS file is located in the software zip file (**EDS_6_x.eds RPG2_EtherNetIP_Firmware.zip**) found on the [main product page](#).

This installation process can be done at any time before adding the RPG2 evaluation kit onto the EtherNet/IP network. For example, the task of adding the EDS file was arbitrarily done before creating the EtherNet/IP network.

Creating the EtherNet/IP Network

Take the following steps to create the EtherNet/IP network:

1. In the Rockwell Studio 5000 RS Logix main window, select **File > New** to bring up the **New Controller** window.
2. Under the **New Controller > Type** pulldown menu, the user must select the applicable EtherNet/IP controller in the dropdown menu. For the purposes of this example, the **1769-L18-ERM-BB1B** was selected. Note that any other EtherNet/IP controller can be used and that the overall logic for the example does not change.
3. Under the **New Controller > Type** dropdown menu, there is an **Expansion I/O** option. This example uses a controller with no modules for expansion I/O. Therefore, there are 0 I/O modules used for the controller, and the 0 modules option must be selected within the **Expansion**

RPG2 ETHERNET IP QUICKSTART GUIDE

I/O window. If the user has expansion I/O, the user must indicate this in the **Expansion I/O** window because not doing this will cause an error in downloading the project later.

4. Ensure that the applicable version of the Rockwell Studio 5000 RS Logix software is in use. For this example, Version 20 was selected. However, there may be a more recent revision. The overall logic of the example does not change. Therefore, ensure that the software version corresponds with the existing version of RS Logix in use.
5. In the Rockwell Studio 5000 RS Logix window, go to the **Path** dropdown menu to select the name and IP address that matches the controller in use. The **Path** relays to your PC what to expect with regards to the type of leader (the EtherNet/IP PLC) and what the IP address of the controller is.
6. Click **Download** and the **Done Downloading** message appears.
7. In the Rockwell Studio 5000 RS Logix main window, go to the **Communications** dropdown menu, select **Change to Remote Run**, and click **Yes**.
8. The main Rockwell Studio 5000 RS Logix window now has an Ethernet option on the tab on the left-hand side. Expanding this tab shows the PLC and IP address in use as a suboption within the **Ethernet** box.
9. The status of the network now shows that the PLC is found and that the basis for the network is established, which is shown under the **Rem Run** dropdown menu (**Run Mode**, **Controller OK**, and **I/O OK** are now all green).
10. If there is a yellow triangle with an exclamation point visible in the **Rem Run** dropdown menu, users should power cycle and reset the PLC and Rockwell Studio 5000 RS Logix.

Adding the EtherNet/IP Device

The purpose of this section is to add the RPG2 evaluation kit to the EtherNet/IP network that was created in the previous section. See [Figure 103](#) for the setup used in this example, where a PC running Studio 5000 (RS Logix), the Rockwell PLC, and RPG2 evaluation kit lie on the same daisy chain.

In a similar configuration, the PLC is supposed to have two Ethernet RJ45 connectors. These two ports must be set as linear and/or DLR in the controller properties so that a single IP address is assigned to both ports, and so that these ports work as part of the same ring. To set up the controller as dynamic host configuration protocol (DHCP) enabled and the Ethernet ports in linear and/or DLR mode, refer to the controller user guide from the Rockwell Automation website.

EtherNet/IP protocol specifications recommend using a DHCP server to assign an IP address, which is the behavior of the RPG2 evaluation kit out of the box. A DHCP server must be running on the same PC as the Rockwell Studio 5000 RS Logix software to run the application example.

The RPG2 EtherNet/IP device must have an assigned IP address by the DHCP server before proceeding with the following steps:

1. In the Rockwell Studio 5000 RS Logix main window, expand the icon showing the PLC previously referenced and an **Ethernet** icon will display three devices daisy-chained on network.
2. Click **New Module** to open the **Select Module Type** window.
3. Within the **Select Module Type** window, under the **Module Type Category Filters** section, select **Communications Adapter**, if not already selected.
4. Within the **Select Module Type** window, under the **Module Type Vendor Filters**, select **Analog Devices**, if not already selected.
5. The **New Module** window now appears. Users can fill in the **Name**, **IP Address**, and **Module Definition** fields. Users can name the EtherNet/IP device anything they want but must provide the IP address that was assigned by the DHCP server. For the purposes of this example, the module is named RPG2. Note that this naming directly affects the names used in Step 7 and Step 8 of the [Running the EtherNet/IP Application](#) section.
6. Click the **Module Definition** tab to bring up the **Digital Connection Point** window with a connection having 2 bytes of input and 2 bytes of output selected. This is the connection for Item 500 in the [RPG2 I/O Configuration Tool User Guide](#) section (see [Table 32](#)). Consult the [RPG2 I/O Configuration Tool User Guide](#) section for a more detailed discussion on the contents of this item. For this example, this connection is used.
7. Click **OK** to return to the Rockwell Studio 5000 RS Logix main window.
8. Right-click on the **RPG2 Module** that now appears in the main window and click **Properties**.
9. On the left side of the main Rockwell Studio 5000 RS Logix window, click **Connection** and set the **Requested Packet Interval (RPI)** field to 50 ms.

RPG2 ETHERNET IP QUICKSTART GUIDE

Running the EtherNet/IP Application

Take the following steps to run the EtherNet/IP application:

1. In the Rockwell Studio 5000 RS Logix main window, go to **Tasks > Main Task > Main Program** and click **Main Routine** to bring up a blank **Ladder Logic Rung** window.
2. The network is simple with a **Nominally Closed Input (Examine Off)** as the first part of the rung and an **Output Energize (Energize Coil)** on the end of the rung (see [Figure 104](#)).

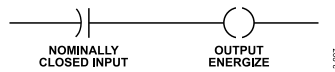


Figure 104. Sample Network

3. From the Rockwell Studio 5000 RS Logix main window, navigate to **Tasks > MainTask > MainProgram > Parameters and Local Tags** to name the input and output variables needed for the application.
4. Expand the options under **Program Tags** for the input data to **RPG2/RPG2.data/RPG2.I1.data[0]**. This is the byte that will be used for **Nominally Closed Input**.
5. Expand the options under **Program Tags** for the input data to **RPG2 > RPG2.data > RPG2..O1data[0]**. This is the byte that will be used for **Output Energize**.
6. Navigate back to the **Tasks > MainTasks > MainProgram > MainRoutine**.
7. Click the **Nominally Closed Input (Examine Off)** icon (see Step 2), go to **Main Program > MainRoutine**, and type **RPG2.I1.data[0].0**.
8. Click the **Output Energize (Energize Coil)** icon, go to **MainProgram > MainRoutine**, and type **RPG2.O1.data[0].0**.
9. Click **Communications > Download** and it prompts the user to change to **Remote Run**.
10. Click **Yes** to show the **Nominally Closed Input (Examine Off)** and **Output Energize (Energize Coil)** icons changing from green to white which indicates that the I/O is toggling on the network side. Note that the application processor simulator (**ni-example-app**) also shows in the I/O toggling.

```

\\.\.\src\NI_main.c:75] INFO: Welcome to ni-example-app!
\\.\.\src\NI_main.c:78] INFO: Processing arguments...
\\.\.\src\NI_main.c:86] INFO: DONE
\\.\.\src\NI_main.c:90] INFO: Performing system startup...
\\.\.\src\NI_main.c:96] INFO: DONE
\\.\.\src\NI_main.c:100] INFO: Start time update process...
\\.\.\src\NI_main.c:106] INFO: DONE
\\.\.\src\NI_main.c:110] INFO: Starting up example application...
\\.\.\src\NI_main.c:137] INFO: DONE
\\.\.\src\NI_main.c:140] INFO: Link init...
\\.\.\src\NI_main.c:146] INFO: DONE
\\.\.\src\NI_main.c:150] INFO: Unified Interface stack init...
\\.\.\src\NI_main.c:159] INFO: DONE
\\.\.\src\NI_main.c:164] INFO: Set response timeout...
\\.\.\src\NI_main.c:170] INFO: DONE
\\.\.\src\NI_main.c:175] INFO: Find Network Interface...
\\.\.\src\NI_main.c:186] INFO: DONE
\\.\.\src\NI_main.c:190] INFO: Set transmit modes...
\\.\.\src\NI_main.c:204] INFO: DONE
\\.\.\src\NI_main.c:208] INFO: Get installed protocol...
\\.\.\src\NI_main.c:220] INFO: DONE (EtherNet/IP)
\\.\.\src\NI_main.c:281] INFO: Set device 400...
\\.\.\src\NI_main.c:292] INFO: DONE
\\.\.\src\NI_main.c:296] INFO: Add item 500 to location 1...
\\.\.\src\NI_main.c:307] INFO: DONE
\\.\.\src\NI_main.c:311] INFO: Add item 501 to location 2...
\\.\.\src\NI_main.c:322] INFO: DONE
\\.\.\src\NI_main.c:326] INFO: Add item 502 to location 3...
\\.\.\src\NI_main.c:337] INFO: DONE
\\.\.\src\NI_main.c:342] INFO: Finalize configuration...
\\.\.\src\NI_main.c:353] INFO: DONE
Current output data: Item A: 0x0001, Item B: 0x00000000, Item C: 0x0000[\\.\.\src\NI_main.c:398] INFO:

```

Figure 105. I/O Toggling on the Host Simulator

NEXT STEP: DESIGN PHASE

After a module has been initialized and configured to use the EtherNet/IP inputs and outputs, the next phase is the design.

For more information on how to progress to the design phase, consult the following user guides:

- ▶ The [RPG2 Hardware Design Integration Guide](#) section explains how to embed and integrate the module for use into the system. The [RPG2 Hardware Design Integration Guide](#) section also includes:
 - ▶ Detailed descriptions of the signals on the module.
 - ▶ Information about the AC, DC, thermal, and power requirements.
 - ▶ Further details about the host interface.

RPG2 ETHERNET IP QUICKSTART GUIDE

- ▶ The pin configuration of the RPG2 module.
- ▶ The [RPG2 I/O Configuration Tool User Guide](#) section explains how to create a customized input and output footprint for the system.
- ▶ The [RPG2 Unified Interface User Guide](#) section explains how to interact with the Analog Devices RPG2 module (embedded or otherwise) via the Unified Interface. The Unified Interface is how a host processor communicates with a network interface module by means of a UART, a parallel interface, or Ethernet.
- ▶ To evaluate another protocol, continue to the quickstart guide of that protocol.

The Unified Interface is the custom protocol that allows interaction between a host processor and the modular solution. When the Unified Interface commands are integrated into the host software, the solution takes care of the Industrial Ethernet protocol communication.

The standard Analog Devices configuration for EtherNet/IP uses the three items detailed in [Table 3](#).

Note that the needs of a system are not typically encompassed in the default configuration data that is provided in the network interface module. The configuration tool allows the user to create a configuration that fits the needs of a particular system.

RPG2 UNIFIED INTERFACE USER GUIDE

INTRODUCTION

The Unified Interface is an interface by which an application processor communicates through an industrial Ethernet network with another system to configure and interact with it.

The Unified Interface provides a common messaging protocol across a variety of different link types. This common messaging protocol allows the application processor to be industrial Ethernet protocol agnostic as the core set of messages have nothing to do with any particular industrial protocol. Extensions may exist to take advantage of protocol specific features but are not a required part of the core messaging protocol.

BLOCK DIAGRAM FOR EMBEDDED DESIGN AND MODULE USERS

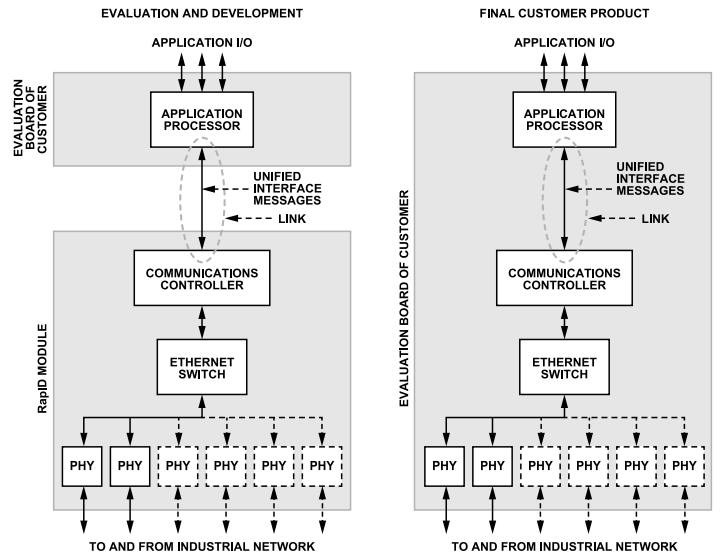


Figure 106.

UNIFIED INTERFACE BACKGROUND

A Unified Interface includes the following:

- ▶ A communication controller, which is the embedded processor responsible for managing transmission and/or reception of Unified Interface messages on the communication side and running the industrial Ethernet protocol software.
- ▶ A communication side, consisting of the [ADSP-CM409F](#) hardware with the Unified Interface application software and an industrial networking application software running on it.
- ▶ An application processor, which is the processor responsible for configuring the communication side via the Unified Interface protocol as well as the application specific input and output (I/O), for example, the motor control, ADC, and more.
- ▶ An application side, which is the system portion that includes the processor in use by the user containing the API described in the [Unified Interface API](#) section.
- ▶ A link, which is the low level hardware interface used between the application processor and the communication controller to transfer Unified Interface messages. For example, serial peripheral interface (SPI), UART, and Ethernet. See the [Application Processor Link Type](#) section for more information.
- ▶ A Unified Interface protocol, which is a set of messages that makes up the Unified Interface.
- ▶ An I/O application, which is the main function that provides the entry point for the embedded firmware.
- ▶ A Unified Interface application, which is the I/O application that implements the Unified Interface protocol.
- ▶ A system, which is the combination of the application side components and the communication side components that make up the industrial Ethernet device.
- ▶ A transaction, which is the transfer of type Unified Interface message bytes over the configured link.

RPG2 UNIFIED INTERFACE USER GUIDE

DESIGN FLOW USING THE UNIFIED INTERFACE

See [Figure 107](#) and the following sections for an outline of the steps to make an industrial Ethernet device using a Unified Interface.

Prior to a user performing the porting exercise, the user must run through the quickstart guide example for the chosen protocol of the user. Refer to the [main product page](#) to download the protocol specific quickstart guide and run the application processor example for that given protocol. The Unified Interface is the same no matter what protocol is chosen.

Run the Quickstart Guide Example

Regardless of the protocol, the quickstart guide for each protocol directs users on how to use the **ni-example-app** executable. Refer to the [RPG2 EtherCAT Quickstart Guide](#), [RPG2 EtherNet IP Quickstart Guide](#), and [RPG2 Profinet Quickstart Guide](#) sections of this reference manual.

Design Hardware for the Industrial Ethernet Device

While this step can occur during several portions of the design phase, doing so before creating the needed link porting layer may be the most efficient. However, this need can vary from device to device. Refer to the [Link Configuration](#) section of this document for more information.

Create a Link Porting Layer for the Applications Processor

The link porting layer is the responsibility of the customer to create. It is the software layer that hooks up the Unified Interface stack to the hardware drivers for any given platform. Refer to the [Porting and Customization](#) section for more details on how to implement this task.

Port the Example to the Applications Processor

There are examples distributed for both a Windows®-based application and one that is based on an Arm® Cortex®-M4-based platform. The intention is to port specific portions of the code as needed and develop the others. Refer to the [Porting and Customization](#) section for more details on what a user must include in the user environment. [Table 4](#) also outlines what the responsibilities of the customer are vs. what is provided by Analog Devices. See the [NI Example Application for Windows](#) section for a complete breakdown of the Windows application and some of the modification steps.

Make Modifications to the Example Using the Unified Interface Application Programming Interface

The Unified Interface application processor interface (API) is a driver that enables the user to interact with the RapID Platform Generation 2 (RPG2) solution. The user makes additions, modifications, or subtractions as applicable to the device and system needs of the user. The needs of every system varies, and the user must determine what to implement or not based on the system needs of the user. See the [Unified Interface API](#) section for a detailed description of the API.

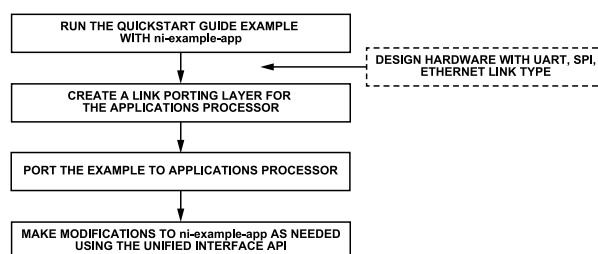


Figure 107. Application Side Software Architecture

APPLICATION PROCESSOR LINK PORTING LAYER

The link porting layer is the layer between the Unified Interface stack and the drivers of a customer that is used to control the hardware on the application processor. The user is expected to provide a driver and modify the link porting layer to interact with that driver.

[Figure 108](#) shows the architecture of the application processor software, and [Table 4](#) details which party is responsible for implementing this architecture. While there may be some specific processor examples for the link porting layer, the user is ultimately responsible for this porting.

RPG2 UNIFIED INTERFACE USER GUIDE

Application I/O Drivers

The application I/O drivers are the portion of the code where the application processor does all of its tasks that are not part of the networking part of the device. Examples are toggling physical I/O, physically starting or stopping the motor, and passing data to a liquid crystal display (LCD). The application I/O drivers layer is not part of the RPG2 product and is therefore not supported.

Application

The application is where the user makes calls to the API provided by the Unified Interface. The user can set or retrieved the I/O data or communicate with the communications processor to do other Ethernet related tasks.

Ni-api-srv

Ni-api-srv is the actual Unified Interface driver that a user incorporates into the software environment of the user.

The ni-api-srv has many different functions delivered for interaction with the ADIN2299 from a host processor. They all do different items, the main highlights are covered in this document. Beyond that, please see the .h files to get an idea of what can be done with the ni-api.

Ui-stk

Ui-stk is the software stack where the Unified Interface messages are handled and processed to and from the network. **Ui-stk** must be included in the software environment of the user.

Ui-xxx-lpl-srv

Ui-xxx-lpl-srv is a project/file where the user links the physical hardware (drivers) of the user with the Unified Interface stack. There are sample link porting layers distributed for Windows and the [ADSP-CM409F](#).

Link Hardware Drivers

The link hardware drivers are the drivers that the user has for the application processor interface type (UART, SPI or Ethernet) of the user.

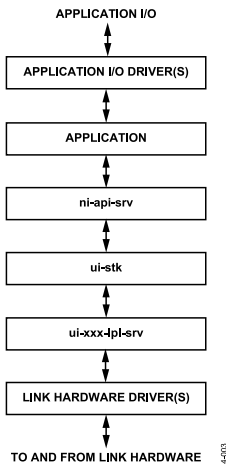


Figure 108. User Software Layers

Table 4. Software Layer Development

Layer	Developed By	Modification
Application I/O Drivers	Analog Devices and provided as an example	For creation, modification, and maintaining by the customer only
Application	Analog Devices and provided as an example	For modification by the customer only
Ni-api-srv	Analog Devices	Not modified by the customer
Ui-stk	Analog Devices	Not modified by the customer
Ui-xxx-lpl-srv	Analog Devices and provided as an example	For modification by the customer

RPG2 UNIFIED INTERFACE USER GUIDE

Table 4. Software Layer Development (Continued)

Layer	Developed By	Modification
Link Hardware Drivers	Analog Devices and provided as an example	For modification by the customer

PORTING AND CUSTOMIZATION

This section describes how to implement a link type into the Unified Interface. The available [software zip file](#) contains examples for Ethernet, SPI, and UART link types for an embedded platform, and the zip file also contains Ethernet and UART link type examples for a Windows-based platform.

Link Porting Layer Customization

To customize the link porting layer for link type hardware and/or a driver for the processor, add a new header and C source file that contains functionality for the following items:

- ▶ Link initialization
- ▶ Link configuration
- ▶ Message transmission
- ▶ Message receive handling
- ▶ Link deinitialization

Note that the link porting layer library can be renamed during the customization process.

Link Initialization

Use the link initialization function to set up the link hardware. A pseudo code example follows:

```
LinkInitialization(Link-specific parameters)
{
    Initialize the link
    if link initialization error
    then return LPL_ERROR;
    return LPL_OK;
}
```

Link Configuration

Use the link configuration function to configure link specific parameters. A pseudo code example follows:

```
LinkConfiguration(Link-specific parameters)
{
    Configure the link
    if link configuration error
    then return LPL_ERROR;
    return LPL_OK;
}
```

If required, the link configuration can be performed inside the link initialization function. A pseudo code example follows:

```
LinkInitialization(Link-specific parameters)
{
    Initialize the link
    if Link initialization error
    then return LPL_ERROR;
```

RPG2 UNIFIED INTERFACE USER GUIDE

```
Configure the link
if link configuration error
then return LPL_ERROR;
return LPL_OK;
}
```

Message Transmission

Use the message transmission function to send a message from the Unified Interface stack via the link. This function is called by the **ui-stk** via a function pointer. The signature (return type, argument types, and argument order) of this function must match **UI_TxFunc_t**. A pseudo code example follows:

```
MessageTransmission(data location, message size)
{
    Transmit the Unified Interface message on the link
    if data transmission error
    then return LPL_ERROR;
    return LPL_OK;
}
```

Message Receive Handling

Message receive handling receives incoming data from the link and forwards this data to the Unified Interface stack. This handling can be accomplished in a thread or an interrupt handler. To minimize the amount of data movement, the Unified Interface stack allows incoming data to be directly copied into its input buffer space. To obtain the location where the incoming data must be placed, call **UI_GetRxLocation()**. The incoming data then is sequentially copied starting at that location. From there, **UI_ProcessMsgData()** is called to begin processing of the newly received data. A pseudo code example follows:

```
MessageReception()
{
    Request the current receive location → UI_GetRxLocation()
    Read the incoming data and place the data into the buffer
    Call UI_ProcessMsgData();
}
```

To keep the internal data pointers in sync, for every call to **UI_ProcessMsgData()**, there has to be a preceding call to **UI_GetRxLocation()**.

Link Deinitialization

Use the link deinitialization function to deinitialize the link hardware. This function may not be required for all platforms. The designer must decide if this functionality is required for the platform in use. A pseudo code example follows:

```
Link Deinitialization()
{
    Deinitialize the link
    if link deinitialization error
    then return LPL_ERROR;
    return LPL_OK;
}
```

RPG2 UNIFIED INTERFACE USER GUIDE

Porting the Platform Support Service

This section describes the functions available in the platform support service of the example application and how to port these functions to the destination platform. These functions are examples that must be modified and adapted by the user. The platform support service is hardware specific.

PLAT_StartupSystem()

The **PLAT_StartupSystem()** function initializes the platform the example application runs on. This function turns on clocks and sets up any peripherals that are required for the operation of the platform. If the platform successfully initializes, this function returns **PLAT_OK**, and if an error occurs during initialization, this function returns **PLAT_ERROR**.

PLAT_ProcessArgs()

The **PLAT_ProcessArgs()** function processes the passed in command line arguments. If the command line arguments successfully processed, this function returns **PLAT_OK**. If no command line arguments were supplied, and no error occurred, this function returns **PLAT_NO_ARGS**. If any of the command line arguments did not process, this function returns **PLAT_ERROR**. If no command line arguments have to be processed, this function simply returns a success code. If the embedded application does not have a command line capability, do not call on this function from **ni-example-app**.

PLAT_InitLink()

The **PLAT_InitLink()** function initializes the local link where Unified Interface messages are transmitted and received. If the link initialization was successful, this function returns a pointer to the function that must be used to transmit the Unified Interface messages. If the link initialization failed, it returns a **NULL** pointer.

The included example application allows for link type selection at run time. If only one link type is required, use the function as follows:

```
UI_TxFunc_t *PLAT_InitLink()
{
    int32_t result;
    UI_TxFunc_t *msgTx_p;
    result = LPL_CustomInit(Link specific parameters);
    if (result != LPL_OK)
        msgTx_p = NULL;
    else
        msgTx_p = LPL_CustomTxMsg;
    return msgTx_p;
}
```

PLAT_StartTimeUpdate()

The **PLAT_StartTimeUpdate()** function starts the process that notifies the network interface API service that a given time interval has elapsed (that is, calls **NI_TimeUpdate()**) by using platform specific tasking and/or timing resources. On a platform with an operating system (for example, Windows), this this functionality can be a thread that calls **NI_TimeUpdate()** periodically. On a platform without an operating system, implement this functionality as a periodic interrupt handler. There are no predefined requirements on the actual implementation of this function other than that this function must reliably supply ticks to the network interface API service. The time update process, whether it is a thread or a timer interrupt, must run at a higher priority than the application. Otherwise, the time does not advance from the perspective of **ni-api-srv** and the timeouts do not work.

PLAT_StartApplication()

The **PLAT_StartApplication()** function starts the example application layer. The startup of the application is dependent on the platform. On an embedded platform, this function can call **NI_application()**. On a platform with an operating system (for example, Windows), this function can spawn a thread that calls **NI_application()**.

RPG2 UNIFIED INTERFACE USER GUIDE

PLAT_TerminateExecution()

The **PLAT_TerminateExecution()** function is called by the example application to terminate execution. This function gives the platform support library the opportunity to trap errors and then clean up and/or release resources. This function is an expected exit point for the executable, and this function is not expected to return. If applicable, a success or failure code returns to the operating system.

PLAT_ProtectUiStack()

The **PLAT_ProtectUiStack()** function can take the mutex on a system with an operating system so that each thread or process must wait for ownership of the mutex before the function can access the Unified Interface stack. On a system without an operating system, this function may disable interrupts.

PLAT_UnprotectUiStack()

The **PLAT_UnprotectUiStack()** function reverses the actions of the **PLAT_ProtectUiStack()** function. On a system with an operating system, this function can release the ownership of the mutex so that other threads or processes can access the Unified Interface stack. On a system without an operating system, this function can enable interrupts.

NI Example Application for Windows

The example application for Windows is called **ni-example-app**. This application is an application processor simulator that contains the needed API calls to interact with the **Communications Controller**. The **ni-example-app** application can be downloaded from the [main product page](#).

Extract the zipped file to a local directory and navigate to **..ni-example-app\project\vsni-example-app**.

Open **ni-example-app.sln** with Visual Studio 2015 or later.

The **Solution Explorer** will show the following five projects:

- ▶ ni-api-srv
- ▶ ni-example-app
- ▶ ni-windows-support-srv
- ▶ ui-stk
- ▶ ui-windows-lpl-srv

If not already selected, select **x86** as the **Solution Platform**.

Note that **ni-example-app** exists as an example for how to interface with a processor. This does not mean it is real time from industrial protocol standards. Therefore the intent is to use **ni-example-app** for how to implement code. For final embedded devices, it is recommended customers do not use the PC as a host.

Specifying the Endianness

To specify the endianness, take the following steps:

1. In the **Solution Explorer**, right-click **ui-stk**.
2. Under **Configuration Properties**, expand **C/C++**.
3. Select **Preprocessor**.
4. Select **Preprocessor Definitions** (see [Figure 109](#)).
5. Click the dropdown arrow and select **<Edit...>**.

The default definition is **UI_LITTLE_ENDIAN**. A preprocessor definition is required either **UI_LITTLE_ENDIAN** or **UI_BIG_ENDIAN**. For a Windows platform, little endian is expected. For an embedded processor, endianness is something that varies depending on the application processor chosen.

RPG2 UNIFIED INTERFACE USER GUIDE

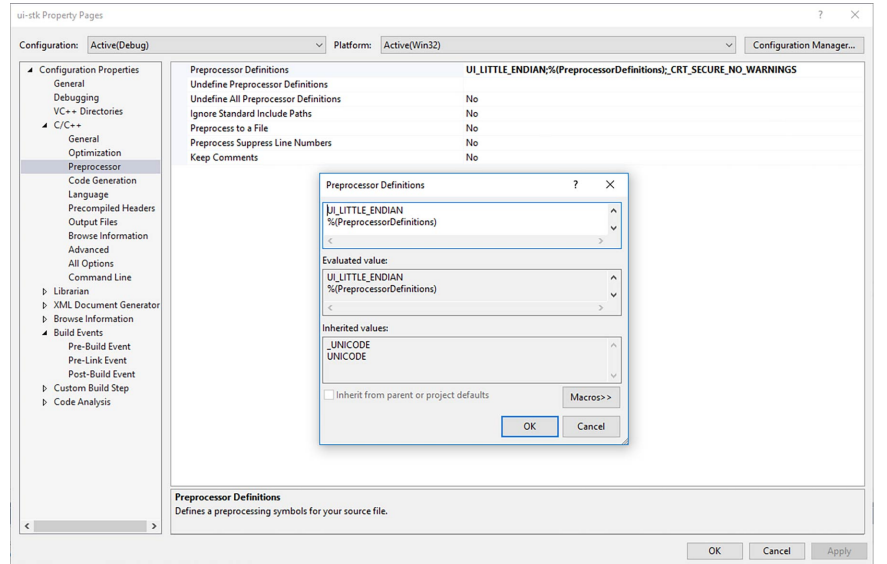


Figure 109. Specifying the Endianness

Setting the Command Arguments

To set the **Command Arguments**, take the following steps:

1. In the **Solution Explorer**, right-click **ni-example-app**.
2. Under **Configuration Properties**, select **Debugging**.
3. Select **Command Arguments**.

Click the dropdown arrow and select **<Edit...>**.

Ethernet as Application Processor Interface

To set the Ethernet as an application processor interface, take the following steps:

1. To see all available network devices, enter **-l ETH** under **Command Arguments** (see Figure 110).
2. Click **Apply**.
3. Click **OK**.
4. Press **Ctrl + Shift + B** to build the solution.
5. Press **Ctrl + F5** to start without debugging.

The following output appears in the **Console** window:

```
>ni-example-app.exe -l ETH
[.....srcNI_main.c:75] INFO: Welcome to ni-example-app!
[.....srcNI_main.c:78] INFO: Processing arguments...
A network device must be specified if the ETH link type is chosen.
Use the -n option to set the desired network device.
List of available network devices:
1: DeviceNPF_{080146B3-7B09-488E-8C10-BC05B800F39B}, (Microsoft)
2: DeviceNPF_{9AC18211-0815-4F54-8060-C5904AD9C4F2}, (NdisWan Adapter)
3: DeviceNPF_{8767E47B-BAF3-4821-8A1D-F8EDE935F8C5}, (Microsoft)
4: DeviceNPF_{E26E29A0-5899-4925-B0EF-2499B98570C8}, (Realtek USB NIC)
5: DeviceNPF_{CAB0CB19-2CCA-4018-9FB5-B816CD055359}, (NdisWan Adapter)
6: DeviceNPF_{F1F73818-E911-448D-9C9A-40511FB69628}, (Microsoft)
7: DeviceNPF_{A550E718-E38D-49ED-8873-71E7B6A74923}, (NdisWan Adapter)
```

RPG2 UNIFIED INTERFACE USER GUIDE

```
[.....srcNI_main.c:81] INFO: FAILED
```

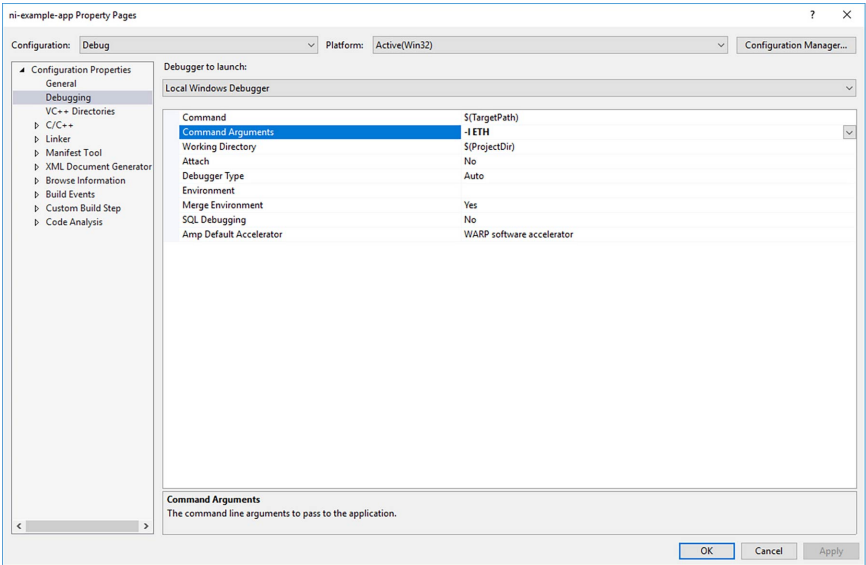


Figure 110. Ethernet Command Arguments

Take the following steps to set the Ethernet **Command Arguments**:

1. Select the appropriate network device and change the **Command Arguments** to **-I ETH -n DeviceNPF_{E26E29A0-5899-4925-B0EF-2499B98570C8}**.
2. Click **Apply**.
3. Click **OK**.

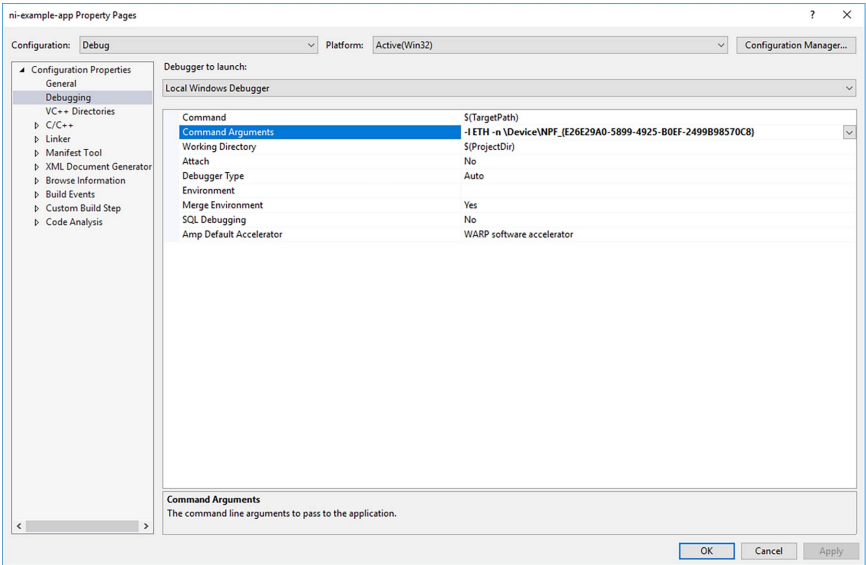


Figure 111. Setting Ethernet Command Arguments

RPG2 UNIFIED INTERFACE USER GUIDE

UART as Application Processor Interface

To set the UART as an application processor interface, take the following steps:

1. Open the **Device Manager** and expand the **Ports (COM & LPT)** section. A list of two Silicon Labs dual CP2105 USB to UART bridge: enhanced COM ports appears as follows:
 - a. Silicon Labs Dual CP2105 USB to UART Bridge: Enhanced COM Port (COM6)
 - b. Silicon Labs Dual CP2105 USB to UART Bridge: Standard COM Port (COM7)
2. Note the name of the enhanced COM port and enter **-I UART -c COM6** under **Command Arguments**.
3. Click **Apply**.
4. Click **OK**.

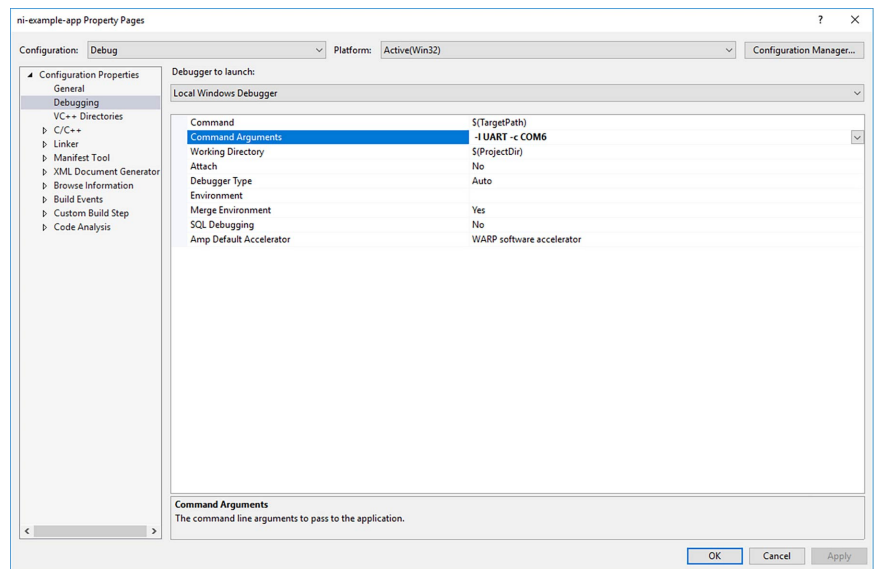


Figure 112. Setting the UART Command Arguments

Running Ni-example-app in Debug Mode

Take the following steps to run **ni-example-app** in debug mode:

1. Before running the **ni-example-app**, users must set up the EV-RPG2 board chosen as detailed in the [RPG2 RapID Platform Generation 2 User Guide](#) section.
2. Press **Ctrl + Shift + B** to build the solution.
3. Press **Ctrl + F5** to start without debugging.

The following output appears in the **Console** window:

```
[.....srcNI_main.c:75] INFO: Welcome to ni-example-app!
[.....srcNI_main.c:78] INFO: Processing arguments...
[.....srcNI_main.c:86] INFO: DONE
[.....srcNI_main.c:90] INFO: Performing system startup...
[.....srcNI_main.c:96] INFO: DONE
[.....srcNI_main.c:100] INFO: Start time update process...
[.....srcNI_main.c:106] INFO: DONE
[.....srcNI_main.c:110] INFO: Starting up example application...
[.....srcNI_main.c:137] INFO: DONE
[.....srcNI_main.c:140] INFO: Link init...
[.....srcNI_main.c:146] INFO: DONE
[.....srcNI_main.c:150] INFO: Unified Interface stack init...
```

RPG2 UNIFIED INTERFACE USER GUIDE

```
[.....srcNI_main.c:159] INFO: DONE
[.....srcNI_main.c:164] INFO: Set response timeout...
[.....srcNI_main.c:170] INFO: DONE
[.....srcNI_main.c:175] INFO: Find Network Interface...
[.....srcNI_main.c:186] INFO: DONE
[.....srcNI_main.c:190] INFO: Set transmit modes...
[.....srcNI_main.c:204] INFO: DONE
[.....srcNI_main.c:208] INFO: Get installed protocol...
[.....srcNI_main.c:220] INFO: DONE (EtherNet/IP)
[.....srcNI_main.c:281] INFO: Set device 400...
[.....srcNI_main.c:292] INFO: DONE
[.....srcNI_main.c:296] INFO: Add item 500 to location 1...
[.....srcNI_main.c:307] INFO: DONE
[.....srcNI_main.c:311] INFO: Add item 501 to location 2...
[.....srcNI_main.c:322] INFO: DONE
[.....srcNI_main.c:326] INFO: Add item 502 to location 3...
[.....srcNI_main.c:337] INFO: DONE
[.....srcNI_main.c:342] INFO: Finalize configuration...
[.....srcNI_main.c:353] INFO: DONE
```

The item and device constructs are the default values for the predistributed sample industrial Ethernet configuration database. For more information, see the [RPG2 I/O Configuration Tool User Guide](#) section.

While not in the **Console** display, once the user has called the configuration complete in the application processor code, the user then must make calls specific to the inputs and outputs as shown in [Table 6 \(NI_SetInputData and NI_GetOutputData\)](#).

The program then continuously calls the **NI_ProcessEvents()** function to process events. Inside of **NI_ProcessEvents()** there is an input data subroutine and an output data subroutine.

It should also be noted that the **ni-example-app** line numbers are put out as a reference only, actual line numbers may change as the executable goes through changes.

Table 5. Commands to API Calls Pre ConfigComplete

Console Output	Line Number	API Function
Find Network Interface...	175	NI_Init()
Set transmit modes	190	NI_SetTransmitModes()
Get installed protocol	208	NI_GetProtocol()
Set device 400	281	NI_SetDevice()
Add item 500 to location 1	296	NI_AddItem()
Add item 501 to location 2	311	NI_AddItem()
Add item 502 to location 3	326	NI_AddItem()
Finalize configuration	342	NI_ConfigComplete()
NO LONGER in Config Time	353 and on	Not applicable
Not Applicable	358	NI_ProcessEvents()

Table 6. Cyclic I/O Processing in NI_API.c

Line Number	API Function	Notes
467	NI_OutputDataHandler()	Calls NI_GetOutputData()
472	NI_InputDataLatchHandler()	Calls NI_SetInputData()
367	NI_GetOutputData()	Gets data from the network through the communications controller
424	NI_SetInputData()	Sends data to the network through the communications controller

```
void NI_ProcessEvents(void) {
    int32_t result;
    UI_msgType_t msgType;
```

RPG2 UNIFIED INTERFACE USER GUIDE

```

// Call any time-dependent handlers
if (NI_inDataTransmitMode_g == NI_transmitMode_onTime) {
if (NI_time_us_g >= NI_nextInDataTxTime_g) {
NI_InputDataLatchHandler();
NI_nextInDataTxTime_g += NI_inDataTxPeriod_us_g;
}
}
/* Checks for additional calls to time-dependent handlers go here */
// Call any message-dependent handlers
result = UI_GetNextMsgType(UI_msgSetEvent, &msgType);
if (result == UI_OK) {
switch (msgType) {
case UI_itemOutData:
NI_OutputDataHandler();
break;
case UI_itemInDataLatch:
UI_ParseItemInDataLatch();
NI_InputDataLatchHandler();
break;
default:
// TODO: Flush unrecognized messages
break;
}
}
}
}

```

UNIFIED INTERFACE API

The Unified Interface API is protocol agnostic. The user makes calls into the given functions as applicable for the application of the user. It should be noted that not all commands are detailed in this document, a user should use the .h header files for understanding the API of the Unified Interface. Shown below are some of the highpoints of the API.

The RapID Platform Generation 2 (RPG2) block diagram is shown in [Figure 113](#). As shown in [Figure 114](#), the RPG2 is a black box that users must only worry about when using the Unified Interface API.

[Figure 115](#) shows a diagram of some of the main Unified Interface messages and details of when these messages are seen in the time of the system. Message validity is determined pre and post **ConfigComplete**, which is shown in [Figure 115](#) for the system. To do industrial Ethernet communication, call **ConfigComplete** first.

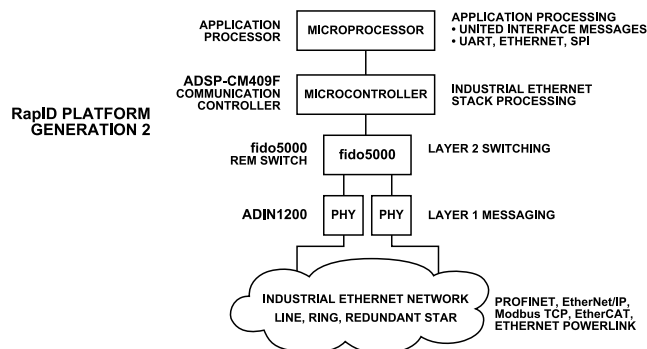


Figure 113. RPG2 System Block Diagram

RPG2 UNIFIED INTERFACE USER GUIDE

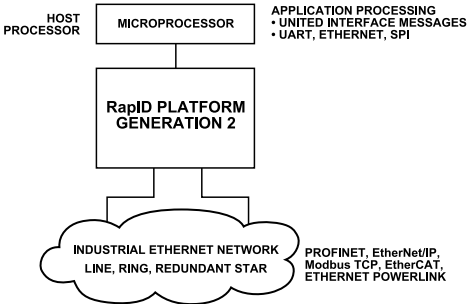


Figure 114. RPG2 System Block Diagram with the RapID Platform Generation 2 as a Black Box

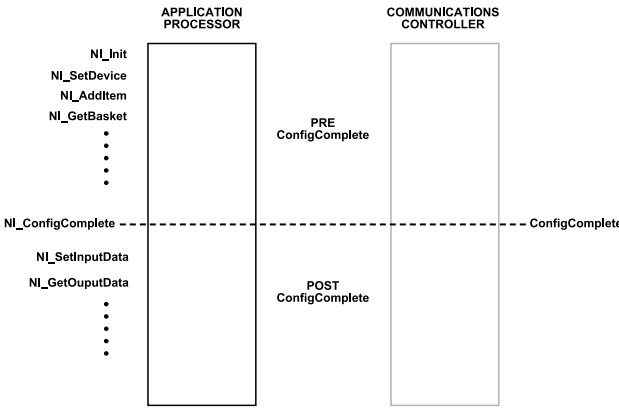


Figure 115. RPG2 System Block Diagram

Some functions are valid only before the **NI_ConfigComplete()** function. Some functions are valid only after the **NI_Config Complete()** function. Some are valid in either case, which is dependent on the industrial Ethernet protocol currently used. [Table 7](#) lists the valid time for NI functions.

Table 7. Command Valid After NI_ConfigComplete()

Command	Valid Time
NI_Init	Before
NI_GetProtocol	Both
NI_SetDevice	Before
NI_AddItem	Protocol dependent
NI_ConfigComplete	Not applicable
NI_GetOutputData	After
NI_SetInputData	After
NI_TimeUpdate	Both
NI_SetResponseTimeout	Before
NI_OutputDataHandler	After
NI_InputDataLatchHandler	After
NI_SetDataTransmitModes	Before
NI_GetBasket	Before
NI_ProcessEvents	After
NI_MsgReadyCallback	After

NI_Init()

The **NI_Init()** function initializes the Unified Interface and gets the Unified Interface to where the Unified Interface is stable and able to receive additional messages that are required. This function cannot be called after the **NI_ConfigComplete()** function.

RPG2 UNIFIED INTERFACE USER GUIDE

NI_Init() Function Prototype

The prototype of this function follows:

```

/* Initialize the local API software in preparation for communication via the
 * Unified Interface and attempt to detect an attached Network Interface module
 *
 * maxAttempts The maximum number of attempts this function will make to
 * detect an attached Network Interface module
 *
 * Returns NI_OK on successful initialization
 * NI_ERROR on failure during initialization
 * NI_ERROR if Network Interface module found but response incorrect
 * NI_NOT_FOUND if couldn't detect a Network Interface module
 */
int32_t NI_Init(uint16_t maxAttempts);

```

The main purpose of **NI_Init()** is to determine if the application processor interface is correctly connected and ready for further Unified Interface communication. The user can specify the maximum number of times the application processor of the user attempts to find the module.

NI_GetProtocol()

The **NI_GetProtocol()** function returns the current protocol that the module is running.

NI_GetProtocol() Function Prototype

The prototype of this function follows:

```

/* Get the Industrial Ethernet protocol installed on the attached Network
 * Interface module
 *
 * No parameters
 *
 * Returns >=0: the installed Industrial Ethernet protocol (typecast returned
 * value to NI_protocol_t to get enumerated value)
 * <0: NI_TIMEOUT if operation timed out
 * NI_ERROR if a Unified Interface stack error was detected
 */
int32_t NI_GetProtocol(void);

```

This function either times out or gives an enumeration as dictated by the protocol used. [Table 8](#) lists the possible returns and their corresponding protocol code.

Table 8. Protocol Codes

Code	Protocol
2	PROFINET
3	EtherNet/IP
4	Modbus/TCP
7	EtherCAT
8	POWERLINK

RPG2 UNIFIED INTERFACE USER GUIDE

NI_SetDevice()

The **NI_SetDevice()** function sets the device that exists from the module configuration data. This function is not valid after **NI_ConfigComplete()** is called.

NI_SetDevice() Function Prototype

The prototype of this function follows:

```

/* Set the system device according to the indicated ID
 *
 * deviceId ID of the device to be set for the system
 *
 * Returns NI_OK on success
 * NI_ERROR if device not found in IO config data
 * NI_ERROR if called after configuration completed
 * NI_TIMEOUT if operation timed out
 */
int32_t NI_SetDevice(uint16_t deviceId);

```

NI_AddItem()

The **NI_AddItem()** function adds an item from the configuration database of the user. This function can be valid after **NI_Config Complete** depending on the protocol that is in use (see [Table 9](#)). This function also has no use in the IP use case.

Table 9. Validity After NI_ConfigComplete for NI_AddItem

Protocol	Valid
PROFINET	Yes
EtherNet/IP	No
Modbus/TCP	No
EtherCAT	No
POWERLINK	No

NI_AddItem() Function Prototype

The prototype of this function follows:

```

/* Add an item to the system
 *
 * itemId ID of the item to be added to the system
 * location into which the item should be installed
 *
 * Returns >= 0: item handle
 * <0: NI_INVALID_PARAM if location out of range (must be >0)
 * NI_ERROR if item not found by ID
 * NI_ERROR if items not currently permitted to be added
 * NI_TIMEOUT if operation timed out
 */
int32_t NI_AddItem(uint16_t itemId, uint32_t location);

```

RPG2 UNIFIED INTERFACE USER GUIDE

NI_ConfigComplete()

The **NI_ConfigComplete()** function is called to indicate that the application processor has no more options to configure the communication controller and is ready for Ethernet communication. After this function is called, some functions become available to the application processor and some cease being available.

NI_ConfigComplete() Function Prototype

The prototype of this function follows:

```
/* Indicate that configuration should be completed
 *
 * No parameters
 *
 * Returns NI_OK on success
 * NI_ERROR if there is a problem with the system
 * NI_TIMEOUT if operation timed out
 */
int32_t NI_CompleteConfig(void);
```

NI_GetOutputData()

The **NI_GetOutputData()** function obtains the newly received output data from the Unified Interface stack. Call this function in response to the **NI_OutputDataHandler()** function.

NI_GetOutputData() Function Prototype

The prototype of this function follows:

```
/* Get the newly-received output data
 *
 * Use this function to retrieve the newly-received output data from the
 * Unified Interface stack.
 *
 * This function should be called as a response to the notification of new
 * output data. It should either be called from or immediately after
 * NI_OutputDataHandler() is called. It should never be called otherwise.
 *
 * dataStatus_p Pointer to location into which the status of the item
 * output data should be copied (see NI_ioStatus_t)
 * itemHandle_p Pointer to location into which the item handle to
 * which the output data corresponds should be copied
 * outDataSize_p Location into which the size (in bytes) of the received
 * output data should be copied
 * outData_p Pointer to location into which the received output data
 * should be copied
 *
 * Returns NI_OK on success retrieving output data
 * NI_ERROR if there was a problem parsing the received output data
 * message
 * NI_TIMEOUT if operation timed out
 */
int32_t NI_GetOutputData(int32_t *dataStatus_p,
int32_t *itemHandle_p,
uint16_t *outDataSize_p,
```

RPG2 UNIFIED INTERFACE USER GUIDE

```
uint8_t *outData_p);
```

NI_SetInputData()

The **NI_SetInputData()** function sets the input data that is sent to the industrial Ethernet network.

NI_SetInputData() Function Prototype

The prototype of this function follows:

```
/* Set the most-recent input data
 *
 * Use this function to push the latest input data into the NI API for
 * transmission to the Network Interface module. This function should be called
 * in either of the following scenarios:
 *
 * A) At the application side's discretion in order to update the input data
 * supplied on the network
 * B) From NI_InputDataLatchHandler() in order to complete an isochronous
 * network cycle
 *
 * itemHandle Handle to the item for which the input data is being supplied
 * inDataSize Size (in bytes) of the supplied input data
 * inData_p Pointer to the input data to be supplied for the indicated
 * item
 * complete Flag indicating whether or not the write of the input data to
 * the Network Interface module is complete
 *
 * NOTE: Use NI_ALL_ITEMS as the item handle parameter value if the supplied
 * input data is for all installed items and is already packed.
 *
 * NOTE: Setting the 'complete' flag will cause the NI API to notify the
 * Network Interface module that this input cycle is complete. Clearing
 * this flag will simply tell provide the input data to the Network
 * Interface module. If NI_ALL_ITEMS is supplied as the item handle,
 * the NI API will automatically tell the Network Interface module that
 * this input cycle is complete (the 'complete' flag is ignored).
 *
 * Returns NI_OK on success setting input data
 * NI_ERROR if there was a problem creating the input data message
 * NI_TIMEOUT if operation timed out
 */
int32_t NI_SetInputData(int32_t itemHandle,
uint16_t inDataSize,
uint8_t *inData_p,
uint8_t complete);
```

NI_TimeUpdate()

Use this **NI_TimeUpdate()** function to get the time the function was last called. Users can call this several times in their code to monitor the timing between different parts of their application.

RPG2 UNIFIED INTERFACE USER GUIDE

NI_TimeUpdate() Prototype

The prototype of this function follows:

```
/* Notify this library that time has passed
 *
 * Use this function to tick this library and let it know how much time has
 * passed since the last call. This allows it to monitor things like response
 * message arrivals and the associated timeouts.
 *
 * NOTE: The precision of response timeouts is subject to how often this
 * function is called. The more often this function is called, the more
 * precise timeouts will be - at the cost of increased overhead. The
 * less often this function is called, the less precise timeouts will
 * be but at the benefit of decreased overhead.
 *
 * elapsedUs The number of microseconds that have passed since the last
 * call to this function
 *
 * Returns nothing
 */
void NI_TimeUpdate(uint32_t elapsedUs);
```

NI_SetResponseTimeout()

The **NI_SetResponseTimeout()** function sets the time duration that is acceptable between an API call and its corresponding response. This function gives users the ability to set parameters for acceptable delays in communication between the application processor and the communication processor.

NI_SetResponseTimeout() Function Prototype

The prototype of this function follows:

```
/* Set the timeout time for expected response messages
 *
 * Use this function to override how long the application will wait before
 * timing a response out and considering it lost. This setting applies to all
 * responses.
 *
 * timeout_us Timeout duration (in us)
 *
 * Returns NI_OK on success setting timeout value
 * NI_INVALID_PARAM if timeout duration invalid
 */
int32_t NI_SetRespTimeout(uint32_t timeout_us);
```

NI_OutputDataHandler()

The **NI_OutputDataHandler()** function serves as a notification to the application processor that output data has arrived from the network. The user must call **NI_GetOutputData** as a response to this function. This call can be done right away or when the users must decouple their inputs and outputs.

RPG2 UNIFIED INTERFACE USER GUIDE**NI_OutputDataHandler() Function Prototype**

The prototype of this function follows:

```

/* Application output data handler
*
* This function is implemented by the application and called the Network
* Interface API service. It is called immediately after new output data from
* the module has arrived. It serves as a notification that NI_GetOutputData()
* should be called. NI_GetOutputData() MUST be called once for every call to
* this function. If the output data and associated metadata is not of interest
* to the output data handler, all parameters to NI_GetOutputData() can be set
* to NULL.
*
* It is permissible to call NI_GetOutputData() from this function or after
* execution of this function completes.
*
* The application can handle this data however it would like. Some typical
* actions might be:
* - Scenario 1: A) Store the output data to be applied later (along with
* supplying input data)
* - Scenario 2: A) Apply the output data to the physical outputs
* B) Read the physical inputs to latch the input data
* C) Push the latched input data into the NI API to be
* transmitted to the Network Interface module
*
* NOTE: Unless the NI API was configured to use a polling approach, this
* function will be called from the same context (i.e. ISR or non-ISR)
* as the LPL function that called the Unified Interface stack.
*
* NOTE: If no items were added to the system that contain output data, this
* function will never be called by the NI API.
*
* No parameters
*
* Returns nothing
*/
void NI_OutputDataHandler(void);

```

NI_InputDataLatchHandler()

The **NI_InputDataLatchHandler()** function is called directly after the network interface module has indicated that the current input data must be latched. The **NI_SetInputData()** function must be called to write new input data to the network interface module and to complete the I/O cycle.

NI_InputDataLatchHandler() Function Prototype

The prototype of this function follows:

```

/* Application input data latch handler
*
* This function is implemented by the application and called by the Network
* Interface API service. It is called immediately after the NI module has
* indicated that the input data should be latched. It serves as a notification
* that NI_SetInputData() should be called in order to write fresh input data
* to the NI module and complete the current network IO cycle.

```

RPG2 UNIFIED INTERFACE USER GUIDE

```

*
* No parameters
*
* Returns nothing
*/
void NI_InputDataLatchHandler(void);

```

NI_SetTransmitModes()

The **NI_SetTransmitModes()** function sets the input data and output data transmit modes. Supported output data transmit modes are on request, on time (that is, periodically), on change, or on network (that is, every network cycle). Supported input data transmit modes are on time (that is, periodically) or on network (that is, every network cycle).

NI_SetTransmitModes() Function Prototype

The prototype of this function follows:

```

/* Set the input and output data transmit modes
*
* Use this function to set what triggers transmission of output data from the
* NI module and what triggers transmission of input data to the NI module.
*
* Output data may be transmitted 'on request', 'on time' (i.e. periodically),
* 'on change' or 'on network' (i.e. every network cycle). This setting affects
* the trigger that causes the NI module to transmit messages that contain
* output data. If 'on request' is selected, the application must send a
* message requesting that the NI module send output data. The timing of this
* request is the responsibility of the application. If 'on time' is selected,
* the NI module will send output data periodically with no respect to network
* cycle. If 'on change' is selected, the NI module will send output data in
* synchronization with the network cycle but only when the data has changed.
* If 'on network' is selected, the NI module will send output data every
* network cycle in synchronization with the network cycle.
*
* Input data may be transmitted 'on time' (i.e. periodically) 'on network'
* (every network cycle). This setting affects the trigger that causes
* NI_InputDataLatchHandler() to be called. If 'on time' is selected,
* NI_InputDataLatchHandler() will be called periodically based on the local
* time and the supplied input data transmit period parameter. If 'on network'
* is selected, NI_InputDataLatchHandler() will be called as a follow on action
* to a received 'latch inputs' event message.
*
* NOTE: If an isochronous application is desired, the selected output data
* transmit mode must be either 'on change' or 'on network'. The other
* modes are not compatible with isochronous operation.
*
* NOTE: If an isochronous application is desired, the selected input data
* transmit mode must be 'on network'.
*
* NOTE: The supported transmit modes for output data are
* - 'on request'
* - 'on time'
* - 'on change'
* - 'on network'
*

```


RPG2 UNIFIED INTERFACE USER GUIDE

```

* NOTE: The supported transmit modes for input data are
* - 'on time'
* - 'on network'
*
* outTxMode The output data transmit mode for which the NI module should
* be configured
* outTxPeriod_us The period (in us, subject to NI system tick rounding) at
* which output data should be transmitted (only valid if
* 'on time' transmit mode is selected, ignored for all others)
* inTxMode The input data transmit mode for which the local system
* should be configured
* inTxPeriod_us The period (in us, subject to local system time update
* rounding) at which input data should be transmitted from the
* local system (only valid if 'on time' transmit mode is
* selected, ignored for all others)
*
*
* Return NI_OK on success setting transmit modes
* NI_INVALID_PARAM on unrecognized transmit mode
* NI_INVALID_PARAM if selected transmit mode is invalid for a given
* data direction (e.g. if selected input data transmit
* mode is 'on change')
* NI_INVALID_PARAM if invalid input or output data transmit period
* (only returned if selected transmit mode is
* 'on time')
* NI_TIMEOUT if operation timed out
* NI_ERROR if NI module reported an error while setting transmit mode
*/
int32_t NI_SetTransmitModes(NI_transmitMode_t outTxMode,
uint32_t outTxPeriod_us,
NI_transmitMode_t inTxMode,
uint32_t inTxPeriod_us);

```

NI_GetBasket()

The **NI_GetBasket()** function gets a basket from the configuration database of the user. This function can be valid after **NI_Config Complete** depending on the protocol that is in use.

Table 10. Validity After NI_ConfigComplete for NI_GetBasket()

Protocol	Valid
PROFINET	Yes
EtherNet/IP	No
Modbus/TCP	No
EtherCAT	No
POWERLINK	No

NI_GetBasket() Function Prototype

The prototype of this function follows:

```

/* Get a basket from the system
*
* basketId ID of the basket to be retrieved
* basket_pp Pointer to location into which the pointer to the retrieved

```

RPG2 UNIFIED INTERFACE USER GUIDE

```

* basket data should be copied (out parameter)
*
* Returns NI_OK on success (basket_pp updated)
* NI_ERROR if basket not found (basket_pp not updated)
*/
int32_t NI_GetBasket(uint16_t basketId, NI_basket_t **basket_pp);

```

NI_ProcessEvents()

Use the **NI_ProcessEvents()** function to check if an event occurred and then call the proper event handler function to notify the application.

NI_ProcessEvents() Function Prototype

The prototype of this function follows:

```

/* Process events
*
* Process any events that have come in since the last call to this function
* (or since the beginning of time).
*
* This function will check if any events have been occurred and then call the
* appropriate event handler to notify the application of the event and its
* information.
*
* In order to expedite event handling, this function should be called
* repeatedly and as often as possible.
*
* No parameters
*
* Returns nothing
*/
void NI_ProcessEvents(void);

```

NI_MsgReadyCallback()

The **NI_MsgReadyCallback()** function is a pointer to the function the stack must call when a new message arrives and is ready for parsing. When a new message is received and queued for parsing by the application, the application implements this function and calls the function by the stack.

NI_MsgReadyCallback() Function Prototype

The prototype of this function follows:

```

/* Unified Interface message ready handler
*
* During Unified Interface stack initialization, the application should point
* the stack at this function to handle received messages
*
* This library will manage waiting for the required response on behalf of the
* application
*
* NOTE: Conforms to the requirements set by the Unified Interface stack in
* UI_common.h. Parameters and return values are not discussed here.

```

RPG2 UNIFIED INTERFACE USER GUIDE

```
*/  
UI_MsgReady_t NI_MsgReadyCallback;
```

APPLICATION PROCESSOR LINK TYPE

The application processor is the system processor where users develop and create their industrial Ethernet device. The customer interfaces the application processor to the [ADSP-CM409F](#) processor that exists as part of the RPG2 solution. The link type is the hardware connections for the application processor to the [ADSP-CM409F](#).

The following are the link types that are supported by the link porting layer:

- ▶ Ethernet
- ▶ UART
- ▶ SPI

In the software zip files found within the [RPG2 Unified Interface User Guide](#) section, there are link porting layers that are available as examples for a variety of platforms.

Link Type Selection

Board Configuration File Method

There are two methods by which the link type is selected. The first method is to allow the user to determine this by loading a different board configuration file. A board configuration file pre-exists for each Unified Interface link type in the software zip file download from the [main product page](#) as well as one for the strapping option method. The board configuration file overrides what is physically strapped for the design.

The user may completely bypass the board configuration file if they strap the link type selector pins (LT1, LT2, LT3 on the ADIN2299) and do not load a contradictory Link Configuration File. The contents of the file system are blank upon arrival, therefore, if a user does not add a link configuration file it will follow what is on the pins which by default is Ethernet.

Strapping Option Method

A user can also elect to utilize the three strapping pins to select an application processor interface type. This method is over-written by the board configuration file if there is a conflict.

Ethernet

Unified Interface messages can be sent via Ethernet. In this case, the Unified Interface message follows a standard Ethernet header. The Ethernet link type can be either direct or indirect. A direct Ethernet link consists of two directly connected reduced media independent interfaces (RMIs), one on the application side and one on the communication side. An indirect Ethernet link consists of two RMIs connected to each other through a physical layer (PHY) on either end with an Ethernet cable in between.

The Unified Interface is agnostic to direct vs. indirect connections.

For Ethernet signal connections, refer to the [Link Configuration](#) section of this document. In addition, the message format is handled automatically by the Unified Interface stack. Forming up these messages is done automatically by the interaction between the link porting layer and the stack. Therefore, this section is for informational purposes only.

Transaction Details

Unified Interface messages are encapsulated in a standard Ethernet frame, immediately following the EtherType.

This frame keeps the application processor software lighter weight (no need for a TCP/IP stack) and obviates the need for supporting protocols like address resolution protocol (ARP) on the communication side, which is required in most cases if a higher level transport protocol was selected.

Essentially, Unified Interface messages are transmitted using raw sockets.

RPG2 UNIFIED INTERFACE USER GUIDE

Virtual local area network (VLAN) tagged frames are ignored when received from the application side and are not transmitted from the communication side. [Table 11](#) shows the format of the Ethernet message.

Table 11. Ethernet Frame Format

Frame Byte	Value
0	Destination MAC Address Byte 1
1	Destination MAC Address Byte 2
2	Destination MAC Address Byte 3
3	Destination MAC Address Byte 4
4	Destination MAC Address Byte 5
5	Destination MAC Address Byte 6
6	Source MAC Address Byte 1
7	Source MAC Address Byte 2
8	Source MAC Address Byte 3
9	Source MAC Address Byte 4
10	Source MAC Address Byte 5
11	Source MAC Address Byte 6
12	Most significant byte of the ether type
13	Least significant byte of the ether type
14 Through n	Unified Interface data

UART

Unified Interface messages can be sent or received over a UART.

In addition, the message format is handled automatically by the Unified Interface stack. Forming these messages is done automatically by the interaction between the link porting layer and the stack. Therefore, this section is for informational purposes only.

Transaction Details

Each byte of a Unified Interface message is framed with an optional parity bit and the configured number of stop bits.

Table 12. UART Transaction

Frame Field	Start Bit	Message Byte	Parity Bit	Stop Bit(s)
Bit Length	1	8	0 to 1	1 to 2

The baud rate, parity setting (none, even or odd), and number of stop bits (1 or 2) is configured via the board configuration data.

In each transmission via the UART, there are as many UART frames as there are message bytes. For example, a 6 byte Unified Interface message causes six UART frames to transmit.

Message bytes transmit LSB first.

Because a UART is not a leader and follower interface or half duplexed, there is no need for either side to request permission from the other side to transmit or check that the other side is currently transmitting.

Because a UART is full duplexed, it is permissible for two Unified Interface messages to be on the wire simultaneously.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

GENERAL DESCRIPTION

The RapID Platform Generation 2 (RPG2) input/output (I/O) configuration tool is used to generate information that describes the input and output footprint of the final product data of the user on an industrial automation network. The configuration tool allows the user to enter the desired configuration values and export a file that can then be loaded into the product so the product can set up the input and output as the input values, output values, and device parameters appear on the network. Although data in the files produced by the RPG2 I/O configuration tool is specific to the network protocol in use (such as PROFINET®, Ethernet/internet protocol (IP), and Modbus/transmission control protocol (TCP)), the data is created and used in such a way that the host system does not need to understand how to create messages as needed for each network protocol.

After entering the information necessary to create the desired input and output configuration into the RPG2 I/O configuration tool, the tool exports a configuration file containing this information in a format understood by the software of the product of the user. If the final product supports multiple network protocols, the tool can create configurations for each protocol. After a configuration file exports and loads into the nonvolatile product memory, the product is ready to start network communications.

The process of creating and loading a configuration often occurs as follows:

- ▶ Use this tool to define one or more devices in the device library. See the [Devices](#) section for more information.
- ▶ Use this tool to define one or more items in the item library. See the [Items](#) section for more information.
- ▶ Optionally, users can place a device from the device library and one or more items from the item library into a basket. More than one basket can be created. Baskets can also logically group a single device in the device library with one or more items in the item library. See the [Baskets](#) section for more information.
- ▶ Export selected devices, items, and baskets to a configuration file. See the [Configuration File Formats](#) section for more information. Note that newly exported configuration files load into the product by using either a Joint Action Test Group (JTAG) interface or the web server on the RPG2 module/embedded reference design.

RPG2 I/O CONFIGURATION TOOL FUNCTIONALITY AND INSTALLATION

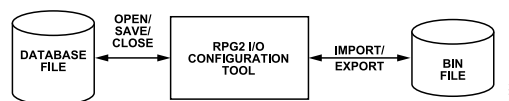
Functionality

Any configuration information saved using the RPG2 I/O configuration tool saves in a database file, such as the .rpc file. This database file allows the user to keep all configuration information created by the user in one central location, even if the information is for more than one protocol. After populating the database, it is possible to export selected database information or all of the database information as a configuration file in binary (bin) format. Similarly, it is possible to import configuration files that are in binary format into the user database. [Figure 1](#) shows the ability of the configuration tool to open, save, and close a database file, as well as to import or export a configuration file.

Installation

To install the RPG2 I/O configuration tool, copy the configuration tool executables to a directory on the host computer. Once the executables are copied, the RPG2 I/O configuration tool is ready to run.

The **ConfigurationTool.exe** file is stored on the supplied USB drive or this file can be downloaded in .zip format from the **Tools Files** directory from the ADIN2299 main product page at www.analog.com/adin2299. Copy this file to a convenient working directory on the host computer.



CONFIGURATION OBJECTS

The database created and stored by the RPG2 I/O configuration tool contains devices, items, and baskets. Each object type can be used for a different purpose, but a 16-bit ID uniquely identifies each object type. The ID of each object is user selectable, but the ID must be unique within the object type and protocol. For example, two Ethernet/IP devices cannot have an ID of 120. The RPG2 I/O configuration tool keeps track of each object, its ID, and the protocol under which the object falls, thereby keeping objects with the same ID but different protocol association separated from each other. For example, this tracking allows the user to have an Ethernet/IP device with an ID of 100 and a Modbus/TCP device with an ID of 100 in the same database.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Devices

Use device objects to define the overall device. Although the data in a device object is dependent on the protocol in use, the device contains an ID, name, and several other communication parameters. Details on these parameters, as well as the difference based on protocol, are in the following sections.

PROFINET Devices

The PROFINET device configuration consists of elements related to the general station description markup language (GSDML) file, the identification and maintenance (I&M) interface, and simple network management protocol (SNMP) descriptions. [Table 13](#) details the PROFINET devices.

Ethernet/IP Devices

The Ethernet/IP device object contains data identity data (used by the Ethernet/IP identity object), default factory data (used by the Ethernet/IP and TCP/IP object), and system options (see [Table 14](#)). Refer to the *ODVA Ethernet/IP Specification Volume 2* for details concerning specific objects and attributes.

Most of the remaining device data (from static IP address to address conflict detection (ACD) enable) is used when a Type 1 reset is requested by a controller, as noted in the *ODVA Ethernet/IP Specification Volume 1*. The Type 1 reset causes the system to discard all prior nonvolatile communications settings and reverts to the settings defined in this user guide. The reset provides a way to return the system to the factory configuration and is required for conformance to Ethernet/IP specifications.

Table 13. PROFINET Devices

Device Name	Description
PROFINET Vendor ID	Same as GSDML file
PROFINET Device ID	Same as GSDML file
PROFINET Profile ID	NonGSDML item, PROFINET specific profile ID, use 0 for nonprofile device
PROFINET Profile Type	NonGSDML item, PROFINET specific profile ID, use 3 for input and output module
Device Type	ProductFamily parameter in GSDML file, 240 characters maximum
Order ID	OrderNumber parameter in GSDML file, 20 characters maximum
Module ID	(DeviceAccessPointItem or ModuleIdentNumber parameter in GSDML file
Submodule ID	(VirtualSubmoduleItem) or SubmoduleIdentNumber parameter in GSDML file
Software Major Revision	SoftwareRelease parameter in GSDML file
Software Minor Revision	SoftwareRelease parameter in GSDML file
Software Internal Revision	SoftwareRelease parameter in GSDML file
Revision Counter	NonGSDML item, necessary for PROFINET stack
SNMP System Description	NonGSDML item, 256 characters maximum
SNMP Enterprise ID	NonGSDML item, private SNMP object identifier (OID)
Internal Interface Description	NonGSDML item, 256 characters maximum
Port 1 Interface Description	NonGSDML item, 256 characters maximum
Port 2 Interface Description	NonGSDML item, 256 characters maximum

Table 14. Ethernet/IP Devices

Device Name	Description
Vendor ID	Ethernet/IP identity object, Attribute 1
Device ID ¹	Ethernet/IP identity object, Attribute 2
Product Code ¹	Ethernet/IP identity object, Attribute 3
Major Revision ¹	Ethernet/IP identity object, part of Attribute 4
Minor Revision ¹	Ethernet/IP identity object, part of Attribute 4
Product Name ¹	Ethernet/IP identity object, Attribute 7
Static IP Address	Part of Attribute 5
Subnet Mask ²	Part of Attribute 5
Gateway Address ²	Part of Attribute 5

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Table 14. Ethernet/IP Devices (Continued)

Device Name	Description
Primary and Secondary Domain Name System (DNS) Server IP Addresses ²	Part of Attribute 5
Dynamic Host Configuration Protocol (DHCP) Enable Flag ²	Part of Attribute 3
Domain Name String ²	Part of Attribute 5
Host Name String ²	Attribute 6
Multicast Time to Live (TTL) Value ²	Attribute 8
Multicast Allocation Control ²	Part of Attribute 9
Number of Multicast Addresses ²	Part of Attribute 9
Multicast Start Address ²	Part of Attribute 9
ACD Enable ²	Attribute 10
Link Speed	Part of Attribute 6; used by Ethernet/IP Ethernet link object, one per port
Link Duplex	Part of Attribute 6; used by Ethernet/IP Ethernet link object, one per port
Link Autonegotiation Enable Flag	Part of Attribute 6; used by Ethernet/IP Ethernet link object, one per port
Ethernet Link Administrative State Flag	Attribute 9; used by Ethernet/IP Ethernet link object, one per port
Enable Device Level Ring (DLR) ³	Enables system DLR functionality

¹ Unless otherwise specified, these devices are the source for the attributes of the Ethernet/IP identity object.

² Unless otherwise specified, these devices are the source for the attribute values of the Ethernet/IP TCP/IP interface object.

³ When cleared, the resulting devices does not support DLR, nor does it expose the DLR common industrial protocol (CIP) object. As such, the system designer can decide whether the end device supports DLR. If the flag is set, the DLR capability enables, and the DLR CIP object is exposed. The type of DLR used depends on the hardware of the module. If the [fido2100](#) DLR Ethernet switch is used, DLR implementation is beacon based. If the Micrel Ethernet switch based module is used, the DLR is announced based.

Modbus/TCP Device

The parameters detailed in [Table 15](#) define the Modbus/TCP device.

Table 15. Modbus/TCP Device Parameters

Parameter Name	Description
Follower ID	Single byte that is reported in the report follower ID response packet.
Follower ID Additional Data	Byte string (30 bytes maximum) that forms the additional information section in the report follower ID response packet.
Order ID	Not used.
Support Email Address	Not used.
Manufacturer Website Address	Not used.
TCP Keepalive Period	Number of milliseconds between keepalive pings. The Modbus/TCP product periodically pings all of the connected clients. If a client does not respond to two consecutive pings at any time after the TCP connection is established, the product declares the connection dead, and the client connection is removed. If the keepalive timeout is 1000, and a client does not respond to any keepalive pings, the connection is declared dead after 3000 ms and removed. A keepalive value of 0 disables the keepalive timer. However, disabling this way is not recommended.
Watchdog Period	Minimum activity period (in ms). If the output data of the device is not updated within the activity period, the outputs are reset to 0, and the common interface input and output status is set to CI_IO_INVALID. Setting the activity watchdog value to 0 disables the activity watchdog.
Cycle Period	Number of milliseconds between updates to the network. The recommended cycle period is 100 ms.
Major Revision	Major revision value for product.
Minor Revision	Minor revision value for product.

EtherCAT Devices

An EtherCAT device supplies information (such as the device name) to the corresponding CANopen over EtherCAT (CoE) objects. EtherCAT devices allow the user to customize the device and insert unique information, such as the vendor ID and product code.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Table 16. EtherCAT Devices

Device Name	Description
Vendor ID	32-bit EtherCAT vendor ID placed in Object 0x1018. Contact the EtherCAT Technology Group (ETG) for information on how to obtain an EtherCAT vendor ID.
Product Code	32-bit EtherCAT product code placed in Object 0x1018 that uniquely identifies the user device to the leader among others.
Device Name	32-character string placed in Object 0x1008 that provides the model number or a brief description of the device.
Hardware Version	8-character string placed in Object 0x1009, which is the version of the hardware this device is using.
Software Version	8-character string placed in Object 0x100A, which is the software version this device is using.
Major Revision	16-bit revision number concatenated with the minor revision and placed in Object 0x1018, which is the major revision number of the device.
Minor Revision	16-bit revision number concatenated with the major revision and placed in Object 0x1018, which is the minor revision number of the device.

POWERLINK Devices

A POWERLINK device supplies information such as the device name to the corresponding CoE objects. POWERLINK devices allow the user to customize their device and insert information such as the vendor ID and product code.

Table 17. POWERLINK Devices

Device Name	Description
Vendor ID	32-bit POWERLINK vendor ID placed in Object 0x1018. Contact the Ethernet POWERLINK Standardization Group (EPSG) for information on how to obtain a POWERLINK vendor ID.
Product Code	32-bit POWERLINK product code placed in Object 0x1018 that uniquely identifies the user device to the leader.
Revision Number	32-bit integer.
Serial Number	32-bit integer.
Device Name	32-character string placed in Object 0x1008 that provides the model number or a brief description of the device.
Hardware Version	8-character string placed in Object 0x1009, which is the version of the hardware this device is using.
Software Version	8-character string placed in Object 0x100A, which is the software version this device is using.

Items

Items define the overall input and output sizes or types of the product. Like the device object, the data this object contains varies based on protocol.

PROFINET Items

The two distinct PROFINET item types are cyclic and acyclic.

Cyclic PROFINET Items

Cyclic items specify the input and output modules, either virtual or physical, plugged into a given device. Cyclic items contain the parameters detailed in [Table 18](#). These elements correspond to the description of a module in the GSDML file and can evaluate the expected configuration sent by the PROFINET controller to the device during the connection process. When adding cyclic items at run time, the location parameter is used as the slot. Cyclic items must be added in the order of their slot numbers.

A device with a fixed input and output can be built with a single cyclic item to represent all cyclic inputs and outputs. A more complex device with multiple possible configurations can be modeled with multiple cyclic items. For example, a modular input and output device can have a separate cyclic item type for each type of module installed in the device. Multiple copies of a given item type can be installed by the user application.

Acyclic PROFINET Items

Acyclic PROFINET items link other kinds of data that are not part of the cyclic data flow to the network. The configuration of each acyclic items contains the fields detailed in [Table 19](#).

In addition, slot number is an important parameter consideration for acyclic items. The location parameter must correspond to the desired slot number when creating a basket or add the item by calling **CI_AddItem()**. Acyclic items can be allocated to the same slot number as

RPG2 I/O CONFIGURATION TOOL USER GUIDE

cyclic items, allowing parameterization data and diagnostic data to allocate for a module. Similarly, multiple acyclic items (indices/sizes) can be associated with a single slot number.

Ethernet/IP Items

Ethernet/IP items specify the details of the assembly object instances exposed by the product. An item can reference input data only, output data only, or both. There are three distinct types of cyclic Ethernet/IP items: unique, incrementing, and accumulating. There are also two types of acyclic Ethernet/IP items: configuration and diagnostic. Items of different types are defined by the configuration data installed into the system and are primarily distinguished by what happens when an item is added to the system or when more than one item of a given ID is added to the system.

All Ethernet/IP items contain the parameters detailed in [Table 20](#). Ethernet/IP items can be created to support all connection types identified in the specifications. A maximum of 32 Ethernet/IP items can be added to the system. There can be no more than 16 Ethernet/IP assembly instances created. Although there are more Ethernet/IP items available than assembly instances, it is possible to run out of assembly instances before Ethernet/IP items because the item type determines whether the item creates a new assembly instance or accumulates input and output data into another existing item.

Table 18. Cyclic PROFINET Parameters

Name	Description
Item Module ID	Module ID associated with item
Item Submodule ID	Submodule ID associated with item
Number of Input Bytes	Number of bytes available to the controller to read the item
Number of Output Bytes	Number of bytes available to the controller to write the item

Table 19. Acyclic PROFINET Parameters

Field	Description
Index	Used by the controller to address data (0x0001 through 0xFFFF)
Input Bytes	Number of bytes available to the controller to read the index
Output Bytes	Number of bytes available to the controller to write the index
Slot Number	Used by the controller to access data

Table 20. Ethernet/IP Parameters

Name	Description
Item Type	Type of item (unique, incrementing, accumulating, or configuration)
Consume Assembly Instance ID	Originator to target assembly instance ID
Consumption Size	Base number of consumed bytes
Production Instance ID	Target to originator assembly instance ID
Production Size	Base number of produced bytes

Unique Ethernet/IP Items

When adding a unique item, the system searches the item table for any existing entries with this type and item ID. If any existing entries are found, an error returns and a new item does not create. If no matching item is found, the new item is allocated and added to the list. Searching the item table ensures that an item and its associated assembly instance IDs are unique within the system. When data is written to unique items, the data is immediately relayed to the Ethernet/IP stack for transmission over the network.

Incrementing Ethernet/IP Items

When adding an incrementing item, the system searches the item table for the last entry of this type and its item ID. The system then attempts to allocate a new item with the sequential assembly instance IDs. Both the produce instance IDs and consume instance IDs are incremented. After incrementing the instance IDs, the table searches again, this time searching for an item corresponding to either of the new assembly instance IDs. If a match is found, an error returns. If a match is not found, the new item is allocated and added to the list. As such, a new assembly object with the size defined by the input and output configuration data is created. However, this new assembly object receives a new and unique set of assembly instance IDs. When writing data to incrementing items, the data relays immediately to the Ethernet/IP stack for transmission over the network.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Accumulating Ethernet/IP Items

When adding an accumulating item, the system searches the item table for the last entry of this type and its item ID. The system then attempts to allocate a new item with the same assembly instance IDs and the same input and output size. The system uses an increasing offset in the expanding input and output data based on the item size. If adding this new item causes the accumulated input and output data size to exceed the allowed maximum for a single assembly object (500 bytes), an error returns and the new item is not added. If the size is not exceeded, a new item that references the same assembly instances but with the input and output data is placed at the next offset and is added to the list. As such, a single assembly instance object is created with the data size accumulating as each new item is added. When writing data to items of this type, the data is not relayed to the Ethernet/IP stack until all data input and output is complete.

Configuration Ethernet/IP Items

Add a configuration item to support a CIP configuration assembly. This item only contains the consume instance ID and size and is added to the device so that the system accepts the configuration data provided by the Ethernet/IP programmable logic controller (PLC) at the time the forward open issues. Users do not typically establish a Class 1 connection to this item, and as such, configuration items are defined as acyclic data items. The consume assembly instance ID must be unique for this item. Do not add this item more than once to the system or more than once to a given basket.

Diagnostic Ethernet/IP Items

Adding a diagnostic item provides a produce assembly that can send system health information to the network. This item only contains a produce instance ID and size. It is possible to establish a Class 1 connection to this item, but typically, diagnostic items are used with explicit messaging in an acyclic fashion. The assembly instance ID for this item must be unique in the system. Multiple diagnostic items can be added to the system or used in the same basket, assuming each has a distinct ID.

Modbus/TCP Items

Use Modbus/TCP items to map the input and output data of the product to discrete inputs, coils, holding registers, and input registers in the Modbus/TCP register and bit space.

To accommodate systems with different endianness needs or systems that store or transfer up to 64-bit values in consecutive registers, 32-bit and 64-bit swapping is available. Take care when swapping because the input and output data size must be an integer multiple of 4 for 32-bit swapping, and the input and output data size must be an integer multiple of 8 for 64-bit swapping. Never swap bit data.

The RPG2 I/O configuration tool allows input and output data to be mapped virtually anywhere in the bit or register space. Accomplish mapping by setting the register address mapping and the bit address mapping to the desired values when creating an item. When a contiguous Modbus/TCP bit and register space is required, the RPG2 I/O configuration tool can place items in auto mode. When an item is set to auto mode, the address for the Modbus/TCP register that corresponds to the input and output data calculates automatically at run time. If all of the items are set to auto mode, the data for the first item maps to Address 0. All items added subsequently map to an address that sequentially follows the item added before it so that the register and bit space can appear gapless and linear.

On an item by item basis, holding and input registers can be mirrored together. When mirroring registers together, data written to the holding registers from the network automatically write to the input registers with the same addresses immediately after the writes to the holding register completes. In addition, when writing items from the host side, the data writes to the input registers as well as the holding registers. As such, the Modbus/TCP controller can write to both holding and input register spaces, and the host can write to both holding and input register spaces. Take care during this process because items that are register only in auto mode only increment addresses in the register space. Likewise, bit only items in auto mode only increment addresses in the bit space. When adding items to the system, add the items in ascending register and bit address order unless auto addressing mode is in use. Adding items out of ascending order results in an error when attempting the addition.

Modbus/TCP items contain the fields detailed in [Table 21](#).

Table 21. Modbus/TCP Item Parameters

Field Name	Description
Swap Mode	Selects swapping mode (none, 16-bit, 32-bit, or 64-bit).
Register Space Mapping	Selects whether register space mapping enables for this item. If mapping enables, this item also selects auto addressing mode or use selected.
Bit Space Mapping	Selects whether bit space mapping enables for this item. If mapping enables, this item also selects auto addressing mode or use selected.
Input Size	Number of input bytes to map to the Modbus/TCP register or bit space. Enter 0 if no input data is required for this item.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Table 21. Modbus/TCP Item Parameters (Continued)

Field Name	Description
Output Size	Number of output bytes to map to the Modbus/TCP register or bit space. Enter 0 if no output data is required for this item.
Register Offset	Specifies the beginning register space address where the item data must be mapped if register space addressing mode is set to use selected.
Bit Offset	Specifies the beginning bit space address where the item data must be mapped if bit space addressing mode is set to use selected.
Mirror Registers	Mirrors the input and holding registers represented by this item together.

EtherCAT Items

Use EtherCAT items to create objects (such as transmission process data objects (TxPDOs), receiving project data objects (RxPDOs), input entries, and output entries) that represent product data on the EtherCAT network. The three types of EtherCAT items are process data, configuration, and diagnostic.

A process data item is a cyclic item that contains the process data the product consumes and produces on the EtherCAT network. The configuration and diagnostic items are acyclic. These items contain data that the EtherCAT leader can send to the product during system startup to set parameters, or data that the leader can read from the device to determine what faults the product application must indicate. For any item, the total size of the input and output data (the sum of the sizes of the input and output subindices) is displayed in the item size summary field (see [Figure 26](#) to [Figure 28](#)).

Process Data Items

A process data item creates a RxPDO (Index 0x16xx) or TxPDO (Index 0x1Axx), along with input and output entries (Index 0x6xxx and Index 0x7xxx). The exact index numbers of the TxPDOs, RxPDOs, and input and output entries depends on the location where the process data item installs (see [Table 22](#)). When adding the item to the system, select the location of the item.

The created RxPDO or TxPDO automatically populate with the input and output entry information when the RapID system starts. As such, the product application does not need to populate the TxPDO. If a process data item does not specify input data, the TxPDO and input entry object for that module does not create. Similarly, if a process data item does not specify any output data, the RxPDO and output entry object for that module does not create. After processing the item, the process data represented by that item is available to the EtherCAT leader via traditional TxPDOs and RxPDOs, as well as CoE objects.

If the data type for the subindices in an input or output object is none, the object name is ignored and that object and its corresponding TxPDO or RxPDO does not create. Note that this process allows the creation of input or output only items.

When selecting a process data item, it is possible to set the data (see [Table 23](#)).

Configuration Items

A configuration item creates an object with an index of 0x8xxx. The exact index number depends on the location where an item of this type installs. The index of the configuration data object is calculated as $0x8000 + ((\text{Location} - 1) \times 0x10)$. When adding the item to the system, select the location of the item. Regardless of location, install process data items in the same location as configuration items.

This object holds configuration data sent from the EtherCAT leader to the product. As such, consider a configuration item output only. All input options for a configuration item are disabled. Configuration data is not available to the EtherCAT network in an output entry, and this data is acyclic data.

When selecting a configuration item, it is possible to set the data (see [Table 24](#)).

Table 22. Object Address Formulas

Object Name	Address
TxPDO Index	$0x1600 + (\text{Location} - 1)$
RxPDO Index	$0x1A00 + (\text{location} - 1)$
Input Entry Index	$0x6000 + ((\text{Location} - 1) \times 0x10)$
Output Entry Index	$0x7000 + ((\text{Location} - 1) \times 0x10)$

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Table 23. Process Data Item Parameters

Parameter	Description
Input and Output Object Name	Name of the object (Index 0x6xxx or Index 0x7xxx) that represents the input and output data of the item.
Input and Output Subindex	Selects the subindex within the input and output object currently displayed or edited.
Input and Output Subindex Name	Sets the name of the currently selected subindex. Via CoE, the EtherCAT leader can request this name. If left blank, the system uses the subindex (index number) as the name for this subindex.
Input and Output Subindex Data Type	Selects the data type that the user specifies for the subindex of the item
Total Input and Output Size	Add the size of the input and output subindices as defined by the selected data type to calculate this size.

Table 24. Configuration Item Parameters

Parameter	Description
Output Object Name	Name of the object (Index 0x8xxx) that represents the configuration data of the item.
Output Subindex	Selects the subindex within the configuration object currently displayed or edited.
Output Subindex Name	Sets the name of the currently selected subindex. Via CoE, the EtherCAT leader can request this name. If left blank, the system uses the subindex (index number) as the name for this subindex.
Output Subindex Data Type	Selects the data type that the user specifies for the subindex of the item.
Total Output Size	Add the size of the output subindices as defined by the selected data type to calculate this size.

Diagnostic Items

A diagnostic item creates an object with an index of 0x9xxx. The exact index number depends of the location where an item of this type installs. The index of the diagnostic data object is calculated as $0x9000 + ((\text{Location} - 1) \times 0x10)$. When adding the item to the system, select the location of the item. Regardless of location, install a process data item in the same location as a diagnostic item.

This object holds diagnostic data sent from the EtherCAT leader regarding the state of the input and output on the product. As such, consider a diagnostic item input only. All output options for a diagnostic item are disabled. Diagnostic data is not available to the EtherCAT network in an input entry, and this data is acyclic data.

When selecting a diagnostic item, it is possible to select the data (see [Table 25](#)).

POWERLINK Items

Use POWERLINK items to create objects such as input entries and output entries that represent the product data on the POWERLINK network. The two types of POWERLINK items are analog and digital. These items are both cyclic items that contain the process data the product consumes and produces on the POWERLINK network.

Analog Items

If a user selects an analog item for the item type, the options are 1, 2, or 4 bytes for input or output. When more than 4 bytes of input or output are required, create multiple quantities for that particular item. Multiple quantities fill in the subindices with the specified number of quantities for that item. As such, when creating an item, enter the fields detailed in [Table 26](#).

Digital Items

If a user selects a digital item, only 1 byte is available for input or output. When more than 1 byte of input or output is required, create multiple quantities for that particular item. Multiple quantities fill in the subindices with the specified number of quantities for that item. See [Table 26](#).

Baskets

Baskets hold a device and one or more items. Baskets can group a set of the available items with a given device. Baskets can only contain a single device, although baskets can contain multiple or duplicate items. The devices and items in a basket must exist in the same database file.

Table 25. Diagnostic Item Parameters

Parameter	Description
Input Object Name	Name of the object (Index 0x9xxx) that represents the diagnostic data of the item.
Input Subindex	Selects the subindex within the diagnostic object currently displayed or edited.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Table 25. Diagnostic Item Parameters (Continued)

Parameter	Description
Input Subindex Name	Sets the name of the currently selected subindex. Via CoE, the EtherCAT leader can request this name. If left blank, the system uses the subindex (index number) as the name for this subindex.
Data Type	Selects the data type that the user specifies for the subindex of the item.
Total Input Size	Add the size of the input subindices as defined by the selected data type to calculate this type.

Table 26. Analog and Digital Item Parameters

Parameter	Description
Item Type	Analog or digital.
Item Size	If the item type is analog, the item size can be 1, 2, or 4 bytes, and if the item type is digital, the item size defaults to 1 byte.
Input Quantity	Number of input subindices for the item between 0 and 254.
Output Quantity	Number of output subindices for the item between 0 and 254.

USING THE RPG2 I/O CONFIGURATION TOOL

Starting the RPG2 I/O Configuration Tool

To start the RPG2 I/O configuration tool, navigate to the directory where the RPG2 I/O configuration tool was installed and double-click the **ConfigurationTool.exe** file.

User Interface Overview

When the RPG2 I/O configuration tool first starts up, the window shown in [Figure 3](#) appears.

From this window, the user can begin adding objects to the database or opening previously created databases. For a brief description of what the toolbar buttons do, hover your cursor over that button.

Device, Item, and Basket Operations

The three basic database operations for any object type are add, edit, and delete.

Adding a Device

To add a device to the database, go to the **Device Library** tab and click **Add Device**. The user must then select the device protocol for the new device (see [Figure 116](#)). The user can also add the device via the menu bar, the context menu, or the toolbar.

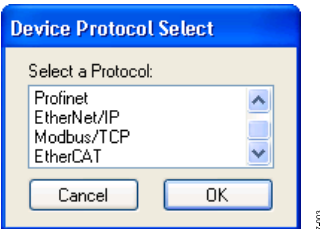


Figure 116. Device Protocol Select Window

Select the protocol for device creation and click **OK**, which then opens a window to add the protocol selected. See the [Example Configurations](#) sections for example device windows. From this window, the user can enter all desired device settings. Once information is entered, click **Add Device**. To stop creating a device and discard any changes, click **Cancel**.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

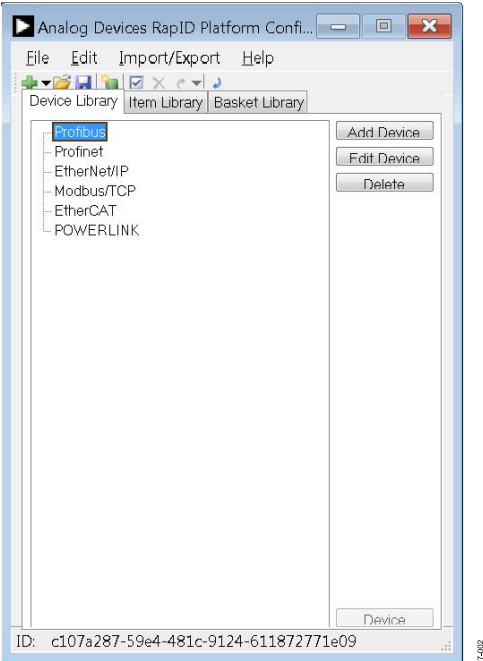


Figure 117. Main RPG2 I/O Configuration Tool Window

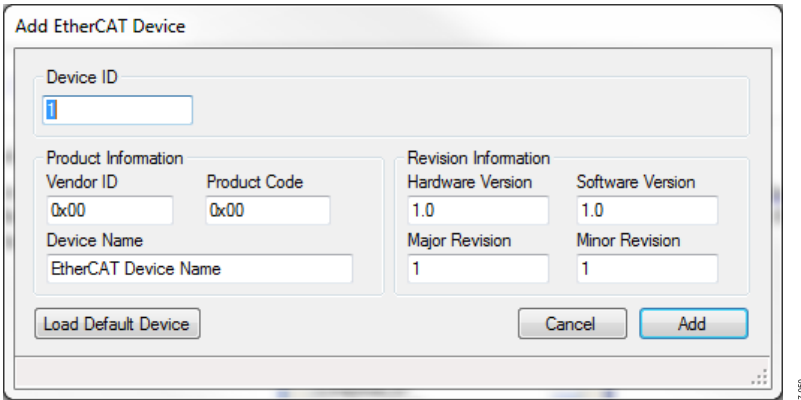


Figure 118. Device Protocol Select Window

To overwrite the information in all fields with the database default device values for that protocol, click **Load Default Device** (see [Figure 118](#)). A default device is useful when creating a large number of similar devices. The user can load the default device for that protocol and then only make minor changes to the device information values.

Setting a Default Device

The user can load a new default device when adding a device or beginning to edit an existing device. To edit the default device, click **Edit > Default > Device** and then select the corresponding protocol of the default device to edit. When done editing the device, click **Commit** to save the default device to the database. It is now possible to load the modified default device by clicking **Load Default Device** in windows to add or edit devices for the corresponding protocol. To stop editing a default device and discard any changes, click **Cancel**.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Editing a Device

To edit a device in the database, select the device to edit in the **Device Library** and then click **Edit Device**. The resulting window contains editable device settings. Right-click the device to edit or select **Edit Device** from the context menu to open the device for editing. To commit the device changes to the database, click **Commit**. To replace this device information with the information of the default device, click **Load Default Device**. To discard any changes, click **Cancel**.

Deleting a Device

To delete a device from the database, select the device to delete and click **Delete**. A prompt to confirm the deletion of this device then appears. Before deleting the device from the database, remove the device from any baskets using the device. The user can also right-click on the device to delete and select **Delete**.

Adding an Item

To add an item to the database, go to the **Item Library** tab and click **Add Item**. A prompt to select the protocol for the new item appears (see [Figure 119](#)). The user can also add the item via the menu bar, the context menu, or the toolbar.

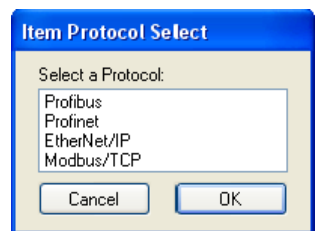


Figure 119. Item Protocol Select Window

Select the protocol for the item and click **OK**. A window then opens to add the protocol selected. See the [Example Configurations](#) sections for example device windows. From this window, the user can enter all desired item settings. Once information is entered, click **Add**. To stop creating a device and discard any changes, click **Cancel**.

To overwrite the information in all fields with the database default device values for that protocol, click **Load Default Device** (see [Figure 4](#)).

A default item is useful when creating a large number of similar items. The user can load the default item for that protocol and then only make minor changes to the device information values. For information on changing the configuration tool default item, see the [Setting a Default Item](#) section.

Setting a Default Item

The user can load a new default item when adding an item or beginning to edit an existing item. To edit the default item, click **Edit > Default > Item** and then select the corresponding protocol of the default item to edit. When done editing the item, click **Commit** to save the default item to the database. It is now possible to load the modified default item by clicking **Load Default Item** to add or edit items for the corresponding protocol. To stop editing a default item and discard any changes, click **Cancel**.

Editing an Item

To edit an item, select the item in the **Item Library** and then click **Edit Item**. The resulting window contains editable item settings. Right-click the item to edit or select **Edit Item** from the context menu to open the item for editing. To commit item changes to the database, click **Commit**. To replace the information of an item with the information of the database default item, click **Load Default Item**. To discard any changes, click **Cancel**.

Deleting an Item

To delete an item from the database, select the item to delete and click **Delete Item**. A prompt to confirm deletion of this item then appears. Before deleting the item from the database, remove the item from any baskets using the item. The user can also right-click on the item to delete and select **Delete Item**.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Adding a Basket

To add a basket to the database, go to the **Basket Library** tab and click **Add Basket**. A prompted to select the protocol for the new basket appears (see [Figure 120](#)). The user can also add the basket via the menu bar, the context menu, or the toolbar.

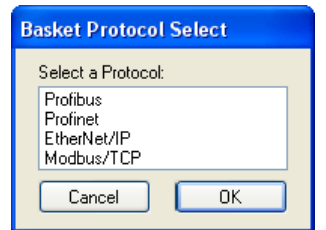


Figure 120. Basket Protocol Select Window

Select the protocol for the basket and click **OK**. The **Add Basket** window then opens (see [Figure 121](#)). There must be at least one device and one item in the database for the protocol selected to begin adding a basket.

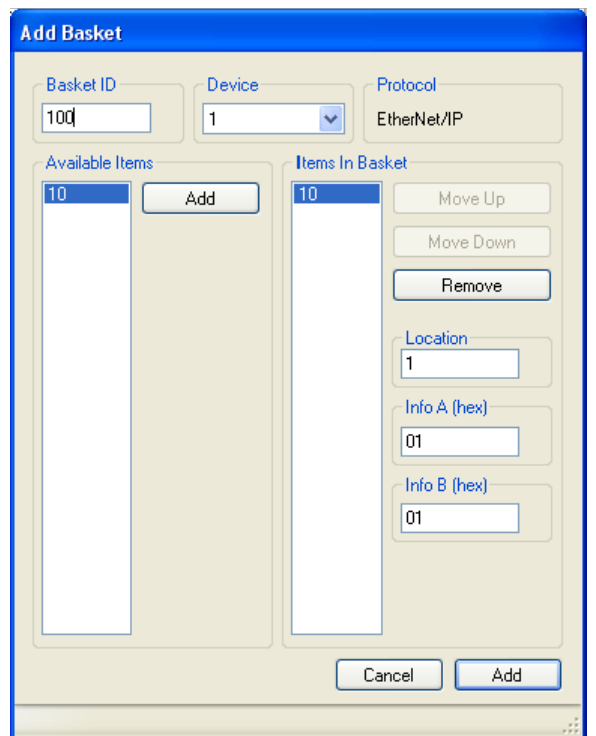


Figure 121. Add Basket Window

In the **Add Basket** window, the user can add the device and any desired items, as well as set the order of the items inside the basket. Once all information is entered, click **Add** to add the item to the database. Before the basket can be added to the database, exactly one device for the basket, and at least one item, must be added to the basket. To stop creating a basket and discard any changes, click **Cancel**.

Each item in a basket has a location parameter associated with it. This parameter refers to the location where the item must install when added to the system at software run time. For Ethernet/IP and Modbus/TCP implementations, set the location parameter to 1 for all items.

Each item also has an **Info A** and **Info B** parameter associated with it. The Unified Interface does not process these parameters. Both of these values are unsigned 32-bit numbers that are user assignable. These parameters can store custom information about each item in a basket that the software can use when the basket fetches at run time.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Editing a Basket

To edit a basket, select the basket to edit in the **Basket Library** and click **Edit Basket**. The resulting window contains editable item settings. Right-click the basket to edit or select **Edit Basket** from the context menu to open the basket for editing. To commit basket changes to the database, click **Commit**. To discard any changes, click **Cancel**.

Deleting a Basket

To delete a basket from the database, select the basket and click **Delete**. A prompt to confirm deletion of this basket appears. Alternatively, the user can right-click on the basket and select **Delete** from the context menu.

Database Operations

Starting a Database

After opening the configuration tool, users can start building a database by adding a device or an item to any protocol list. See the [Adding a Device](#) section and [Adding an Item](#) section for details on adding devices and items. After adding at least one device and one item for a particular protocol, place those objects in a basket.

Saving a Database

After building a database, save it for storage or for future editing. To save the database, click **File > Save Database**. Select a location, enter a filename, and then click **Save**.

Opening a Database

To open a previously saved database, click **File > Open**. Select the database file to open and click **Open**.

Closing a Database

To close the database being working on, click **File > Close Database**. If the database is not saved, a prompt appears to choose to save it. Once the existing database closes, a new, empty database opens.

Database Export and Import Operations

Exporting a Database Report File

During development of a database, it can be useful to have a summary of the objects contained in the database. The RPG2 I/O configuration tool provides the ability to export a database report file. This file contains the same data that is visible in the tool in a .txt file.

To export a database report file, click **Import/Export > Export Database Report**. Navigate to the directory where you want to save the report file, select a file name, and click **Save**.

Exporting a Configuration File

After creating a database with the necessary devices, items, and baskets, export the database to a configuration file so that the configuration information can be loaded onto the product and referenced by the industrial protocol software. The file format chosen by the user depends on the production time needs and the method used to load the configuration information onto the product.

Selecting Objects for Export

To export a configuration file, select the objects to export by setting the configuration tool to export select mode. To put the tool into export select mode, click **Import/Export > Select Objects to Export**. The objects in the database are then available for selection. To select an object, check off the check box next to the object. To export all devices, items, or baskets under a certain protocol, check off the check box next to the protocol.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Exporting Selected Objects

To export the selected objects, click **Import/Export > Export Selected Object To** and select the file format by which to export the objects. See the [Selecting Objects for Export](#) section and [Configuration File Formats](#) section for more information. A prompt then appears for directory selection location and file naming. After entering a file name, click **Save**. Click **Cancel** to stop exporting a configuration file.

After an executable and linkable format (ELF) configuration file exports, there is the option to generate a batch file. When the batch file runs, it loads the ELF configuration file generated into the product via JTAG. The batch file does not need to run from the command line and can run by double-clicking the file. Before the batch file runs, install the CodeSourcery tool chain and JTAG wiggler drivers and connect a JTAG wiggler to the host computer. For the load process to work properly, the ELF file, the generated batch file, and **Rapid.xml** file must all be in the same directory when the batch file runs.

A warning appears when attempting to export a configuration file if selecting a device in a certain protocol for export but no items in that protocol are selected. Likewise, a warning appears if selecting an item in a protocol for export but no devices in that protocol are selected.

Configuration File Formats

The RPG2 I/O configuration tool can export configuration files in four different formats. However, the only one that is used for the RPG2 product line is the .bin option. Do not use the other formats for this solution because the other formats are specific to testing only.

Importing a Configuration File

During development, it may be convenient to populate the database with information contained in configuration files. To import a configuration file, click **Import/Export > Import File**, select the configuration file to import, and click **Open**.

If the database currently open in the tool is not empty, the **Discard Duplicates?** window opens (see [Figure 122](#)). Click **Yes** to keep any duplicate objects in the database and to discard the duplicate objects in the configuration file during the import. Click **No** to discard any duplicate objects in the database and replace these objects with the duplicate objects in the configuration file. Click **Cancel** to cancel the import.

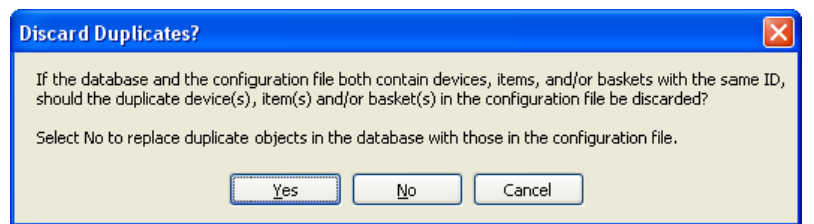


Figure 122. Discard Duplicates? Window

EXAMPLE CONFIGURATIONS

The following section shows how to create configurations for five products that all have the same input and output features but communicate via PROFINET, Ethernet/IP, EtherCAT, POWERLINK, and Modbus/TCP. In the following examples, the input and output features are one set of 16-bit digital inputs and outputs, two sets of 16-bit analog inputs and outputs, and one 16-bit control value that is received from the controller but not sent to it.

PROFINET Example Configuration

Example PROFINET Device

The device object for the example PROFINET device appears in [Figure 9](#).

Example PROFINET Items

Four PROFINET items represent the input and output data for the example product: one for the digital input and output data, one for the analog input and output data, one for the control data, and one for the module data that is associated with each of previous three items that are used for input and output.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

The digital input and output data appears in [Figure 10](#), the analog input and output data appears in [Figure 11](#), the control data appears in [Figure 12](#), and the module data appears in [Figure 13](#).

The controller provides this acyclic data as defined by the included example GSDML file as module data. The value of the module data is arbitrary.

Example PROFINET Basket

If defining at least one device and one item for the product, place the device and item in a basket to make adding them to the system at run time easier. Using a basket to add a device and set of items is not a requirement. The device and set of items can also be added individually. An example PROFINET basket appears in [Figure 128](#).

Acyclic Data Item ID 503 was added to the example basket three times. Each instance of this item has a location that matches one of the other data items. As such, each input and output data item has corresponding acyclic module data.

Add Profinet Device

Configuration Device ID
400

Profinet Device Information

Vendor ID	Device ID	Profile ID	Profile Type
0x1B9	61441	0	3
Device Type	Order ID		
Innovasic I/O Device	RapID-NI V2002		
Module ID	Submodule ID		
0x10100001	0x10110001		

Revision Information

Major Revision	Minor Revision	Internal Revision	Revision Counter
3	0	0	1

Profinet SNMP Description

SNMP System Description	Enterprise ID
Innovasic RapID Platform, PROFINET Device	36927
Internal Interface Description	
Innovasic RapID Platform, internal	
Port 1 Interface Description	Port 2 Interface Description
Innovasic RapID Platform, port-001	Innovasic RapID Platform, port-002

Load Default Device Cancel Add

Figure 123. Example PROFINET Device

Add Profinet Item

Item ID Item Type

500 Cyclic

Cyclic Item Information

Module ID	Submodule ID	Input Size (bytes)	Output Size (bytes)
0x10400000	0x10440001	2	2

Load Default Item Cancel Add

Figure 124. Example PROFINET Item (Digital Data)

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Add Profinet Item

Item ID: 501 Item Type: **Cyclic**

Cyclic Item Information

Module ID	Submodule ID	Input Size (bytes)	Output Size (bytes)
0x10500000	0x10550001	4	4

Load Default Item Cancel Add

Figure 125. Example PROFINET Item (Analog Data)

Add Profinet Item

Item ID: 502 Item Type: **Cyclic**

Cyclic Item Information

Module ID	Submodule ID	Input Size (bytes)	Output Size (bytes)
0x10600000	0x10660001	0	2

Load Default Item Cancel Add

Figure 126. Example PROFINET Item (Control Data)

Add Profinet Item

Item ID: 503 Item Type: **Acyclic**

Acyclic Item Information

Index	Input Size (bytes)	Output Size (bytes)
1	0	3

Load Default Item Cancel Add

Figure 127. Example PROFINET Item (Module Data)

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Add Basket

Basket ID: 1000 Device: 400 Protocol: Profinet

Available Items: 500, 501, 502, 503 Add

Items In Basket: 500, 501, 502, 503 Move Up, Move Down, Remove

Location: 3 Info A (hex): 00 Info B (hex): 00

Cancel Add

Figure 128. Example PROFINET Basket

Current IRT Example PROFINET Configuration

With the implementation of PROFINET IRT, a new GSDML file has been introduced. The current IRT example configuration has a single digital input and output data item representing eight separate submodules in the included example GSDML file. Each of these submodules can be added as part of the **ni-example-app** source code. See the sample configuration and the GSDML file for details, as shown in [Figure 129](#) to [Figure 131](#).

Add Profinet Device

Configuration Device ID: 403

Profinet Device Information

Vendor ID	Device ID	Profile ID	Profile Type
0x1B9	61441	0	3
Device Type	Order ID		
I/O Device	EV-RPG2-PNZ-U1		
Module ID	Submodule ID		
0x20001	0x20002		

Revision Information

Major Revision	Minor Revision	Internal Revision	Revision Counter
6	0	2	0

Profinet SNMP Description

SNMP System Description	Enterprise ID
Analog Devices, RapID Platform, PROFINET Network Interface	24696
Internal Interface Description	
Analog Devices, RapID Platform, Internal	
Port 1 Interface Description	Port 2 Interface Description
Analog Devices, RapID Platform, Port1	Analog Devices, RapID Platform, Port2

Load Default Device Cancel Add

Figure 129. Add Profinet Device Window

RPG2 I/O CONFIGURATION TOOL USER GUIDE

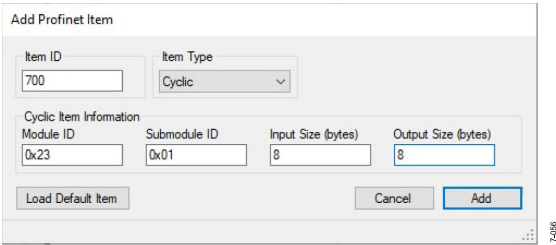


Figure 130. Add Profinet Item Window

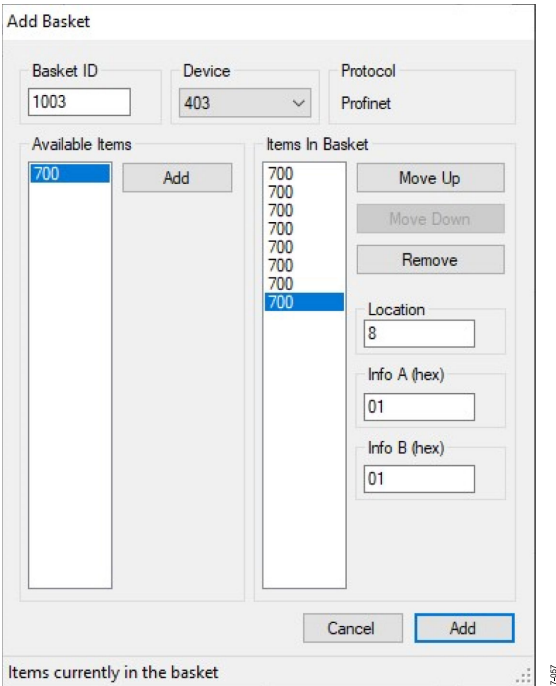


Figure 131. Add Basket Window

Ethernet/IP Example Configuration

Example Ethernet/IP Device

The device object for the example Ethernet/IP product appears in [Figure 15](#).

Example Ethernet/IP Items

Three Ethernet/IP items represent the input and output data for the example product: one for the digital input and output data, one for the analog input and output data, and one for the control data.

The digital input and output data appears in [Figure 16](#), and the analog input and output data appears in [Figure 17](#), and the control data appears in [Figure 18](#).

Example Ethernet/IP Basket

If defining at least one device and one item for the product, place the device and item in a basket to make adding them to the system at run time easier. Using a basket to add a device and set of items to the system is not a requirement. The device and items can also be added individually.

The example Ethernet/IP basket appears in [Figure 136](#). The Ethernet/IP network application does not use the location parameter. The location parameter must be set to 1 for all items in an Ethernet/IP basket.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Add EtherNet/IP Device

Device ID

400

Ethernet Configuration

Port 1 Speed

100 Mbps

Port 1 Duplex

Full

Port 1 Auto Negotiation

Auto

Port 1 Admin State

Enable

Port 2 Speed

100 Mbps

Port 2 Duplex

Full

Port 2 Auto Negotiation

Auto

Port 2 Admin State

Enable

Product Information

Vendor ID

1060

Device Type

12

Product Code

1

Major Revision

1

Minor Revision

1

Product Name

Test Product

Network Configuration

Static IP Address

192.168.21.99

Subnet Mask

255.255.255.0

Gateway Address

0.0.0.0

DNS 1 Address

0.0.0.0

DNS 2 Address

0.0.0.0

Enable DHCP

☒

Enable ACD

☒

Enable DLR

☐

Domain Name

domain

Hostname

hostname

Multicast TTL

1

Multicast Allocation Control

Disable

Max Multicast Addresses

16

Multicast Start Address

239.192.1.0

Load Default Device

Cancel

Add

Figure 132. Example Ethernet/IP Device

Add EtherNet/IP Item

Item ID

500

Output Data Location and Size

Consume Instance ID

100

Consumption Size (bytes)

2

Item Type

Unique

Input Data Location and Size

Produce Instance ID

101

Production Size (bytes)

2

Load Default Item

Cancel

Add

Figure 133. Example Ethernet/IP Item (Digital Data)

Add EtherNet/IP Item

Item ID

501

Output Data Location and Size

Consume Instance ID

102

Consumption Size (bytes)

4

Item Type

Unique

Input Data Location and Size

Produce Instance ID

103

Production Size (bytes)

4

Load Default Item

Cancel

Add

Figure 134. Example Ethernet/IP Item (Analog Data)

RPG2 I/O CONFIGURATION TOOL USER GUIDE

The 'Add Ethernet/IP Item' dialog box is shown. It has a title bar 'Add Ethernet/IP Item'. Inside, there are two main sections. The top section has 'Item ID' (text box with '502') and 'Item Type' (dropdown menu with 'Unique' selected). The bottom section has 'Output Data Location and Size' with 'Consume Instance ID' (text box with '104') and 'Consumption Size (bytes)' (text box with '2'). Below that is 'Input Data Location and Size' with 'Produce Instance ID' (text box with '0') and 'Production Size (bytes)' (text box with '0'). At the bottom are three buttons: 'Load Default Item', 'Cancel', and 'Add'.

Figure 135. Example Ethernet/IP Item (Control Data)

The 'Add Basket' dialog box is shown. It has a title bar 'Add Basket'. Inside, there are three fields at the top: 'Basket ID' (text box with '1000'), 'Device' (dropdown menu with '400' selected), and 'Protocol' (dropdown menu with 'Ethernet/IP' selected). Below these are two lists: 'Available Items' (500, 501, 502) and 'Items In Basket' (500, 501, 502). Between the lists is an 'Add' button. To the right of the 'Items In Basket' list are three buttons: 'Move Up', 'Move Down', and 'Remove'. Below the lists are three text boxes: 'Location' (1), 'Info A (hex)' (00), and 'Info B (hex)' (00). At the bottom are two buttons: 'Cancel' and 'Add'.

Figure 136. Example Ethernet/IP Basket

Modbus/TCP Example Configuration

Example Modbus/TCP Device

The device object for the example Modbus/TCP product appears in [Figure 20](#).

Example Modbus/TCP Items

Three Modbus/TCP items represent the input and output data for the example product: one for the digital input and output data, one for the analog input and output data, and one for the control data. In [Figure 21](#) and [Figure 22](#), the **Register Offset** box and **Bit Offset** box are disabled because register space mapping is disabled, and the bit space mapping is in auto mode for this item. This item configuration prevents the digital data from mapping into the register space and allows the digital data to map automatically into the bit space.

The digital input and output data appears in [Figure 21](#), the analog input and output data appears in [Figure 22](#), and the control data appears in [Figure 23](#). The **Register Offset** box is enabled and the **Bit Offset** box is disabled in [Figure 23](#) because the bit space mapping is disabled, and

RPG2 I/O CONFIGURATION TOOL USER GUIDE

the use selected option is selected for register space mapping. Configuring the item this way prevents control data from being mapped into the bit space and allows control data to be mapped into a specific area of the register space away from the input and output data.

Example Modbus/TCP Basket

If defining at least one device and one item for the product, place the device and item in a basket to make adding them to the system at run time easier. Using a basket to add a device and set of items to the system is not a requirement. The device and items can also be added individually.

An example Modbus/TCP basket appears in [Figure 141](#). The Modbus/TCP network application does not use the location parameter. The location parameter must be set to 1 for all items in a Modbus/TCP basket.

74285

Figure 137. Example Modbus/TCP Device

74286

Figure 138. Example Modbus/TCP Item (Digital Data)

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Add Modbus/TCP Item

Item ID 501	Swap Mode None	Register Mirroring <input type="checkbox"/> Mirror Registers
Register Space Mapping Use Auto-mode for Register Offset	Bit Space Mapping Disable Bit Space Mapping	
Item Information		
Input Size 4	Output Size 4	Register Offset 0
		Bit Offset 0
Load Default Item		Cancel Add

Figure 139. Example Modbus/TCP Item (Analog Data)

Add Modbus/TCP Item

Item ID 502	Swap Mode None	Register Mirroring <input type="checkbox"/> Mirror Registers
Register Space Mapping Use Selected Register Offset	Bit Space Mapping Disable Bit Space Mapping	
Item Information		
Input Size 0	Output Size 2	Register Offset 0x3E8
		Bit Offset 0
Load Default Item		Cancel Add

Configuration item ID

Figure 140. Example Modbus/TCP Item (Control Data)

RPG2 I/O CONFIGURATION TOOL USER GUIDE

The screenshot shows the 'Add Basket' dialog box. At the top, there are three input fields: 'Basket ID' with the value '1000', 'Device' with a dropdown menu showing '400', and 'Protocol' with the text 'Modbus/TCP'. Below these fields are two vertical lists. The left list, titled 'Available Items', contains the numbers 500, 501, and 502, with 502 highlighted. The right list, titled 'Items In Basket', also contains 500, 501, and 502, with 502 highlighted. Between these two lists is an 'Add' button. To the right of the 'Items In Basket' list are three buttons: 'Move Up', 'Move Down', and 'Remove'. Below the lists are three more input fields: 'Location' with the value '1', 'Info A (hex)' with the value '00', and 'Info B (hex)' with the value '00'. At the bottom of the dialog are two buttons: 'Cancel' and 'Add'.

Figure 141. Example Modbus/TCP Basket

EtherCAT Example Configuration

Example EtherCAT Device

The device object for the example EtherCAT product appears in [Figure 25](#).

Example EtherCAT Items

Three items represent the input and output data for the example product: one for the digital input and output data, one for the analog input and output data, and one for the control data.

The digital input and output data appears in [Figure 26](#), the analog input and output data appears in [Figure 27](#), and the control data appears in [Figure 28](#).

Example EtherCAT Basket

If defining at least one device and one item for the product, place the device and item in a basket to make adding them to the system at run time easier. Using a basket to add a device and set of items to the system is not a requirement. The device and items can also be added individually. An example EtherCAT basket appears in [Figure 146](#).

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Add EtherCAT Device

Device ID

400

Product Information

Vendor ID

0xC0DECAFE

Product Code

0xF1D05200

Device Name

RaplD_Platform_NI

Revision Information

Hardware Version

1.0

Software Version

5.0

Major Revision

1

Minor Revision

1

Load Default Device

Cancel

Add

Figure 142. Example EtherCAT Device

Edit EtherCAT Item

Item ID

500

Item Type

Process Data

Input Object Info

Input Object Name

Digital Ins

Input Subindex Info

Input Subindex

1

Input Subindex Name

Digital Inputs

Input Subindex Data Type

BITARR16

Output Object Info

Output Object Name

Digital Outs

Output Subindex Info

Output Subindex

1

Output Subindex Name

Digital Outputs

Output Subindex Data Type

BITARR16

Item Size Summary

Total Input Size (bytes)

2

Total Output Size (bytes)

2

Load Default Item

Cancel

Commit

Figure 143. Example EtherCAT Item (Digital Data)

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Edit EtherCAT Item

Item ID

501

Item Type

Process Data

Input Object Info

Input Object Name

Analog Ins

Input Subindex Info

Input Subindex

1

Input Subindex Name

Analog In Ch1

Input Subindex Data Type

UINT16

Output Object Info

Output Object Name

Analog Outs

Output Subindex Info

Output Subindex

1

Output Subindex Name

Analog Out Ch1

Output Subindex Data Type

UINT16

Item Size Summary

Total Input Size (bytes)

4

Total Output Size (bytes)

4

Load Default Item

Cancel

Commit

Figure 144. Example EtherCAT Item (Analog Data)

Edit EtherCAT Item

Item ID

502

Item Type

Process Data

Input Object Info

Input Object Name

Input Subindex Info

Input Subindex

1

Input Subindex Name

Input Subindex Data Type

UNUSED

Size (bytes)

0

Output Object Info

Output Object Name

Control

Output Subindex Info

Output Subindex

1

Output Subindex Name

Control Data

Output Subindex Data Type

UINT16

Item Size Summary

Total Input Size (bytes)

0

Total Output Size (bytes)

2

Load Default Item

Cancel

Commit

Figure 145. Example EtherCAT Item (Control Data)

analog.com

Rev. A | 106 of 180

RPG2 I/O CONFIGURATION TOOL USER GUIDE

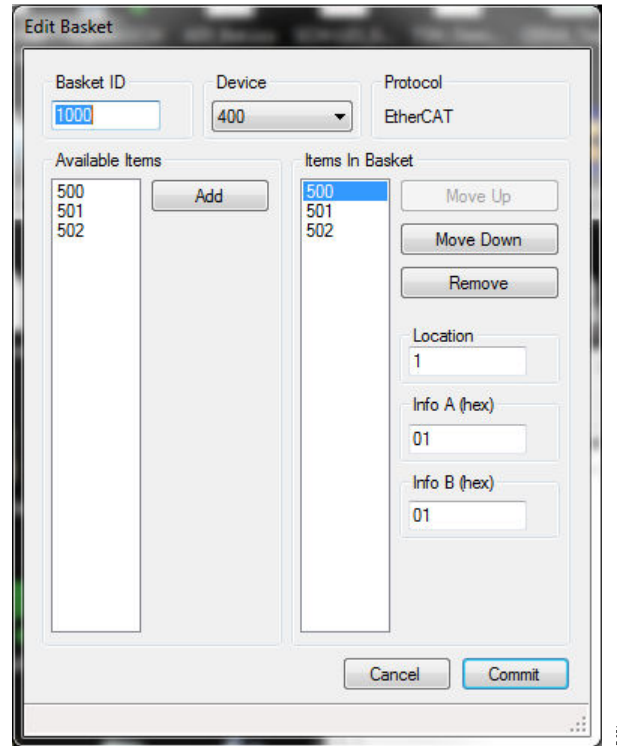


Figure 146. Example EtherCAT Basket

POWERLINK Example Configuration

Example POWERLINK Device

The device object for the example POWERLINK object appears in [Figure 30](#).

Example POWERLINK Items

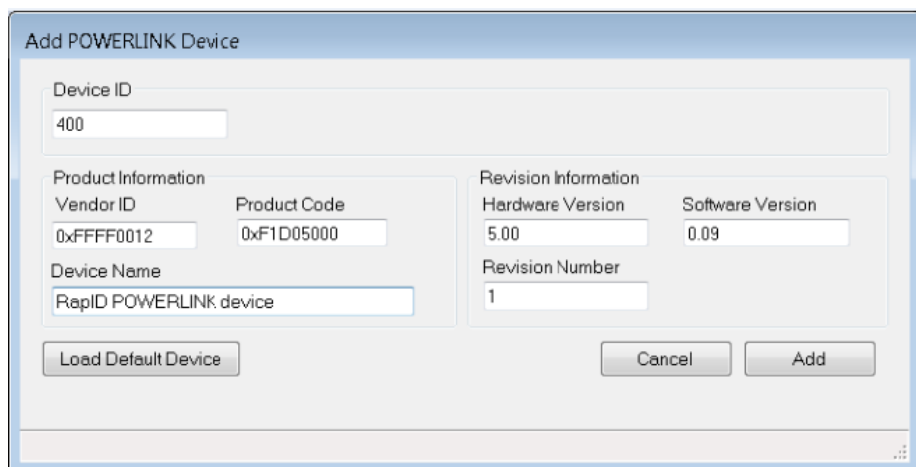
Three items represent the input and output data for the example product: one for the digital input and output data, one for the analog input and output data, and one for the control data.

The digital input and output data appears in [Figure 31](#), the analog input and output data appears in [Figure 32](#), the control data appears in [Figure 33](#).

Example POWERLINK Basket

If defining at least one device and one item for the product, place the device and item in a basket to make adding them to the system at run time easier. Using a basket to add a device and set of items to the system is not a requirement. The device and items can also be added individually. An example POWERLINK basket appears in [Figure 151](#).

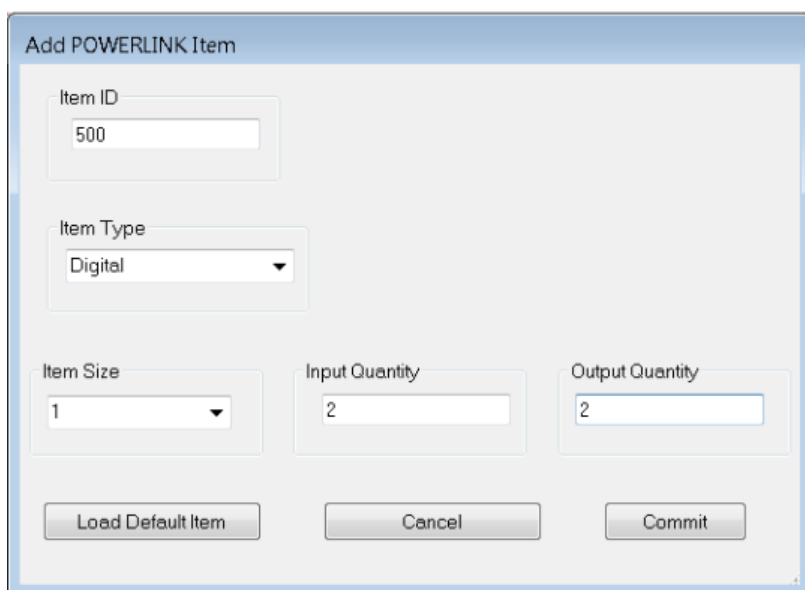
RPG2 I/O CONFIGURATION TOOL USER GUIDE



The 'Add POWERLINK Device' dialog box contains the following fields and controls:

- Device ID:** Text input field with value '400'.
- Product Information:**
 - Vendor ID:** Text input field with value '0xFFFF0012'.
 - Product Code:** Text input field with value '0xF1D05000'.
 - Device Name:** Text input field with value 'RapID POWERLINK device'.
- Revision Information:**
 - Hardware Version:** Text input field with value '5.00'.
 - Software Version:** Text input field with value '0.09'.
 - Revision Number:** Text input field with value '1'.
- Buttons:** 'Load Default Device', 'Cancel', and 'Add'.

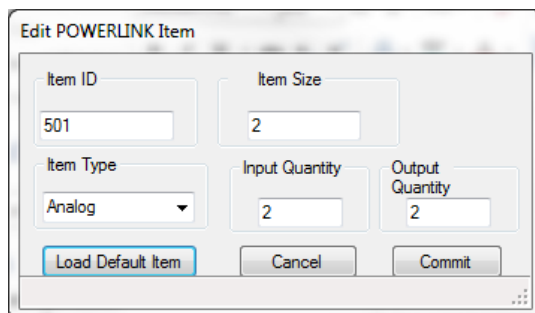
Figure 147. Example POWERLINK Device



The 'Add POWERLINK Item' dialog box contains the following fields and controls:

- Item ID:** Text input field with value '500'.
- Item Type:** Dropdown menu with 'Digital' selected.
- Item Size:** Dropdown menu with '1' selected.
- Input Quantity:** Text input field with value '2'.
- Output Quantity:** Text input field with value '2'.
- Buttons:** 'Load Default Item', 'Cancel', and 'Commit'.

Figure 148. Example POWERLINK Item (Digital Data)



The 'Edit POWERLINK Item' dialog box contains the following fields and controls:

- Item ID:** Text input field with value '501'.
- Item Size:** Text input field with value '2'.
- Item Type:** Dropdown menu with 'Analog' selected.
- Input Quantity:** Text input field with value '2'.
- Output Quantity:** Text input field with value '2'.
- Buttons:** 'Load Default Item', 'Cancel', and 'Commit'.

Figure 149. Example POWERLINK Item (Analog Data)

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Figure 150 shows the 'Edit POWERLINK Item' dialog box. It has the following fields and values:

Field	Value
Item ID	502
Item Size	1
Item Type	Digital
Input Quantity	0
Output Quantity	2

Buttons: Load Default Item, Cancel, Commit.

Figure 150. Example POWERLINK Item (Control Data)

Figure 151 shows the 'Edit Basket' dialog box. It has the following fields and values:

Field	Value
Basket ID	1000
Device	400
Protocol	POWERLINK

Available Items: 500, 501, 502. Items In Basket: 500, 501, 502.

Buttons: Add, Move Up, Move Down, Remove, Cancel, Commit.

Figure 151. Example POWERLINK Basket

PROFINET GSDML FILES

A sample set of system configuration and GSDML files is part of the software zip file download from the ADIN2299 main product page at www.analog.com/adin2299. The configuration follows the examples described in this user guide, and the GSDML file reflects that as well. The configuration data contains three items that describe three modules with an associated parameter data item. The second configuration data file has eight modules.

The GSDML file details how these three modules connect a PROFINET controller to the example device. Then, to describe the user device, modify the configuration and GSDML files. Review the example configuration in this section thoroughly before modifying the configuration and associated GSDML files.

This configuration tool creates the configuration file, and the configuration file uploads to the network interface module, which contains data that describes the input and output items and the physical device.

When this data is entered into this configuration tool, each configuration entity, for example, a device or item, is given a unique ID number. The module runtime uses these ID numbers to locate the device settings and input or output data of the item. The ID numbers are arbitrary, or the user can assign the ID numbers in any way desired. However, the values must be unique. When the device and all the necessary items are created, the data is placed into a basket, which is referenced singularly by the host software. The example configuration provides a single device (ID 400 or ID 403) and three input and/or output items (Item 500, Item 501, and Item 502, or alternatively multiples of Item 700). In this example, all of this information is placed into a basket with the ID of 1000.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

See [Table 27](#) for details on Item 500 through Item 502.

GSDML File Generation Guidelines

The PROFINET GSDML file must follow the current specification from PROFIBUS International. The specification and a GSDML viewer is downloaded via the PROFIBUS internal website. Note that users must be a member to obtain this information from PROFIBUS International. The GSDML viewer checks the GSDML for specification errors. However, this viewer does not detect all .xml syntax errors, and an .xml editor with a syntax checker is also required.

The user system can modify the following sections of the GSDML file.

DeviceIdentity Section

This section of the file defines the vendor ID and device ID, which must match the vendor ID and device ID in the device description of the configuration file. Each PROFINET vendor must have a vendor ID issued by PROFIBUS International. Send a request to info@profibus.com for the vendor ID. Note that there is no charge for the ID.

DeviceAccessPointList Section

This section describes which modules can plug into which slots and the physical device. The ModuleIdentNumber and SubmoduleIdentNumber must match the **Module ID** and **Sub Module ID** in the device description of the configuration file. If the product family has multiple DeviceAccessPointItems, the configuration file must contain device descriptions for each DeviceAccessPointItem.

ModuleList Section

The data in this section describes the input and output modules of the product, which can be physical modules that plug into a back plane or a description of the input and output of a single device. Each ModuleItem must have a matching cyclic item description in the configuration file.

ParameterDataRecordItem Section

Each module can have a ParameterDataRecordItem, which is predefined data set when making a connection. If a module has a ParameterDataRecordItem item, define an acyclic item in the configuration file and add this item to the same slot as the associated cyclic item. The acyclic item must have the same index and data size as the ParameterDataRecordItem item of the module.

GraphicsList Section

Use this section of the GSDML file to include a graphic for the device in the controller software. Each DeviceAccessItem item can be associated with a unique graphic.

CategoryList Section

This section of the file organizes the device modules into a directory structure in the controller software.

Table 27. Item 500 to Item 502 and Item 700 Descriptions (Item 700 Appears in the Second I/O Configuration File)

Item ID	PROFINET Module ID	PROFINET Submodule ID	Data Size (Bytes)	
			Consumer Output and Input Data	Producer Output and Input Data
Item 500	0x10400000	0x10440001	2	2
Item 501	0x10500000	0x10550001	4	4
Item 502	0x10600000	0x10660001	2	Not applicable
Item 700	0x00000023	0x00000001	8	8

ETHERNET/IP EDS FILES

A sample set of system configuration and EDS files is part of the software zip file download from the ADIN2299 main product page at www.analog.com/adin2299. The configuration and the EDS files follow the examples described in this user guide in the following sections. The configuration data contains three items that describe five instances of EtherNet/IP assembly objects. Descriptions of these five objects are within the EDS file such that an industrial PLC can connect to the example device with minimal effort. Closely examine the example configuration before modifying the configuration and EDS files to conform to the needs of your specific application.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

The example configuration data is contained in two files: **EIP_demo_cfg** (no file extension) and **EIP_demo_cfg.bin**. The **EIP_demo_cfg** file is raw data that this configuration tool creates and consumes. The **EIP_demo_cfg.bin** file contains the same data as the **EIP_demo_cfg** file in a format that the RPG2 module can consume. This configuration tool automatically creates the **.bin** file when the configuration file is saved.

The example EDS file is in the **EIP_demo_cfg.EDS** file.

This configuration tool creates the configuration file, and the details of that process are described in this user guide. After the file creates, it then loads to the network interface module. The file contains data that describes the input and output items, as well as the device itself. Each configuration entity (for example, a device or item) is given a unique ID number when data is written to the tool. The module run time uses these ID numbers to locate the device settings and input and output data of the item. The ID numbers are arbitrary and can be assigned by the user in any way desired, but each ID number must be unique.

When the device and all the necessary items are created, this configuration tool places the data into a basket (also contained in the file). The host software must only reference this basket ID. The example configuration provides a single device (ID 400) and three input items (Item 500, Item 501, and Item 502) that are placed into a basket with an ID of 1000.

Table 28 describes the input and output details of the example configuration.

A description of the EtherNet/IP EDS file corresponding to this example configuration follows. In addition, information is included that can assist with modifying this example EDS file to suit the needs of an individual system.

EDS File Generation Guidelines

The EDS file describes the valid connection points and assembly objects within the device. The params sections are referenced by the connection points and can be used to specify details such as offset, scaling, and engineering units. Refer to the provided demonstration file, **EIP_demo_cfg.eds**. This file is heavily commented and provides an example implementation that matches the example configuration data.

The network interface module implements all of the Ethernet/IP objects necessary to support explicit messaging and Class 1 input and output connections and does not support the use of parameter objects or parameter object stubs. Rather, the EDS file has a number of params sections.

Comments in the EDS file start with a dollar sign (\$) and continue to the end of the line. For full details of the EtherNet/IP EDS syntax file see the ODVA Specification Volume 1, Chapter 7.

Table 28. Input and Output Parameters for Example EtherNet/IP Configuration

Valid EtherNet/IP Connection Types	Configuration Item ID	Consume Assembly Object		Produce Assembly Object	
		Instance ID	Size (Byte)	Instance ID	Size (Bytes)
Exclusive Owner (EO), Input Only (IO), Listen Only (LO), and Output Only (OO)	500	100	2	101	2
EO, IO, LO, OO	501	102	4	103	4
EO, OO	502	104	2	None	Not applicable

Table 29. Sample Connection Parameters

Parameter	Description
O → T	Originator to target, which refers to the direction the Class 1 data flows.
T → O	Target to originator, which refers to the direction the Class 1 data flows.
EO	Exclusive owner, which refers to a type of Class 1 connection point, and the input and output data flows in both directions.
LO	Listen only, which refers to a type of Class 1 connection point. LO is dependent on another connection, and the input and output data only flows in the T → O direction.
IO	Input only, which refers to a type of Class 1 connection point, and the input and output data only flows in the T → O direction.
OO	Output only, which refers to a type of Class 1 connection point, and the input and output data only flows in the O → T direction.
Originator	The EtherNet/IP scanner, for example, the PLC.
Target	The EtherNet/IP adapter, for example, the network interface module.
Class 1	The cyclic CIP input and output connection.
CIP	The common industrial protocol.
ODVA	Open DeviceNet Vendor Association.
Consume	Refers to the adapter consuming Class 1 input and output data. Used to drive device outputs.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Table 29. Sample Connection Parameters (Continued)

Parameter	Description
Produce	Refers to the adapter producing Class 1 input and output data. Derived from device inputs.

The EDS file is broken into several sections, each of which delimits by a keyword contained in square brackets. The following are sections of importance for this application:

- ▶ File
- ▶ Device
- ▶ Params
- ▶ Assembly
- ▶ Connection Manager

Ignore the remaining sections and use as is.

File Section

This section describes the device in general terms, names, and versions the EDS file. This data refers to the EDS file itself, not the device. The revision field has no relationship to the revision field in the device section. See the ODVA Specification, Volume 1, Section 7-3.6.2.

Device Section

The data in the device section must match the data provided by the device in the identity object. This configuration tool specifies the identity object data and this data is stored in the device object of the configuration data. Part of this data is an ODVA Vendor ID, which is only obtained by joining ODVA. See the ODVA Specification, Volume 1, Section 7-3.6.3.

Params Section

The demo file contains five items in the params section: Param1 through Param5. None of these items are addressable within the device itself (the link path field is a null string). The EDS file contains these items so the user can specify engineering units, scaling, offsets, and so forth. The sample EDS file has one param item for each of the five assembly objects defined in the demonstration configuration data. The params section is one of two locations where the data size and data type of each assembly object is specified (the type codes are found in the ODVA Specification, Appendix C-6.1). The assem items in the assembly section refer to these params as Param1 through Param5. The data size here must match the size specified in the assembly section. The offset and scaling fields are left blank in this example but can be changed to suit the user application. See the ODVA Specification, Volume 1, Section 7-3.3.6.

Assembly Section

Leave most of the assembly section unchanged. To suit a configuration, the user can require an adjustment in the MaxInst and Max_Number_Of_Static_Instances variables. All assembly object instances in the network interface module are static. The example EDS file sets this field to 5 by default.

To suit an application, the Assem1 through Assem5 items within this section can require some adjustments. More or less of these items can be required depending on how many assembly objects the device has. Use these items to name and size the assembly objects. The other fields used in these items are member size and member reference. The param size must match the size of the items in the configuration data. See the ODVA Specification, Volume, 1 Section 7-3.6.8.1.

Connection Manager Section

Leave most of the items in the Connection Manager section unchanged. There is only one Connection Manager object in the network interface module.

Use the Connection1 through Connection5 items in the connection manager section to make any adjustments to suit your application. More or less of these items can be required, depending on how many Class 1 input and output connection points the device has. These items describe the particulars of the Class 1 input and output connection points in the device. The PLC uses the connection points to exchange cyclic data with the device.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

There are four types of connection points defined within the ODVA specification: EO, IO, OO, and LO.

Support for redundant owner connections is not available. See the ODVA Specification, Volume 1, Section 3-6.4. The network interface module supports the EO, IO, and LO connection points.

It is possible to define an OO connection point. The ODVA specification does not call out the definition for this connection point explicitly but OO can exist. The OO connection sends data to a device output only, and in this example, is used to create a device control register. The device control register can also be written to with an explicit message. The Connection5 parameter in the EDS file shows the OO connection as a Class 1 connection point. Writing to the control register with an explicit message allows connection to the control register with a Class 1 cyclic connection.

The following are EDS connection point item fields that are significant to this application:

- ▶ Bits[24:27] of the trigger and transport mask (the application type field, see the ODVA Specification, Section 7-3.6.10.1.1)
- ▶ Bits[8:10] and Bits[12:14] of the connection parameters word (the O → T and T → O fields, see the ODVA Specification, Section 7-3.6.10.1.2)
- ▶ The connection name and help string (see the ODVA Specification, Section 7-3.6.10.1.10 and Section 7-3.6.10.1.11)
- ▶ The connection path (ODVA Specification, Section 7-3.6.10.1.12 and Section 3-5.5.1.12)

The sample EDS file is heavily commented and shows each connection point type in use. The connection type (EO, IO, LO, and OO) determines the application type field of the trigger and transport mask and determines the appropriate values for the O → T and T → O header fields of the connection parameters word.

Table 30 describes the connection types and values of these fields.

The last item in the connection point definition section is the link path. The link path is heavily commented in the example EDS file. The following describes the link path in more detail:

- ▶ This item consists of four pairs of bytes (hexadecimal) that describe the CIP connection path within the network interface module.
- ▶ The ODVA Specification, Section 3-5.5.1.12 describes the exact decoding of this data.
- ▶ The last three pairs of bytes are the config, consume, and produce objects in the device that the Class 1 connection point references.
- ▶ The network interface module does not use the config path (Byte 3 and Byte 4). Therefore, do not change the provided values.
- ▶ The last two pairs of bytes (consume and produce), Byte 6 and Byte 8, specify the assembly instance IDs that the connection point uses.
- ▶ All values in these fields specified in hexadecimal.
- ▶ For specific examples, see the detailed comments in the example EDS file.

Note that Byte 6 and Byte 8 of the link path are affected by the connection type (EO, IO, LO, and OO). Table 31 describes the conditions required for each byte to adhere to a connection type.

Table 30. EtherNet/IP Connections Overview

Connection Type	Application Type Field	O → T Field	T → O Field
EO	4	4	0
IO	2	3	0
LO	1	4	3
OO	4	4	3

Table 31. EtherNet/IP Connection Types for Example Configuration

Connection Type	Byte 6	Byte 8
EO	O → T (consume) assembly instance ID	T → O (produce) assembly instance ID
IO	FE (specifies instance 254)	T → O (produce) assembly instance ID
LO	FF (specifies instance 255)	T → O (produce) assembly instance ID
OO	O → T (consume) assembly instance ID	FE

EDS File Details Pertaining to Configuration Assemblies

Refer to the example EDS file, **Test_Product_With_CFG_Data.eds**. This file works in conjunction with the common interface configuration data file, **EIP_demo_cfg**. These files are included with the EtherNet/IP system software download. Within the EDS file, any text beginning with

RPG2 I/O CONFIGURATION TOOL USER GUIDE

a dollar sign (\$) represents a comment. The example EDS file is heavily commented. Refer to these comments when using this user guide. Note that the configuration data file, **EIP_w_CFG**, contains Item 500 and Item 501 for EtherNet/IP.

In the params section in the EDS file, the params provide a way to create elements that can be referenced by other sections of the EDS file. In this example, the params define the allowable requested packet interface (RPI) range of the exclusive owner connection and define the configuration and the input and output data size and format. The defined params are as follows:

- ▶ Param1: RPI range, used in the connection manager section by Connection1.
- ▶ Param2 and Param3: output (consume) assembly data size and format, used in the assembly section by Assem101.
- ▶ Param4 through Param6: input (produce) assembly data size and format, used in the assembly section by Assem102.
- ▶ Param7 through Param11: config assembly data size format, used in the assembly section by Assem100. These params specify the data format, bit fields, and resulting PLC tag names.
- ▶ Param7: creates a 16-bit (INT) controller tag, Start_Ramp.
- ▶ Param8: creates a 16-bit (INT) controller tag, Stop_Ramp.
- ▶ Param9: creates a pair of bit field (BOOL) controller tags, Selection_1_0 and Selection_1_1.
- ▶ Param10: creates a bit field (BOOL) controller tag, Selection_2_0.
- ▶ Param11: creates an 8-bit (byte) controller tag, Bitwise_Selection.
- ▶ Enum11: creates a set of named tags aliased for each bit in the bytes listed in this section.

The data type and data size fields in each param must match. See the data type field codes in the CIP Specification, Volume 1, Appendix C-6.

The minimum, maximum, and default fields are important to the RPI range param. The RPI is limited to a minimum of 2 ms, a maximum of 200 ms, and a default of 20 ms. These fields specify the range and default values for the configuration assembly data.

The assembly section references these params. Specifically look at the member size and reference pair of each Assemxxx item, which defines the size and format of the assembly item. Most importantly, note the use of these reference pairs in the Assem100 configuration assembly. This format is specified to the bit field level, and the user can insert pad fields to control where the configuration data appears in a particular byte. See the CIP Specification, Volume 1, Section 7-3.6.8.1.7 and Section 7-3.6.8.1.8.

Next, review the connection manager section, in which a single exclusive owner connection, Connection1, must be defined. Connection1 is defined in terms of the param and assem items in the following fields:

- ▶ O → T RPI
- ▶ O → T size
- ▶ O → T format
- ▶ T → O RPI
- ▶ T → O size
- ▶ T → O format
- ▶ Target config size
- ▶ Target config format

Table 32. Item 500, Configuration Item

Item Type	Configuration
Consume ID	100 (assembly instance ID in decimal)
Consume Size	6 bytes
Produce ID	0 (must be 0 for configuration assembly type)
Produce Size	0 (must be 0 for configuration assembly type)

Table 33. Item 501, Unique Item

Item Type	Unique
Consume ID	101 (assembly instance ID in decimal)
Consume Size	6 bytes
Produce ID	102 (assembly instance ID in decimal)
Produce Size	8 bytes

RPG2 I/O CONFIGURATION TOOL USER GUIDE

The final field in the Connection1 element, the path field, specifies the logical path (in hexadecimal) used by the protocol to specify the assembly object instance IDs used to create a Class 1 connection to the device. The 20 04 24 64 2C 65 2C 66 value is specified for this logical path. Refer to the CIP Specification, Volume 1, Section 3-5.5.1.12 for a detailed discussion of this field.

When defining the EDS file, review the following examples:

- ▶ The 20 04 is the application segment and denotes the assembly object class (04).
- ▶ The next three pairs of bytes represent the configuration, the consume (O → T) and the produce (T → O) assemblies, respectively.
- ▶ The first byte of each pair refers to the logical segment. The encoding of this byte is described in the CIP Specification in Volume 1, Appendix C, Section C-1.4.2. Byte 24 and Byte 2C are required here.
- ▶ The second byte of each pair is the desired assembly instance ID. In hexadecimal, these IDs are 64, 65, and 66. In decimal, these IDs are 100, 101 and 102.

Note that these assembly instance IDs correspond to the assembly instance IDs used in the module configuration data described in this user guide. Also, refer to the CIP Specification, Volume 1, Section 7-2.6.10.1.

To import the EDS file, use RSLogix 5000 software, Version 20 or above, from Rockwell Automation, to create a device and the necessary controller tags that correspond to these items.

ETHERCAT ESI FILES

This section describes the creation of an ESI to use with the network interface module for EtherCAT. For additional details, see the ETG.2000 document from the EtherCAT Technology Group website.

A sample set of system configuration and ESI files is part of the software zip file download from the ADIN2299 main product page at www.analog.com/adin2299. The configuration and ESI files follow the examples described in this user guide. The configuration data contains three items that describe the modular device profile (MDP) of the EtherCAT device. These three items correspond with the sample ESI file included in the EtherCAT software download.

The **ECAT_example_cfg** (no file extension) and **ECAT_example_cfg.bin** files contain the example configuration data. The first file is the raw data created and consumed by this configuration tool, and the second file contains the same data in a format that the RPG2 reference design can consume. This configuration tool automatically creates the .bin file when the configuration file in the same directory as this configuration tool.

The **Analog_Devices_Sample_ESI.xml** file contains the example ESI file.

This configuration tool creates the configuration file as described in this user guide. After the configuration file creates, the file loads onto the network interface module, containing data that describes the input and output items and the physical device. Each configuration entity, for example, items and devices, has an assigned unique ID number when entered into this configuration tool. The module runtime uses these ID numbers to locate the device settings and input and output data of the item. The ID numbers are arbitrary, or these numbers can be assigned by the user in any way desired. However, the values must be unique.

When the device and the necessary items are created, this configuration tool places the data in a basket (also contained in the file). The host software only references this basket ID (ID 1000). The example configuration provides a single device (ID 400) and three input and output items (Item 500, Item 501, and Item 502).

[Table 34](#) describes the input and output details of the example configuration.

Table 34. Input and Output Item Details for Example Configuration

Item ID Number	Input Size (Bytes)	Output Size (Bytes)
Item 500	2	2
Item 501	4	4
Item 502	Not applicable	2

Device Modification With the ESI Files

Users can use the example ESI file as a baseline and modify the file to suit the application and design requirements. To create a suitable ESI file, modify the following elements.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Modifying the Device Parameters

To create a custom ESI file based on the device configuration made with this configuration tool, modify the following **EtherCATInfo** parameter and subparameters.

Vendor

To set the vendor ID, set ID to the vendor ID from the device structure of the configuration data. The vendor ID for Analog Devices is #x00ADCAFE and differs from the vendor ID of the user.

Descriptions > Groups > Group

Set the following:

- ▶ Set the **SortOrder** attribute to the ordinal value that indicates where in the list of device groups this group must appear (for example, 0 for first, 1 for second).
- ▶ Set the **Type** element to a brief string that describes this group of devices.
- ▶ Set the **Name** element to the desired general name for the group (for example, RapID_NI).
- ▶ Set the **ImageData16x14** element to the hex data string that describes the desired icon for this device group.

Descriptions > Devices > Device > Type

Set the following:

- ▶ Set the **ProductCode** to the product code from the device structure of the configuration data. Use the #x prefix to indicate a hexadecimal number.
- ▶ Set the **RevisionNo** to the major revision number, shifted up 16 bits and OR'ed with the minor revision number. For example, if the major revision is 2 and the minor revision is 10, the **RevisionNo** is 0x0002000A. Use the #x prefix to indicate a hexadecimal number.
- ▶ Set the value of this element to the order ID of the device. Note that the order ID is not contained anywhere in the configuration data, and that the order ID is arbitrary and selected by the manufacturer to logically group devices.

Descriptions > Devices > Device > Name

Set the **Name** text to the device name from the device structure of the configuration data.

Descriptions > Devices > Device > GroupType

Set the **Type** text to the **Group Type** from the device structure of the configuration data.

Figure 152 shows the location to modify the device descriptions.

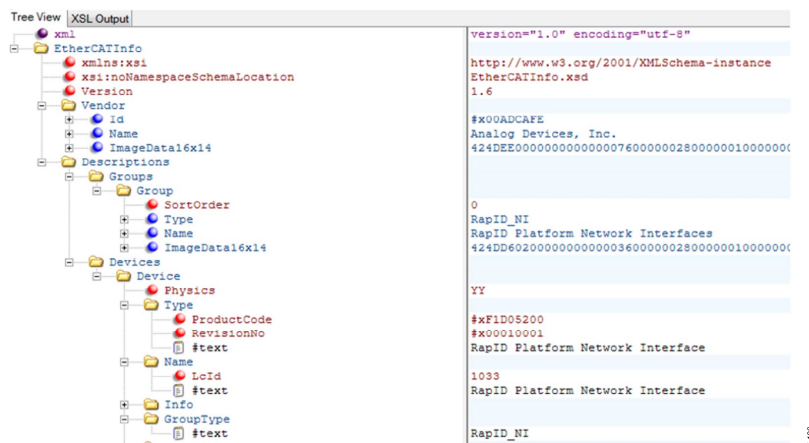


Figure 152. ESI Device, Tree View

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Modifying the Item Parameters

When integrating item structure data, all modifications occur in the device element (for example, profile, TxPdo, or RxPdo) rather than a module element. This procedure assumes that a basic element structure already exists and that the user is proceeding with modification. See the *ETG.2000 EtherCAT Slave Information (ESI) Specification* from the EtherCAT Technology Group for information on the structure of a **Device** element (or the ESI file as a whole) if a basic element structure does not exist and must be created.

The procedure described in this user guide is atypical because defining a **TxPdo > Entry** or **RxPdo > Entry** element requires that the name of the corresponding data type be supplied.

The instructions in the following example refer to the **DataType Dictionary**, the **Object Dictionary**, and the **TxPDO/RxPDO** list.

The **DataType Dictionary** is all of the elements under **EtherCATInfo > Descriptions > Devices > Device > Profile > Dictionary > DataTypes**. The **Object Dictionary** is all of the elements under **EtherCATInfo > Descriptions > Devices > Device > Profile > Dictionary > Objects**. The **TxPDO/ RxPDO** list is all of the **TxPdo** or **RxPdo** type elements under **EtherCATInfo > Descriptions > Devices > Device**.

DataType Dictionary Overview

Base **DataTypes** are the building blocks (unsigned byte, signed integer) for other more complicated data structures (such as, input and output process data object data types, and configuration and diagnostic data types).

The base **DataTypes** typically define the type for a single subindex within an object.

Object **DataTypes** are custom data types that use the base **DataTypes** to define the structure of an object.

[Table 35](#) details the supported base **DataTypes**. Keep the base **DataTypes** together at the top of the list of **DataTypes**.

Table 35. Supported Base DataTypes

Configuration Tool Name	Description	DataType Name (ESI File)	Remark
Unused	Unused subindex	Not applicable	Not applicable
INT8	8-bit signed value	SINT	Not applicable
INT16	16-bit signed value	INT	Not applicable
INT32	32-bit signed value	DINT	Not applicable
INT64	64-bit signed value	LINT	Not applicable
UINT8	8-bit unsigned value	USINT	Not applicable
UINT16	16-bit unsigned value	UINT	Not applicable
UINT32	32-bit unsigned value	UDINT	Not applicable
UINT64	64-bit unsigned value	ULINT	Not applicable
REAL32	32-bit floating point	REAL	Not applicable
BITARR8	Bit array (8 bits)	BITARR8	Not applicable
BITARR16	Bit array (16 bits)	BITARR16	Not applicable
BITARR32	Bit array (32 bits)	BITARR32	Not applicable
INT8ARR	Array of signed 8-bit values	ARRAY[0..n] of SINT	Where n is the length of the array minus 1
INT16ARR	Array of signed 16-bit values	ARRAY[0..n] of INT	Where n is the length of the array minus 1
INT32ARR	Array of signed 32-bit values	ARRAY[0..n] of DINT	Where n is the length of the array minus 1
UINT8ARR	Array of unsigned 8-bit values	ARRAY[0..n] of BYTE	Where n is the length of the array minus 1 (note that this is an array of bytes)
UINT16ARR	Array of unsigned 16-bit values	ARRAY[0..n] of UINT	Where n is the length of the array minus 1
UINT32ARR	Array of unsigned 32-bit values	ARRAY[0..n] of UDINT	Where n is the length of the array minus 1

Profile > Dictionary > DataTypes (Subldx > DataTypes)

The following list defines the **Profile > Dictionary > DataTypes (Subldx, DataTypes)** for each of the subindex data types used:

- If the subindex uses a nonarray type, the **DataType** is already provided, and no work is necessary. All supported nonarray data types are predefined in the example **DataType Dictionary** and do not require definition by the user.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

- ▶ If the subindex uses an array type, an example array **DataType** is provided and can require modification. Examples of supported array **DataTypes** are given. However, because it is impossible to know the final array size for any application at the example stage, the examples can require modification or duplication to fit the requirements of the application.
- ▶ Confirm that an array type definition in the **DataType Dictionary** already exists and matches the subindex described as follows:
- ▶ If the array type definition exists and matches the subindex, no work is necessary.
- ▶ If the array type definition does not exist and/or match the subindex, find an adequate array **DataType** definition with the proper base type, for example, UINT, and duplicate the array.
- ▶ Modify the duplicated array **DataType** to the proper length by doing the following:
- ▶ Set the **Name** element to indicate the length of the new **DataType**, for example, ARRAY [0..7] of UINT to ARRAY [0..5] of UINT to modify the array from an 8 element UINT array to a 6 element UINT array.
- ▶ Set the **BitSize** element to indicate the number of bits in the new **DataType**, in other words, the number of bits in the **BaseType** × the number of elements in the array.
- ▶ Set the **ArrayInfo < Elements** element to indicate the number of elements in the array. Note that the number of elements in the array is not the same as the number of bytes in the array.

Figure 153 shows where to modify the **DataTypes** portion of the ESI file.

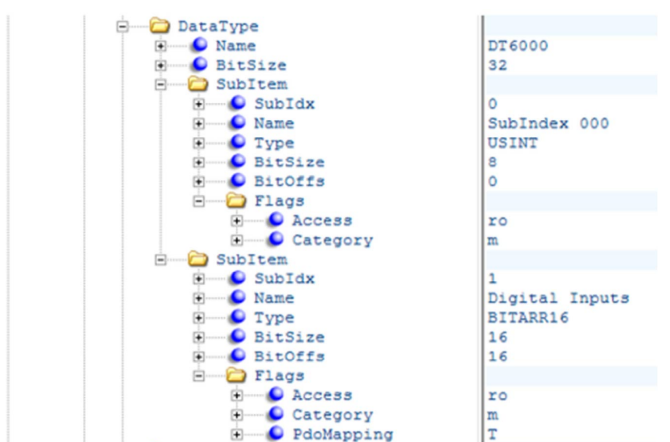


Figure 153. EtherCAT Device Expanded View

Device > Profile > Dictionary > DataTypes (Input, Output, Configuration, and Diagnostic Object DataTypes)

For each data item in use, take the following steps to modify the ESI file:

1. If the item has input data associated with it, create or modify a **DataType** to describe the structure of the input object.
2. Set the name element to reflect the name of the **DataType**, as well as the object it represents, for example, DT6000.

The name of these **DataTypes** indicate the index of the object they describe, for example, DT6000 to describe Object 0x6000.

The index of the object depends on the type of object and the order that the software processes the object.

Input data object indices start at 0x6000 and increase by 0x0010 for each new object.

For example, the first input data object installed by the software has an index of 0x6000, and the second input data object installed by the software has an index of 0x6010.

3. Set the **BitSize** element to the sum of the following:
 - a. 8 (for the size of Subindex 0 to a USINT value).
 - b. 8 (for an alignment byte between Subindex 0 and Subindex 1 to keep the subindices 16-bit aligned).
 - c. The total bit size of all of the input data this object represents, for example, four subindices each representing UINT values has a bit size of 4 bits × 16 bits = 64 bits).
4. Create or modify the first **SubItem** element (under the **DataType** element) and set it as follows:

RPG2 I/O CONFIGURATION TOOL USER GUIDE

- a. **SubIdx**: 0
 - b. **Name**: SubIndex 000
 - c. **Type**: USINT
 - d. **BitSize**: 8
 - e. **BitOffs**: 0
 - f. Set the **Flags** element as follows:
 1. **Access**: ro (read only).
 2. **Category**: m (mandatory).
5. For each of the subindices defined in the input data portion of the configuration data item, create or modify a **SubItem** element following the first one to describe the subindex as follows:
- a. If the subindex is unused, omit the **SubItem** element.
 - b. Set the **SubIdx** element to the subindex number this element describes.
 - c. Set the **Name** element to the name of the corresponding subindex in the input portion of the configuration data item.
 - d. Set the **Type** element to the name of the **DataType** previously created for this subindex.
 - e. Set the **BitSize** element to the size of this subindex (in bits), which must be the same as the **BitSize** from the **DataType** previously created for this subindex.
 - f. Set the **BitOffs** element to the sum of the **BitSize** elements of all of the previous **SubItem** elements in this **DataType** list, plus eight to account for the alignment byte.
 - g. Set the **Flags** element as follows:
 1. **Access**: ro (read only).
 2. **Category**: m (mandatory).
 3. **PdoMapping**: T for input data object.
6. If the item has output data associated with it, create or modify a **DataType** element to describe the structure of the output object.
7. Set the **Name** element to reflect the name of the **DataType** element, as well as the object the element represents, for example, DT7000.

The name of these **DataType** elements indicate the index of the object the elements describe, for example, DT7000 to describe Object 0x7000.

The index of the object depends on the type of object and the order that the software processes the object.

The output data object indices start at 0x7000 and increase by 0x0010 for each new object.

For example, the first output data object installed by the software has an index of 0x7000, and the second output data object installed by the software has an index of 0x7010.

8. Set the **BitSize** element to the sum of the following:
- a. 8 (for the size of Subindex 0 to a USINT value).
 - b. 8 (for an alignment byte between Subindex 0 and Subindex 1 to keep the subindices 16-bit aligned).
 - c. The total bit size of all of the output data this object represents, for example, four subindices each representing UINT values have a bit size of $4 \text{ bits} \times 16 \text{ bits} = 64 \text{ bits}$.
9. Create or modify the first **SubItem** element (below the **DataType** element) and set the element as follows:
- a. **SubIdx**: 0
 - b. **Name**: SubIndex 000
 - c. **Type**: USINT
 - d. **BitSize**: 8
 - e. **BitOffs**: 0
 - f. Set the **Flags** element as follows:
 1. **Access**: ro (read only).
 2. **Category**: m (mandatory).

RPG2 I/O CONFIGURATION TOOL USER GUIDE

10. For each of the subindices defined in the output data portion of the configuration data item, create or modify a **SubItem** element following the first element to describe the subindex as follows:
- If the subindex is unused, omit the **SubItem** element for that subindex.
 - Set the **SubIdx** element to the subindex number this element describes.
 - Set the **Name** element to the name of the corresponding subindex in the output portion of the configuration data item.
 - Set the **Type** element to the name of the previously created **DataType** element for this subindex.
 - Set the **BitSize** element to the size of this subindex in bits. This value must match the **BitSize** element from the previously created **DataType** element values for this subindex.
 - Set the **BitOffs** element to the sum of the **BitSize** elements of all of the previous **SubItem** elements in this **DataType** list, plus eight to account for the alignment byte.
 - Set the **Flags** element as follows:
 - Access:** ro (read only).
 - Category:** m (mandatory).
 - PdoMapping:** R for output data object.

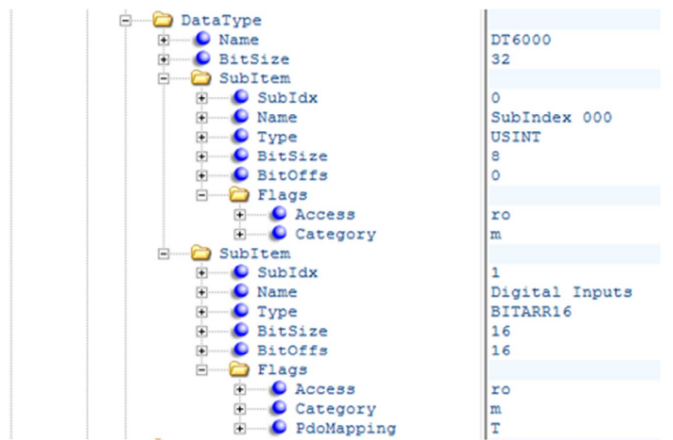


Figure 154. SubIndex, Tree View

For each configuration item in use, take the following steps to modify the ESI file:

- Set the **Name** element to reflect the name of the **DataType** element, as well as the object this element represents, for example, DT8000.

The name of these **DataType** elements indicate the index of the object they describe, for example, DT8000 to describe Object 0x8000.

The index of the object depends on the type of object and the order in which the software processes the object.

Configuration data object indices start at 0x8000 and increase by 0x0010 for each new object.

For example, the first configuration data object installed by the software has an index of 0x8000 and the second configuration data object installed by the software has an index of 0x8010.
- Set the **BitSize** element to the sum of the following:
 - 8 (for the size of Subindex 0 to a USINT value).
 - 8 (for an alignment byte between Subindex 0 and Subindex 1 to keep the subindices 16-bit aligned).
 - The total bit size of all of the configuration data that this object represents, for example, two subindices that each represent USINT values and have a bit size of 2 bits × 8 bits = 16 bits.
- Create or modify the first **SubItem** element (under the **DataType** element) and set it as follows:
 - SubIdx:** 0
 - Name:** SubIndex 000
 - Type:** USINT

RPG2 I/O CONFIGURATION TOOL USER GUIDE

- d. **BitSize**: 8
- e. **BitOffs**: 0
- f. Set the **Flags** element as follows:
 1. **Access**: ro (read only).
 2. **Category**: m (mandatory).
4. For each of the subindices defined in the configuration data portion of the configuration data item, create or modify a **SubItem** element following the first element to describe the subindex as follows:
 - a. If the subindex is unused, omit the **SubItem** element for it.
 - b. Set the **SubIdx** element to the subindex number this element describes.
 - c. Set the **Name** element to the name of the corresponding subindex in the configuration portion of the configuration data item.
 - d. Set the **Type** element to the name of the previously created **DataType** for this subindex.
 - e. Set the **BitSize** element to the size of this subindex in bits, which must be the same as the **BitSize** element of the previously created **DataType** element for this subindex.
 - f. Set the **BitOffs** element to the sum of the **BitSize** elements of all of the previous **SubItem** elements in this **DataType** list, plus eight to account for the alignment byte.
 - g. Set the **Flags** element as follows:
 1. **Access**: ro (read only).
 2. **Category**: m (mandatory).

For each diagnostic item used, make the following edits to the ESI file:

1. Set the **Name** element to reflect the name of the **DataType** element, as well as the object that the element represents, for example, DTA000.
 The names of these **DataType** elements indicate the index of the object they describe, for example, DTA000 to describe Object 0xA000.
 The index of the object depends on the type of object and the order in which the software processes the object.
 Diagnostic data object indices start at 0xA000 and increase by 0x0010 for each new object.
 For example, the first diagnostic data object installed by the software has an index of 0xA000, and the second diagnostic data object installed by the software has an index of 0xA010.
2. Set the **BitSize** element to the sum of the following:
 - a. A (for the size of Subindex 0 to a USINT value)
 - b. A (for an alignment byte between Subindex 0 and Subindex 1 to keep the subindices 16-bit aligned).
 - c. The total bit size of all of the diagnostic data this object represents, for example, two subindices that each represent USINT values have a bit size of $2 \text{ bits} \times 8 \text{ bits} = 16 \text{ bits}$.
3. Create or modify the first **SubItem** element (under the **DataType** element) and set it as follows:
 - a. **SubIdx**: 0
 - b. **Name**: SubIndex 000
 - c. **Type**: USINT
 - d. **BitSize**: A
 - e. **BitOffs**: 0
 - f. Set the **Flags** element as follows:
 1. **Access**: ro (read only).
 2. **Category**: m (mandatory).
4. For each of the subindices defined in the diagnostic data portion of the diagnostic data item, create or modify a **SubItem** element following the first element to describe the subindex as follows:
 - a. If the subindex is unused, omit the **SubItem** element for the subindex.
 - b. Set the **SubIdx** element to the subindex number this element describes.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

- c. Set the **Name** element to the name of the corresponding subindex in the diagnostic portion of the diagnostic data item.
- d. Set the **Type** element to the name of the previously created **DataType** element for this subindex.
- e. Set the **BitSize** element to the size of this subindex in bits. This value must match the **BitSize** element value of the previously created **DataType** element for this subindex.
- f. Set the **BitOffs** element to the sum of the **BitSize** elements of all of the previous **SubItem** elements in this **DataType** list, plus A to account for the alignment byte.
- g. Set the **Flags** element as follows:
 1. **Access**: ro (read only).
 2. **Category**: m (mandatory).

Device > Profile > Dictionary > Objects (TxPDO Map and RxPDO Map Object DataTypes)

The process data object (PDO) subindices larger than 30 bytes require a subindex in the corresponding RxPDO or TxPDO map object to describe each 30 byte block in the PDO subindex. For example, if PDO has one subindex that is 512 bytes in size, the RxPDO map **DataType** element must define 19 **SubItem** elements: 1 to represent Subindex 0 of the PDO, 17 **SubItem** elements to represent the first 510 bytes (17 **SubItem** elements × 30 bytes) of the PDO, and 1 to represent the last two bytes of the PDO. Use this approach because each RxPDO/TxPDO map entry only contains a 1 byte field to describe the size of PDO subindex. As a result, each RxPDO/TxPDO map object entry describes the PDO subindices in 30 byte blocks.

For each data item used, make the following edits to the ESI file:

1. Set the **Name** element to reflect the name of the **DataType**, as well as the object the element represents, for example, DTA000.

The index of a TxPDO map object depends on how many input data objects with Index 0x6000 or greater are installed, and the order in which these objects are installed.

The TxPDO map object indices start at 0x1A00 and increase by 0x0001 for each new object.

If an item does not have input data, a TxPDO map object is not required, and the index for that TxPDO map object is skipped.

If adding an output only item, and then an input and output item, it is not required to describe Object 0x1A00, and the system does not add this object to the online object dictionary. In addition, the TxPDO map object data types begin at Index 0x1A01.

2. Set the **BitSize** element to the sum of the following:

- a. 8 (for the size of Subindex 0 to a USINT value).
- b. 8 (for an alignment byte between Subindex 0 and Subindex 1 to keep the subindices 16-bit aligned).
- c. The total bit size of all of the used TxPDO map entries.

Typically, one TxPDO map entry describes one subindex for base **DataType** elements from the corresponding input data object.

If using an array **DataType** element in the input data object, and the array is larger than 30 bytes, it is required to use more than one TxPDO map entry to describe the large subindex.

Typically, to calculate the bit size of all of the used TxPDO map entries for a single subindex larger than 30 bytes, use the following equation:

$$(32 \times (\text{floor}((\text{PDO Subindex Size in Bytes}) \div 30) + 1))$$

For example, the bit size of all of the TxPDO map entries used for a 64-byte subindex and a 512-byte subindex follows:

$$(32 \times (\text{floor}(64 \text{ bytes} \div 30) + 1)) + (32 \times (\text{floor}(512 \text{ bytes} \div 30) + 1)) = (32 \times 3) \text{ bytes} + (32 \times 18) \text{ bytes} = 96 \text{ bytes} + 576 \text{ bytes} = 672 \text{ bytes}$$

3. Create or modify the first **SubItem** element (under the **DataType** element) and set it as follows:

- a. **SubIdx**: 0
- b. **Name**: SubIndex 000
- c. **Type**: USINT
- d. **BitSize**: 8
- e. **BitOffs**: 0

RPG2 I/O CONFIGURATION TOOL USER GUIDE

- f. Set the **Flags** element as **Access**: ro (read only)
4. For each used TxPDO map entry, following the first element, create or modify a **SubItem** element to describe the TxPDO map entry as follows:
 - a. **SubIdx**: TxPDO map entry number described (1-based).
For example, set the entry number to 2 for the second 30 byte block of the 64 byte subindex described by the TxPDO map, Entry 1 to Entry 3, or for an additional example, set the entry number to 6 for the third 30 byte block of the 512 byte subindex described by the TxPDO map, Entry 4 to Entry 21.
 - b. **Name**: SubIndex xxx, where xxx is the TxPDO map described, for example, SubIndex 003 for the third TxPDO map entry described.
 - c. **Type**: USINT.
 - d. **BitSize**: 32.
 - e. Set the **BitOffs** element to the sum of the **BitSize** elements of all of the previously created **SubItems** in this **DataType** list, plus eight to account for the alignment byte. For example, the **BitOffs** element value for the fourth **SubItem** element is 8 bits + 32 bits + 32 bits + 8 bits = 80 bits.
 - f. Set the **Flags** element as **Access**: ro.

If the item has output data associated with it, create or modify a **DataType** element to describe the structure of the RxPDO map object as follows:

1. Set the **Name** element to reflect the name of the **DataType** element, as well as the object the element represents, for example, DT1600.

The index of an RxPDO map object depends on how many output data objects are installed, and the order in which the objects are installed.

RxPDO map object indices start at 0x1600 and increase by 0x0001 for each new object.

If an item does not have output data, an RxPDO map object is not required, and the index for that RxPDO map object is skipped.

If adding an input only item and then an input and output item to the system, it is not required to describe Object 0x1600, and the system does not add this object to the online object dictionary.

In addition, the RxPDO map object data types begin with an index of 0x1601.

2. Set the **BitSize** element to the sum of the following:
 - a. 8 (for the size of Subindex 0 to a USINT value).
 - b. 8 (for an alignment byte between Subindex 0 and Subindex 1 to keep the subindices 16-bit aligned).
 - c. Total bit size of all of the used RxPDO map object entries.

Typically, one RxPDO map entry describes one subindex from the corresponding output data object.

If using an array **DataType** in the output data object, and the array is larger than 30 bytes, use more than one RxPDO map entry to describe the large subindex.

Typically, to calculate the bit size of all RxPDO map entries used for a single subindex larger than 30 bytes, use the following equation:

$$(32 \times (\text{floor}((\text{PDO Subindex Size in Bytes}) \div 30) + 1))$$

For example, the bit size of all RxPDO map entries used for a 64 byte subindex and a 512 byte subindex follows:

$$(32 \times (\text{floor}(64 \div 30) + 1)) + (32 \times (\text{floor}(512 \div 30) + 1)) = (32 \times 3) + (32 \times 18) = 96 + 576 = 672$$

3. Create or modify the first **SubItem** element (under the **DataType** element) and set this element as follows:
 - a. **SubIdx**: 0
 - b. **Name**: SubIndex 000
 - c. **Type**: USINT
 - d. **BitSize**: 8
 - e. **BitOffs**: 0
 - f. Set the **Flags** element as **Access**: ro (read only)

RPG2 I/O CONFIGURATION TOOL USER GUIDE

For each previously created RxPDO map entry, take the following steps:

1. Create or modify a **SubItem** element following the first element to describe the RxPDO map entry as follows:
 - a. **SubIdx**: RxPDO map entry number described (1-based). For example, set an entry number of 2 for the second 30 byte block of the 64 byte subindex described by the RxPDO map, Entry 1 to Entry 3, or as another example, set a value of 6 for the third 30 byte block of the 512 byte subindex described by the RxPDO map, Entry 4 to Entry 21.
 - b. **Name**: SubIndex xxx, where xxx is the RxPDO map entry described, for example, SubIndex 003, for the third RxPDO map entry described.
 - c. **Type**: UDINT
 - d. **BitSize**: 32
 - e. Set the **BitOffs** element to the sum of the **BitSize** elements of all of the previously created **SubItem** elements in this **DataType** list, plus eight to account for the alignment byte. For example, the sum of the **BitOffs** element for the fourth **SubItem** element is 8 bits + 32 bits + 32 bits + 8 bits = 80 bits.
 - f. Set the **Flags** element as **Access**: ro.

Note that the TxPDO and RxPDO map objects are not required for configuration or diagnostic data objects.

Device > Profile > Dictionary > DataTypes (TxPDO Assign and RxPDO Assign Object DataTypes)

The TxPDO and RxPDO assign objects contain lists of the TxPDO and RxPDO map objects used. Therefore, a storage **DataType** element must be defined for these objects.

To define the TxPDO assign object **DataType** element, take the following steps:

1. Create or modify a **DataType** element to describe the UINT array of the TxPDO assign object as follows:
 - a. Set the **Name** element to DT1C13ARR.
 - b. Set the **BitSize** element to 16× number of TxPDO map object **DataType** elements created.
 - c. Set the **ArrayInfo** elements to **LBound** = 0 and **Elements** to the number of TxPDO map object **DataTypes** created.
2. Create or modify a **DataType** element to describe the TxPDO assign object as follows:
 - a. Set the **Name** element to DT1C13.
 - b. Set the **BitSize** element to the sum of the following:
 1. 8 (for the size of Subindex 0 to a USINT value).
 2. 8 (for an alignment byte between Subindex 0 and Subindex 1 to keep the subindices 16-bit aligned).
 3. The **BitSize** of the DT1C13ARR **DataType** element.
3. Create or modify the first **SubItem** element (under the **DataType** element) and set it as follows:
 - a. **SubIdx**: 0
 - b. **Name**: SubIndex 000
 - c. **Type**: USINT
 - d. **BitSize**: 8
 - e. **BitOffs**: 0
 - f. Set the **Flags** element as follows:
 1. **Access**: ro (read only)
 2. **Category**: m
4. Create or modify an additional **SubItem** element following the element previously created, and set the element as follows:
 - a. **Name**: Elements
 - b. **Type**: DT1C12ARR
 - c. **BitSize**: size of the DT1C12ARR **DataType** element
 - d. **BitOffs**: 16
 - e. Set the **Flags** element as follows:

RPG2 I/O CONFIGURATION TOOL USER GUIDE

1. **Access:** ro (read only)
2. **Category:** m

To define an RxPDO assign object **DataType** element, take the following steps:

1. Create or modify a **DataType** element to describe the UINT array of the RxPDO assign object as follows:
 - a. Set the **Name** element to DT1C12ARR
 - b. Set the **BaseType** element to UINT
 - c. Set the **BitSize** element to $16 \times$ the number of RxPDO map object **DataType** elements.
 - d. Set the **ArrayInfo** elements to **LBound** = 0 and set **Elements** to the number of RxPDO map object **DataType** created.
2. Create or modify a **DataType** to describe the RxPDO assign object as follows:
 - a. Set the **Name** element to DT1C12.
 - b. Set the **BitSize** element to the sum of the following:
 1. 8 (for the size of Subindex 0 to a USINT value)
 2. 8 (for an alignment byte between Subindex 0 and Subindex 1 to keep the subindices 16-bit aligned)
 3. The **BitSize** element of the DT1C12ARR **DataType** element.
3. Create or modify the first **SubItem** element (under the **DataType** element) and set it as follows:
 - a. **SubIdx:** 0
 - b. **Name:** SubIndex 000
 - c. **Type:** USINT
 - d. **BitSize:** 8
 - e. **BitOffs:** 0
 - f. Set the **Flags** element as follows:
 1. **Access:** ro (read only)
 2. **Category:** m
4. Create or modify one additional **SubItem** element (following the most recently created **SubItem** element) and set it as follows:
 - a. **Name:** Elements
 - b. **Type:** DT1C13ARR
 - c. **BitSize:** size of the DT1C13ARR **DataType** element
 - d. **BitOffs:** 16
 - e. Set the **Flags** element as follows:
 1. **Access:** ro (read only)
 2. **Category:** m

Device > Profile > Dictionary > Objects (Input, Output, Configuration, and Diagnostic)

For each previously created input data object **DataType** element, take the following steps to edit the ESI file:

1. Create or modify an **Object** element to describe the input data object as follows:
 - a. Set the **Index** element to the hexadecimal string representation of the index of the input object, for example, #x6010 for Object 0x6010.
 - b. Set the **Name** element to the name of the input data object from the configuration data item.
 - c. Set the **Type** element to the name of the previously created **DataType** element for this object, for example, DT6010 for Object 0x6010.
 - d. Set the **BitSize** element to the same **BitSize** element as the **DataType** element referenced in this set of instructions.

For each previously created output data object **DataType** element, take the following steps to edit the ESI file:

1. Create or modify an **Object** element to describe the output data object as follows:
 - a. Set the **Index** element to the hexadecimal string representation of the index of the output object, for example, #x7020 for Object 0x7020.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

- b. Set the **Name** element to the name of the output data object from the configuration data item.
- c. Set the **Type** element to the name of the previously created **Data Type** element for this object, for example, DT7020 for Object 0x7020.
- d. Set the **BitSize** element to the same **BitSize** element as the **Data Type** element.

For each previously created configuration data object **Data Type** element, take the following steps to edit the ESI file:

1. Create or modify an **Object** element to describe the configuration data object as follows:
 - a. Set the **Index** element to the hexadecimal string representation of the index of the input object, for example, #x8000 for Object 0x8000.
 - b. Set the **Name** element to the name of the output object from the configuration data item.
 - c. Set the **Type** element to the name of the previously created **Data Type** for this object, for example, DT8000 for Object 0x8000.
 - d. Set the **BitSize** element to the same **BitSize** value as the **Data Type** element.

For each previously created diagnostic data object **Data Type** element, take the following steps to edit the ESI file:

1. Create or modify an **Object** element to describe the diagnostic data object as follows:
 - a. Set the **Index** element to the hexadecimal string representation of the index of the input object, for example, #xA040 for Object 0xA040.
 - b. Set the **Name** element to the name of the input object from the configuration data item.
 - c. Set the **Type** element to the name of the previously created **Data Type** element for this object, for example, DTA040 for Object 0xA040.
 - d. Set the **BitSize** element to the same **BitSize** value as the **Data Type** element.

Device > Profile > Dictionary > Objects (TxPDO Map and RxPDO Map)

For each previously created TxPDO map object **Data Type**, take the following steps to edit the ESI file:

1. Create or modify an **Object** element to describe the TxPDO map object as follows:
 - a. Set the **Index** element to the hexadecimal string representation of the index of the TxPDO map object, for example, #x1A00 for Object 0x0x1A00.
 - b. Set the **Name** element to the concatenation of the name of the input data object from the configuration object and the string TxPDO map. For example, if the name of the input data object is **Digital Ins**, the name of the TxPDO map object is Digital Ins TxPDO Map.
 - c. Set the **Type** element to the name of the previously created **Data Type** element for this object, for example, DT1A00 for Object 0x1A00.
 - d. Set the **BitSize** element to the same **BitSize** value as the **Data Type** element.

For each previously created RxPDO map object **Data Type**, take the following steps to edit the ESI file:

1. Create or modify an **Object** element to describe the RxPDO map object as follows:
 - a. Set the **Index** element to the hexadecimal string representation of the index of the RxPDO map object, for example, #x1601 for Object 0x0x1601.
 - b. Set the **Name** element to the concatenation of the name of the output data object from the configuration object and the string RxPDO map. For example, if the name of the output data object is **Analog Outs**, the name of the RxPDO map object is Analog Outs RxPDO Map.
 - c. Set the **Type** element to the name of the previously created **Data Type** element for this object, for example, DT1601 for Object 0x1601.
 - d. Set the **BitSize** element to the same **BitSize** value as the **Data Type** element.

Device > Profile > Dictionary > Objects (TxPDO Assign and RxPDO Assign)

The TxPDO and RxPDO assign objects remain largely unchanged. The only element in these objects that require modification is the **BitSize** element.

To modify the **BitSize** elements in the TxPDO and RxPDO assign objects, take the following steps:

1. In the **Object** element that corresponds to the TxPDO, change the **BitSize** element to the same value as the **BitSize** element in the DT1C13 **Data Type** element.
2. In the **Object** element that corresponds to the RxPDO, change the **BitSize** element to the same value as the **BitSize** element in the DT1C12 **Data Type** element.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Device > RxPdo

For each input data object previously created, take the following steps to edit the ESI file:

1. Create or modify a **TxPdo** element as follows:
 - a. Set the **Fixed** attribute to 1.
 - b. Set the **Mandatory** attribute to 1.
 - c. Set the **Sm** attribute to 3.
 - d. Set the **Index** element to the index of the TxPDO map object, for example, #x1A01 for Object 0x1A01, which contains the input object information.

The TxPDO map object of Input Object 0x6000 is Object 0x1A00, the TxPDO map object of Input Object 0x6010 is Object 0x1A01, and the TxPDO map object of Input Object 0x6020 is Object 0x1A02.
 - e. Set the **Name** element to the name of the input object from the configuration data item.

For each subindex in the input data object, take the following steps to edit the ESI file:

1. Create or modify an **Entry** element.
2. If the subindex is unused, take the following steps:
 - a. Set the **Index** element to 0.
 - b. Set the **SubIndex** element to the subindex number currently described.
 - c. Set the **BitSize** element to the same value as the **BitSize** element of the **DataType** element that describes this subindex.
 - d. Set the **Name** element to the subindex name from the configuration data item.
 - e. Omit the **DataType** element.
3. If the subindex is 30 bytes in size or less, take the following steps:
 - a. Set the **Index** element to the index of the input data object, for example, #x6010 for Object 0x6010.
 - b. Set the **SubIndex** element to the subindex number currently described.
 - c. Set the **BitSize** element to the same value as the **BitSize** element of the **DataType** element that describes this subindex.
 - d. Set the **Name** element to the subindex name from the configuration data item.
 - e. Set the **DataType** element to the name of the **DataType** element that describes this subindex.
4. If the subindex is more than 30 bytes in size, take the following steps:
 - a. Set the **Index** element to the index of the output data object, for example, #x6010 for Object 0x6010.
 - b. Set the **SubIndex** element to the subindex number currently described.
 - c. Set the **BitSize** element to 240.
 - d. Set the **Name** element to the subindex name from the configuration data item.
 - e. Set the **DataType** element to the name of the **DataType** element that describes this subindex.
 - f. For $(\text{Floor}((\text{Subindex Size in Bytes}) \div 30))$ times, create another **Entry** element description that sets the **Entry** element as follows:
 1. **Index**: 0.
 2. **SubIndex**: 0.
 3. **BitLen**: 240. If this is the last **Entry** element, change the **BitLen** element to $((\text{Subindex Size in Bytes}) \div 30) \times 8$.

For example, for a 512 byte subindex, the **BitLen** value of the last **Entry** element (for example, the 18th **Entry** element) is $(512 \div 30) \times 8 = (2 \times 8) = 16 \times$.

If the subindex size is evenly divisible by 30, the **BitLen** element of the last **Entry** element is 0 (required).

Device > TxPdo

For each previously created input data object, take the following steps to edit the ESI file:

1. Create or modify an **RxPdo** element.
 - a. Set the **Fixed** attribute to 1.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

- b. Set the **Mandatory** attribute to 1.
- c. Set the **Sm** attribute to 3.
- d. Set the **Index** element to the index of the RxPDO map object, for example, #x1601 for Object 0x1601, which contains the input object information.

The RxPDO map object of Input Object 0x7000 is Object 0x1600, the RxPDO map object of Input Object 0x7010 is Object 0x1601, and the RxPDO map object of Input Object 0x7020 is Object 0x1602.

- e. Set the **Name** element to the name of the input object from the configuration data item.

For each subindex in the input data object, take the following steps to edit the ESI file:

1. Create or modify an **Entry** element.
2. If the subindex is unused, take the following steps:
 - a. Set the **Index** element to 0.
 - b. Set the **SubIndex** element to the subindex number currently described.
 - c. Set the **BitSize** element to the same value as the **BitSize** element of the **DataType** element that describes this subindex.
 - d. Set the **Name** element to the subindex name from the configuration data item.
 - e. Omit the **DataType** element.
3. If the subindex is 30 bytes in size or less, take the following steps:
 - a. Set the **Index** element to the index of input data object, for example, #x7010 for Object 0x7010.
 - b. Set the **SubIndex** element to the subindex number currently described.
 - c. Set the **BitSize** element to the same value as the **BitSize** element of the **DataType** element that describes this subindex.
 - d. Set the **Name** element to the subindex name from the configuration data item.
 - e. Set the **DataType** element to the name of the **DataType** element that describes this subindex.
4. If the subindex is more than 30 bytes in size, take the following steps:
 - a. Set the **Index** element to the index of output data object, for example, #x7010 for Object 0x7010.
 - b. Set the **SubIndex** element to the subindex number currently described.
 - c. Set the **BitSize** element to 240.
 - d. Set the **Name** element to the subindex name from the configuration data item.
 - e. Set the **DataType** element to the name of the **DataType** element that describes this subindex.
 - f. For $(\text{Floor}((\text{Subindex Size in Bytes})/30))$ times, create another entry description that sets the **Entry** element as follows:
 1. **Index**: 0.
 2. **SubIndex**: 0.
 3. **BitLen**: 240. If this is the last entry element, change the **BitLen** element to $((\text{Subindex Size in Bytes}) \div 30) \times 8$.

For example, for a 512 byte subindex, the **BitLen** value of the last entry (in other words, the 18th **Entry** element) is $(512 \div 30) \times 8 = (2 \times 8) = 16$.

If the subindex size is evenly divisible by 30, the **BitLen** of the last **Entry** element is 0 (required).

Device > Sm

EtherCAT devices have four synchronization managers (**Sm** elements) to handle network communications. The last two synchronization managers are specific to the input and output data. Therefore, the user must modify the size to reflect the configuration data produced by this configuration tool.

Take the following steps to modify the size of the **Sm** elements:

1. Modify the third **Sm** element. Set the **DefaultSize** attribute to the total size of all of the output data covered by all output data objects (in bytes).
2. Modify the fourth **Sm** element. Set the **DefaultSize** attribute to the total size of all of the input data covered by all input data objects (in bytes).

RPG2 I/O CONFIGURATION TOOL USER GUIDE

Icon Files

The text inside all **ImageData16x14** elements is a string of hexadecimal characters that represent the data inside a 16 pixel × 14 pixel bitmap image.

To create icon data for an **ImageData16x14** element, create the image from scratch or modify an existing image.

To create the image from scratch take the following steps:

1. Create a bitmap image that uses 16 colors and is 16 pixels wide by 14 pixels high in an image editing application, such as Microsoft Paint.
2. Open the bitmap image in a hex editor.
3. Copy the hexadecimal representation of the data as a string into the **ImageData16x14** element (with no spaces).

To modify an existing image, take the following steps:

1. Create a new file in a hex editor.
2. Copy the existing **ImageData16x14** text for the icon to modify.
3. Paste the string into the hex editor.
4. Save the file with a .bmp extension.
5. Open the created .bmp with an image editing application, such as Microsoft Paint.
6. Edit the image as desired.
7. Save the image.
8. Open the image in a hex editor.
9. Copy the hexadecimal representation of the data as a string into the **ImageData16x14** element (with no spaces).

POWERLINK XDD FILES

This section describes the creation of an XDD file to use with the POWERLINK module. A sample set of system configuration and XDD files is part of the software zip file download from the ADIN2299 main product page at www.analog.com/adin2299. The configuration file follows the examples described in this user guide, and the XDD file was created to reflect that example. The configuration data contains three items that describe the modular device profile (MDP) of the POWERLINK device. These three items correspond with the sample XDD file included in the POWERLINK software download (see the [main product page](#)).

The **plink_std_demo** (no file extension) and **plink_std_demo.bin** files contain the example configuration data. The **plink_std_demo** file is the raw data created and consumed by this configuration tool, and the **plink_std_demo.bin** file contains the same data as the **plink_std_demo** file in a format that the network module bootloader can consume. The **fido-none-elf-binarygen.exe** tool processes the raw configuration tool data into the .bin file format. This configuration tool automatically creates the .bin file in the same directory as this configuration tool.

The **0xffff0012_RapID_POWERLINK_STD_DEMO_CN.xdd** file contains the example XDD file.

This configuration tool creates the configuration file described in this user guide. After the configuration file is created, the bootloader uploads the file to the network interface module. The configuration file contains data that describes both the input and output items and the actual device itself. Each configuration entity (device, item) has a unique ID number when entering the data in the configuration file into this configuration tool. The module runtime uses these ID numbers to locate the device settings and the input and output data of the item. The ID numbers are arbitrary, or these numbers can be assigned by the user in any way required. However, the ID numbers must be unique.

When the device and all the necessary items are created, this configuration tool places the data in a basket (also contained in the file). The host software only references this basket ID. The example configuration provides a single device (ID 400) and three input and output items (Item 500, Item 501, and Item 502). In this example, all of this information is placed into a basket with the ID of 1000.

The default XDD file resembles the standard configuration database (see [Table 36](#)).

Device Modification With the POWERLINK XDD Files

Users can begin device modification with the example XDD file as a baseline and modify the example file to suit the needs of their application. Most of the information in the XDD file does not require modification because the user device behaves in the same manner as the sample device with respect to POWERLINK network behavior.

RPG2 I/O CONFIGURATION TOOL USER GUIDE

A POWERLINK device for the RapID platform is preloaded and defines device data as it applies to the RapID platform device. Modify the example XDD file with the needed device parameters. The device data is mostly identity related.

The following parameters require modification because these parameters are specific to the device of the user, and the sample defaults contain values that are specific to Analog Devices:

- ▶ Vendor ID: this value distributes as 0xFFFF0012, which must be modified to the specific vendor ID of the user. This ID number is in the **Device Identity Field** of the XDD file and in the **Communication Profile Area**. The POWERLINK organization assigns the vendor ID.
- ▶ Product code: by default, the device ships with a product code of 0xf1d05200. The user changes this product code in the **Communications Profile Area** of the XDD file to a value that is specific to the product/application of the user.
- ▶ Revision number: change this number in the **Communications Profile Area**.
- ▶ Serial number: change this number in the **Communications Profile Area**.
- ▶ Device name: change this number in the **Communications Profile Area**. The device ships with a default device name, RapID POWERLINK device. Rename the device to reflect the device of the user.
- ▶ Hardware version: change the version in the **Communications Profile Area** and the **Device Identity Field**.
- ▶ Software version: change the version in the **Communications Profile Area** and the **Device Identity Field**.

Table 36. Sample POWERLINK Input and Output Parameters

Item Number	Item Type	Item Size (Bits)	Input Quantity	Output Quantity
Item 500	Digital	8	2	2
Item 501	Analog	16	2	2
Item 502	Digital	8	0	2

Item Modification With the POWERLINK XDD Files

The only location to be modified for the XDD file exists in the **Standardized Device Profile Area**. The current entries that correspond to the default configuration file appear as shown in the following code example.

```
<!-- Standardised Device Profile Area (0x6000 - 0x9FFF): may be used according to a CiA device
profile. The profile to be used is given by NMT_DeviceType_U32 -->

<Object index="6000" name="DigitalInput_00h_AU8" objectType="8" dataType="0005">
<SubObject subIndex="00" name="NumberOfEntries" objectType="7" dataType="0005" accessType="const" de
faultValue="2" PDOmapping="no"/>
<SubObject subIndex="01" name="DigitalInput" objectType="7" dataType="0005" accessType="ro" PDOmap
ping="TPDO"/>
<SubObject subIndex="02" name="DigitalInput" objectType="7" dataType="0005" accessType="ro" PDOmap
ping="TPDO"/>
</Object>

<Object index="6200" name="DigitalOutput_00h_AU8" objectType="8" dataType="0005">
<SubObject subIndex="00" name="NumberOfEntries" objectType="7" dataType="0005" accessType="const" de
faultValue="4" PDOmapping="no"/>
<SubObject subIndex="01" name="DigitalOutput" objectType="7" dataType="0005" accessType="rw" PDOmap
ping="RPDO"/>
<SubObject subIndex="02" name="DigitalOutput" objectType="7" dataType="0005" accessType="rw" PDOmap
ping="RPDO"/>
<SubObject subIndex="03" name="ControlByte" objectType="7" dataType="0005" accessType="rw" PDOmap
ping="RPDO"/>
<SubObject subIndex="04" name="ControlByte" objectType="7" dataType="0005" accessType="rw" PDOmap
ping="RPDO"/>
</Object>

<Object index="6401" name="AnalogueInput_00h_AI16" objectType="8" dataType="0003">
<SubObject subIndex="00" name="NumberOfEntries" objectType="7" dataType="0005" accessType="const" de
```

RPG2 I/O CONFIGURATION TOOL USER GUIDE

```

faultValue="2" PDOmapping="no"/>
<SubObject subIndex="01" name="AnalogueInput" objectType="7" dataType="0003" accessType="ro" PDOmap▶
ping="TPDO"/>
<SubObject subIndex="02" name="AnalogueInput" objectType="7" dataType="0003" accessType="ro" PDOmap▶
ping="TPDO"/>
</Object>

<Object index="6411" name="AnalogueOutput_00h_AI16" objectType="8" dataType="0003">
<SubObject subIndex="00" name="NumberOfEntries" objectType="7" dataType="0005" accessType="const" de▶
faultValue="2" PDOmapping="no"/>
<SubObject subIndex="01" name="AnalogueOutput" objectType="7" dataType="0003" accessType="rw" PDOmap▶
ping="RPDO"/>
<SubObject subIndex="02" name="AnalogueOutput" objectType="7" dataType="0003" accessType="rw" PDOmap▶
ping="RPDO"/>
</Object>

```

For each subindex defined for a given item, add another entry of duplicate size until the item size reaches the required value, while incrementing the subindex.

The subindices for digital items are always one byte as shown in this RPG2 I/O configuration tool user guide. The sample XDD file contains four entries for digital output to correspond with the 4 bytes and two entries for digital input to correspond with the 2 bytes.

The subindices for the analog items can be 1 byte, 2 bytes, or 4 bytes. This particular example uses a subindex of 2 bytes. Modifying the type of subindex changes the relationship of mapping the input and output footprint for the device. There are two 2 byte entries for both the analog input and analog output channels.

When adding input and output data based on the needs of the configuration database, follow this index number convention:

- ▶ Digital/input/8-bit: 0x6000
- ▶ Digital/output/8-bit: 0x6200
- ▶ Analog/input/8-bit: 0x6400
- ▶ Analog/input/16-bit: 0x6401
- ▶ Analog/input/32-bit: 0x6402
- ▶ Analog/output/8-bit: 0x6410
- ▶ Analog/output/16-bit: 0x6411
- ▶ Analog/output/32-bit: 0x6412

RPG2 PROGRAMMING USER GUIDE

FEATURES

- ▶ Programming over the API
- ▶ Programming by means of the embedded web server
- ▶ Programming by means of JTAG

EQUIPMENT NEEDED

- ▶ JTAG programmer: J-Link
- ▶ Ethernet cable
- ▶ UART cable
- ▶ API: UART, Ethernet, and SPI

DOCUMENTS NEEDED

- ▶ The following documents are available for download from the [main product page](#).
 - ▶ Embedded design materials
 - ▶ Schematics
 - ▶ Bill of materials
 - ▶ Example module layout
 - ▶ [RPG2 I/O Configuration Tool User Guide](#) section of this document.
 - ▶ [RPG2 Unified Interface User Guide](#) section of this document

SOFTWARE NEEDED

- ▶ [Link Configuration file](#)
- ▶ [Web Server file](#)
- ▶ [I/O Configuration file](#)
- ▶ [Flash Image firmware](#)
- ▶ Segger J-Flash software
- ▶ Python Software Foundation (PSF) Python 3

GENERAL DESCRIPTION

The RapID Platform Generation 2 (RPG2) module provides support for multiple industrial Ethernet protocols. When the RPG2 evaluation kit is purchased, there is a predetermined software set loaded in the evaluation kit. When a user has performed the evaluation of the RPG2 solution using the evaluation kit, the kit can be reprogrammed to either evaluate another industrial Ethernet protocol or to customize the RapID platform to fit the needs of the system of the user.

To reprogram the RPG2 solution, choose one of the three following methods: over the application processor interface (API), by means of the web server, or by means of JTAG.

The first method to reprogram the kit is by sending unified interface commands over the API. Note that a specific API must be used with specific function calls to accomplish this. The programming process is touched upon procedurally within this user guide. However, the exact API function is described in detail in the [RPG2 Unified Interface User Guide](#) section.

The second method to reprogram the RPG2 solution is by using the embedded web server, which is predistributed in all RPG2 evaluation kits and is included as part of the RPG2 software solution. For the embedded web server, access to the server varies on a protocol by protocol basis because the RPG2 device requires an IP address, and how the IP address is obtained also varies on a protocol by protocol basis.

The third method to reprogram the RPG2 solution is by means of JTAG by using J-Flash, which requires PSF Python 3, a J-Flash installation, and a J-Flash license for production purposes.

RPG2 PROGRAMMING USER GUIDE

COMMUNICATIONS CONTROLLER APPLICATIONS AND IMAGES

There are several pieces of code (elements) that must be loaded onto the flash of the [ADSP-CM409F](#) for the RPG2 evaluation kit or for the embedded reference design. This group of pieces of code is a three part list that functions as an industrial Ethernet device.

Some of the elements are stored in flash, while others are stored on the file system in the RPG2 solution. Elements that are stored in flash are referred to as flash elements, and elements that are stored in the file system are referred to as data elements. A breakdown of where the different code pieces are ultimately stored is shown in [Table 37](#).

Table 37. System Elements

Element	Element Type
Network Application	Flash
Unified Interface	Flash
Bootloader	Flash
Link Configuration File	Data
I/O Configuration File	Data
Web Server Content	Data

Elements that are programmed as flash element types all need to be programmed as one binary image. The data element types can be loaded individually.

Network Application

The network application controls all Ethernet functionality on a given device that uses the RPG2 solution.

Ethernet functionality includes industrial Ethernet protocol management, web server traffic, and other forms of Ethernet traffic. The protocol stacks are included as part of the precompiled binaries. Therefore, there is no need to obtain any protocol stacks when designing a new industrial Ethernet device.

[Table 38](#) gives the supported industrial Ethernet protocol images, which for evaluation purposes, either use an Ethernet application processor link type out of the box or adhere to the [link configuration](#) file that is in the file system.

Table 38. Flash Elements

Protocol	Image Name
PROFINET	rapid-profinet-software-suite.bin
EtherNet/IP	rapid-ethernetip-software-suite.bin
EtherCAT	rapid-ethercat-software-suite.bin

The network application is programmed as part of the flash element image.

Unified Interface

The unified interface is the input and output application that controls the total functionality of the industrial Ethernet device. The unified interface can be the same regardless of the industrial Ethernet protocol in use. Refer to the [RPG2 Unified Interface User Guide](#) section of this document for more additional details on the unified interface.

The unified interface is programmed as part of the flash element image.

Bootloader

The bootloader controls the start-up sequence and authentication of other images that are loaded into the RPG2 flash memory.

In the RPG2 evaluation kit, some application image mismatches can potentially produce errors and illuminate light emitting diodes (LEDs) on the evaluation kit. The system behavior at start-up described in the [RPG2 Hardware Design Integration Guide](#) section relies on these optional LEDs. The customer can choose whether or not to implement LED5 and LED6 and LED7 and LED8 in the embedded reference design or the evaluation kit in the end application.

The bootloader is programmed as part of the flash element image.

RPG2 PROGRAMMING USER GUIDE

Link Configuration File

The link configuration file is included as part of the application suite. Users program the link configuration to change the link type or link parameters. Refer to [RPG2 Hardware Design Integration Guide](#) section for additional details on changing the link type. The link configuration file is a data element.

I/O Configuration File

The I/O configuration file is what determines the I/O footprint and protocol specific device parameters. Once a user is done with the evaluation, the user then moves to creating custom configuration data to use on the RPG2 module. The I/O configuration file is a data element. Refer to the [RPG2 I/O Configuration Tool User Guide](#) section for additional information.

Web Server Content

The web server content must be loaded. The web server is also a data element. Custom web content can be created for the RPG2 solution, which is described in more detail in the [RPG2 Web Server User Guide](#) section of this document.

NEEDED IMAGES FOR THE RPG2 SOLUTION

To have an industrial Ethernet device, an RPG2 user must load several things onto the device. There are two high level pieces that must be part of the RPG2 software load. One is the flash image, and the other is the file system.

Flash Element

The flash image is the firmware stored in the flash of the RPG2 module. This image is industrial Ethernet protocol specific and one of the following:

- ▶ PROFINET
- ▶ EtherNet/IP
- ▶ EtherCAT

Each of these protocols contains all of the pieces of the software stored in flash and the flash elements shown in [Table 38](#).

Data Elements

Data elements are stored in the file system, and some customization can be done, such as the following:

- ▶ [Web server file](#)
- ▶ [IO configuration file](#)
- ▶ [Link configuration file](#)
- ▶ MAC address file

The web server file and the IO configuration file are required to do any industrial Ethernet communication with the RPG2 module. The following sections detail what the expected file system and flash image combinations are.

RPG2 Programming Out of the Box

When loading the RPG2 module as an evaluation kit, embedded reference design, or in any other fashion, some content must be loaded for the RPG2 module to function as an industrial Ethernet device. At the top level, the following must be programmed onto the RPG2 module:

- ▶ The file system, which contains the data elements
- ▶ A firmware image, which is the flash element

The file system can come as a precompiled binary or be created based on the needs of the system of the user. The firmware image contains the needed application for an industrial Ethernet binary.

RPG2 PROGRAMMING USER GUIDE

Loading PROFINET

The actual content when loading the files needed to create a PROFINET device with the RPG2 module follows:

- ▶ file-system
- ▶ private
- ▶ pki
- ▶ ca.crt
- ▶ server.crt
- ▶ server.key
- ▶ public
- ▶ web server
- ▶ io-config.bin
- ▶ link-config.txt
- ▶ rapid-profinet-software.bin

Note that if a user is programming the RPG2 embedded reference design for the first time, the file system portion must be loaded first. See the [Programming by Means of JTAG](#) section for more information.

Loading EtherNet/IP

The actual content when loading the files needed to create a EtherNet/IP device with the RPG2 module follows:

- ▶ file-system
- ▶ private
- ▶ pki
- ▶ ca.crt
- ▶ server.crt
- ▶ server.key
- ▶ public
- ▶ web server
- ▶ io-config.bin
- ▶ link-config.txt
- ▶ rapid-ethernetip-software.bin

Note that if a user is programming the RPG2 ERD for the first time, the file system portion must be loaded first. See the [Programming by Means of JTAG](#) section for more information.

Loading EtherCAT

The actual content when loading the files needed to create a EtherCAT device with the RPG2 module follows:

- ▶ file-system
- ▶ private
- ▶ pki
- ▶ ca.crt
- ▶ server.crt
- ▶ server.key
- ▶ public
- ▶ web server
- ▶ io-config.bin
- ▶ link-config.txt

RPG2 PROGRAMMING USER GUIDE

► rapid-ethercat-software.bin

Note that if a user is programming the RPG2 ERD for the first time, the file system portion must be loaded first. See the [Programming by Means of JTAG](#) section for more information.

PROGRAMMING BY MEANS OF THE API

When programming the RPG2 module or embedded design by sending commands over the API, there are several commands that are used to complete this process. This process varies if the user is going to update the flash elements or the data elements.

The basic process for reprogramming flash elements is conceptually shown in [Figure 155](#), and [Figure 156](#) shows the user how to update the data elements.

Follow this process to update the flash elements:

- Call **NI_FileSave()** with the appropriate image (see the [Flash Element](#) section for additional information) as the argument to transfer the needed file over the link and save the image in the flash of the RPG2 module.
- Call **NI_MarkSoftwareUpdate()** for that image to indicate to the RPG2 module that upon reset that the saved file is loaded and used.
- Call **NI_Reset()** to reset the device and use the newly updated files.

More information on the specific arguments for the API can be found in both the source code for the [ni-example-app](#) and in the [RPG2 Unified Interface User Guide](#) section.

The process for updating data elements is one step. Call **NI_FileSave()** with the appropriate image (see [Table 39](#) for additional information) as the argument to transfer the file over the link. Each data element has its own user guide where it is defined, which include the following:

- [IO Configuration File](#)—[RPG2 I/O Configuration Tool User Guide](#) section
- [Link Configuration File](#)—[RPG2 Hardware Design Integration Guide](#) section
- [Web Server File](#)—[RPG2 Web Server User Guide](#) section

For the data elements, the updated file does not require a reset for use. The file is available for use immediately.

These functions and additional file management functions are detailed in the [RPG2 Unified Interface User Guide](#) section.

Additionally, the specific image names detailed in [Table 39](#) are expected by the RPG2 file system.

Table 39. Data Elements, Local File Name, and Image Name

Data Element	Local File Name	Image Name on the File System
IO Configuration File	User defined	io-config.bin
Link Configuration File	User defined	link-config.txt
Web Server File	User defined	Customizable

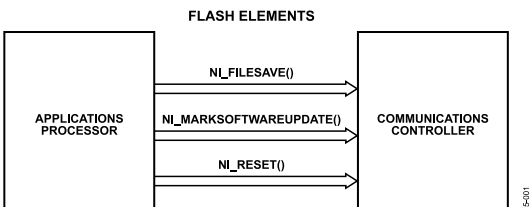


Figure 155. Updating Flash Elements

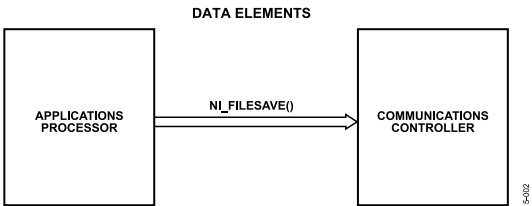


Figure 156. Updating Data Elements

RPG2 PROGRAMMING USER GUIDE

An Example for Reprogramming Over the API When Using the Ni-Example-App

This section describes an example for both programming data elements and flash elements.

The following arguments are used to reprogram either the flash or data elements.

- ▶ `ni-example-app.exe`—always used.
- ▶ `-l` (universal asynchronous receiver transmitter (UART) or Ethernet (ETH)—the out of the box configuration is Ethernet, however, UART can be used as well.
- ▶ If the link type is Ethernet, the next arguments are as follows:
 - ▶ `-n`
 - ▶ String for the Ethernet network interface card (NIC) identifier
- ▶ If the link type is UART, the next arguments are as follows.
 - ▶ `-c`
 - ▶ COMx, where x is the COM port on the PC in use, usually a virtual COM from a USB connection
- ▶ Location of the file on the RPG2 module that is loaded to.
- ▶ Local path for the file to be loaded onto the RPG2 module.

Two examples of the command line inputs follow:

- ▶ `ni-example-app.exe -l ETH -n {0CF10A5B-A995-4AC8-8E69-1B9225308092} --file-save io-config.bin "C:\Users\Desktop\io-config-local.bin"`.
- ▶ The actual string following `-n` and the `io-config-local.bin` file are variable depending on where the user has the file, how the file is named, and the actual device string for the Ethernet NIC.
- ▶ `ni-example-app.exe -l UART -c COM11 --file-save io-config.bin "C:\Users\Desktop\io-config-local.bin"`.
- ▶ The actual string following `-n` and the `io-config-local.bin` file are variable depending on where the user has the file, and x for the COMx is dependent on the COM port in use (this can be found by typing `mode` from a command line)

Data Elements

A sample console command is shown in [Figure 157](#) using the `ni-example-app`.

```
Administrator: Command Prompt
C:\Users\cstelman\Desktop>ni-example-app.exe -l UART -c COM11 --file-save io-config.bin "C:\Users\cstelman\Desktop\io-config.bin"
Welcome to ni-example-app!

Processing arguments...
DONE
Performing system startup...
DONE
Start time update process...
DONE
Starting up example application...
DONE
Link init...
DONE
Unified Interface stack init...
DONE
Set response timeout...
DONE
Find Network Interface...
DONE
Save C:\Users\cstelman\Desktop\io-config.bin to module as io-config.bin...
DONE
Reset Network Interface...
DONE
Wait 10000 us for module to finish reset...
DONE
Get installed protocol...
DONE (Ethernet/IP)
Set device 400...
DONE
Add item 500 to location 1...
DONE
Add item 501 to location 2...
DONE
Add item 502 to location 3...
DONE
Set transmit modes...
DONE
Finalize configuration...
DONE
C:\Users\cstelman\Desktop>
```

Figure 157. Programming Data Elements

`io-config.bin` is what the name of the [IO configuration file](#) must be saved as in the file system for the RPG2 module. This file must also be saved at the root directory peer to the private and public folders, which is also true for the [link configuration file](#). The link configuration file must be saved as `link-config.txt`. Formatting for the link configuration file is discussed later in this document. Formatting for the IO configuration

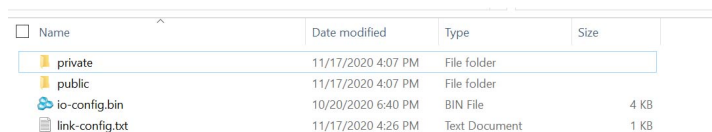
RPG2 PROGRAMMING USER GUIDE

file is taken care of by the **IO-Configuration-Utility.exe** and is discussed in the [RPG2 I/O Configuration Tool User Guide](#) section. [Figure 158](#) shows a screenshot of what the file system looks like at the root directory.

Flash Elements

Flash elements can be programmed by using the **ni-example-app** that is presupplied. In the console output shown in [Figure 159](#), an example of saving the application into the file system and marking it for use upon reset is shown. This flash element can then be marked and used upon reset. For this example, the binary saved onto flash is **rapid-ethernetip-software-suite.bin**.

Note that for a module user, there is a predistributed EtherNet/IP network application that allows the API to initialize and send the needed commands to reprogram both the data and flash elements. While it is predistributed with EtherNet/IP, it is recommended to reload the EtherNet/IP software from the portal when moving to conformance testing so that the software is up to date.



Name	Date modified	Type	Size
private	11/17/2020 4:07 PM	File folder	
public	11/17/2020 4:07 PM	File folder	
io-config.bin	10/20/2020 6:40 PM	BIN File	4 KB
link-config.txt	11/17/2020 4:26 PM	Text Document	1 KB

Figure 158. File System Root Directory

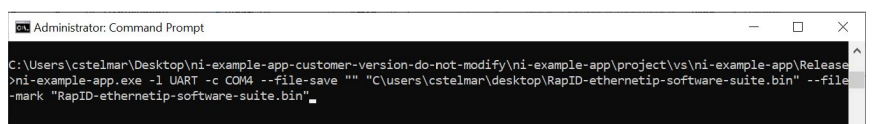


Figure 159. Updating Flash Elements

PROGRAMMING BY MEANS OF THE EMBEDDED WEB SERVER

The RPG2 solution gives a user the means of updating files by using the embedded web server that exists on the device. There are several steps to take to perform this operation.

Accessing the Device Web Page

The web server is accessed like other devices by the IP address in a browser. Before a user can access the web page, the device needs an IP address, and how it obtains an IP address varies depending on the protocol that is in use.

PROFINET

For PROFINET, it is required for conformance purposes that the device obtain an IP address from the leader on the PROFINET network. The controller assigns the RPG2 device a name and an IP address, and the IP address is then used by the RPG2 device.

EtherNet/IP

For an EtherNet/IP device, typically the device obtains an IP address by means of a dynamic host configuration protocol (DHCP) server, which is the recommended method for use by the Open DeviceNet Vendor Association (ODVA). The device is allowed to have a static IP address, and this option is also available for the user as well.

EtherCAT

For an EtherCAT device, the only way to access the device web server is by means of Ethernet over EtherCAT (EOE), which is due to the features of the industrial Ethernet EtherCAT protocol. Note that, this implementation of Ethernet over EtherCAT varies depending on the EtherCAT leader that is in use.

Other Protocols

All protocols have their own way of obtaining an IP address. The rules of the protocol must be adhered to in order to obtain an IP address before accessing the embedded web server.

RPG2 PROGRAMMING USER GUIDE

The Device Web Page and the Reprogramming Process

This section explains the procedure by which the firmware on the device is updated, assuming that the final binary is built and available. The final binary includes the IO application, the network application, and the bootloader in a single binary file. The firmware update can be carried out using the web server, and the LEDs (LED1 and LED2) indicate the firmware update status.

The [I/O configuration file](#) and [link configuration file](#) can be reprogrammed using the same process. However, unlike the other parts of the application space, this data and file can be updated separately using a .bin output from the [RPG2 I/O Configuration Tool User Guide](#) section of this document or the required link configuration file. Consult the [RPG2 I/O Configuration Tool User Guide](#) section for more information on the I/O configuration file.

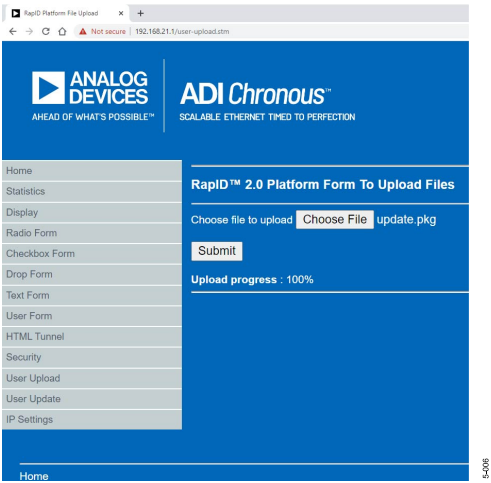


Figure 160. Web Page View

Once the RPG2 web page is accessed from the [main product page](#), upload files as follows:

- ▶ Open the web page of the RPG2 module with a web browser. (The IP address must be prefixed with an https. For example, enter https://192.168.1.1 to access a device at 192.168.1.1).
- ▶ Click **User Upload** in the left-hand column.
- ▶ Authenticate the web server with the username and password provided or set. Be sure to use the user account.
- ▶ Upload the applicable file, which is either in the [RPG2 I/O Configuration Tool](#) output, the [link configuration file](#), web server content, or one of the binaries referenced (PROFINET, EtherNet/IP, or EtherCAT).
- ▶ After the upload reaches 100%, click **User Update**. Then, select the **update.pkg** file that just uploaded in the list, and click **Submit** to kick off the firmware update.
- ▶ The device of the user automatically reboots and performs the secure firmware update. To verify, refer to the LEDs to confirm that the update is occurring (see [Table 40](#)).
- ▶ The bootloader indicates which step of the authenticating and booting applications process the device is in with the NET and MOD LED (see [Table 40](#)).
- ▶ After the bootloader finishes booting, it turns all LEDs off.

Table 40. LED Behavior Descriptions for Bootloader

Bootloader Step Progress Status	LED1	LED2
Update Found, Authentication in Progress	Green	Off
Application Verification and Download in Progress	Off	Green
Boot Process Complete	Green	Green
Firmware Update Finished	Off	Off

RPG2 PROGRAMMING USER GUIDE

PROGRAMMING BY MEANS OF JTAG

The RPG2 solution can be programmed by means of JTAG using J-Flash, or a user can use the Python 3 script that is distributed on the [main product page](#) under the normal software download files. Note that it is up to the user to obtain a license for production programming when this is done.

- ▶ RapID PROFINET software suite
- ▶ RapID EtherCAT software suite
- ▶ RapID EtherNet/IP software suite

Inside each zip file is the .bin file that must be entered to program by means of JTAG.

Using J-Flash

When using J-Flash, users must select the following:

- ▶ **Device:** [ADSP-CM409BSWZ-AF](#)
- ▶ **Interface:** JTAG
- ▶ **Speed:** 4 kHz
- ▶ **Data File,** which is the .bin file obtained in the zip file.
- ▶ **Program Address:** 0x18000000

Selecting anything else causes the RPG2 solution to not function. Loading these binaries as the flash element allows other means of programming to load data elements onto the RPG2 module.

Erasing the Flash Image

When reprogramming the RPG2 module from one flash image to another using JTAG, the flash must be erased before reprogramming, and LED2 indicates when this can be done. When the device powers up, the user must wait until LED2 illuminates to click **Erase Chip** on the main screen in J-Flash. Once erasing completes, the user can then reprogram using the Python 3 scripts.

Using a Python Script

When using a Python 3 script, several things must be specified. The Python 3 script has four arguments that are specified in more detail within in the following sections.

- ▶ The [link configuration file](#) is found in the **RapID_Example_Content** zip file.
- ▶ The file system is created using the existing pieces on the ADIN2299 main product page at www.analog.com/adin2299 from the different zip files as specified in the needed images for the RPG2 solution.
- ▶ The network application binary, which is the protocol specific software.
- ▶ The format option formats the file system before loading the new one onto the RPG2 solution.

The Board Configuration

The board configuration file is found in the **util** subdirectory under the **RapID_Programming_Utility_Suite**. For the purposes of this example in using the script, the name is **ADIN2299.bin**. This name can vary but must always be obtained under the **util** subdirectory.

The File System

The file system is flexible and can contain many different pieces, such as the following:

- ▶ Web server content, which is the default or the custom web content found in the **RapID_Example_Content_Library**. The web server content is not included by default and must be specified in the file system when loading. Consult the [RPG2 Web Server User Guide](#) section for more information.
- ▶ [IO configuration file](#), which is always saved as **io-config.bin** and found in the **RapID_Example_Content_Library** or generated using the **IO-Configuration-Utility.exe**.

RPG2 PROGRAMMING USER GUIDE

- ▶ Link configuration file, which is always saved as **link-config.txt** and is selected from content in the **RapID_Example_Content_Library** and renamed as described in the [RPG2 Hardware Design Integration Guide](#) section.
- ▶ The MAC configuration file, which is always saved as **rems-interface.txt** and found in the **RapID_Example_Content_Library**. This file is formatted as shown in [Figure 161](#).

```
PRIMARY MAC ADDRESS: 78:C6:BB:00:00:00
PORT 0 MAC ADDRESS: 78:C6:BB:00:00:01
PORT 0 AUTO-NEG ENABLE: 1
PORT 0 SPEED: 100
PORT 0 DUPLEX MODE: "FULL"
PORT 1 MAC ADDRESS: 78:C6:BB:00:00:02
PORT 1 AUTO-NEG ENABLE: 1
PORT 1 SPEED: 100
PORT 1 DUPLEX MODE: "FULL"
```

Figure 161. MAC Configuration File

For the purposes of using the script in this example, the file system proposed only has a MAC address file, IO configuration data, a link configuration file, and no web server content.

The Network Application

The network application is a .bin file that is the flash element and predistributed as detailed in [Table 37](#). For the purposes of this example, the file used is **rapid-ethernetip-software-suite.bin**.

All available software suites are on the main product page on a protocol by protocol basis. Check the ADIN2299 main product page at www.analog.com/adin2299 for the industrial Ethernet protocol of interest.

Programming Script Usage

The following is an example of how to use the Python 3 script. When using a command line with the Python 3 script, the command line prompt must run as elevated. Load the following files:

- ▶ rapid-ethernetip-software-suite.bin
- ▶ File-System
- ▶ io-config.bin, which can be either the default configuration file or the configuration file the customer created using the [RPG2 I/O Configuration Tool User Guide](#) section of this document.
- ▶ link-config.txt, which can be either the default [link configuration file](#) or the custom link configuration file of the customer.
- ▶ Private/settings/rems-interface.txt, which defines the MAC address of the RPG2 design.

Note that loading files for the ADIN2299 from the load script can be accomplished by clicking on **Load.bat** from the main zip file.

It is not a requirement that the user supply all three arguments, and programming with the script can be done with any of the files individually. Upon success, something similar to the messages shown in [Figure 163](#) and [Figure 164](#) are seen.

The total process once the script is entered takes approximately 30 seconds to one minute to invoke J-Flash from Python 3. Note that there are two points where the command line instructs the user to press any key. Follow the instructions on the command line to go through the script.

After programming the file system image, LED2 flashes green continuously. After programming the flash image, LED2 blinks green once and then the optional LEDs (LED3 and LED4) are solid green. If the RPG2 design does not have LED3 and LED4, the SDONE signal goes high once the device is ready for programming. SDONE is connected to LED3 in the RPG2 sample schematic on the [main product page](#).

If something fails, a log file displays peer to the script, and an error message appears. In addition, if there is a problem, it displays on the command line as well.

There is also a command to format the file system called fs-format that formats the file system before writing files. It is recommended to use this command especially when reprogramming the RPG2 module. The help screen for the Python 3 script shows a use case in [Figure 165](#).

For most use cases, the **—board-config** option can be omitted because it is only required for certain hardware.

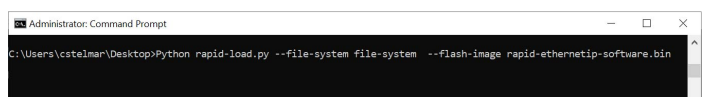
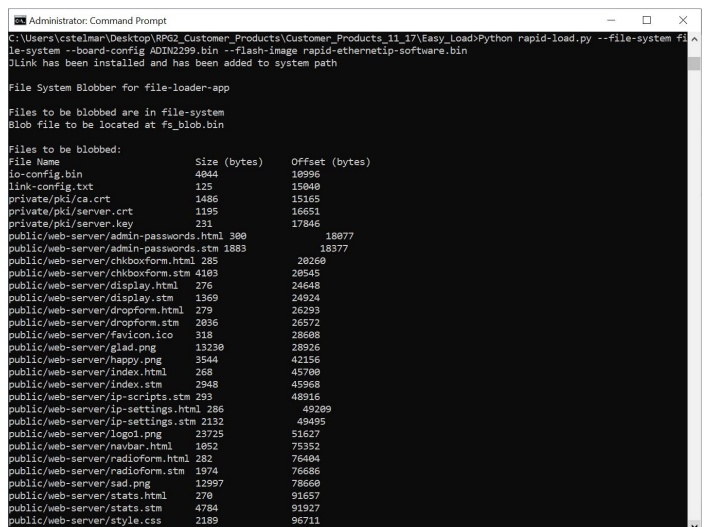


Figure 162. Sample Python 3 Script Command

RPG2 PROGRAMMING USER GUIDE



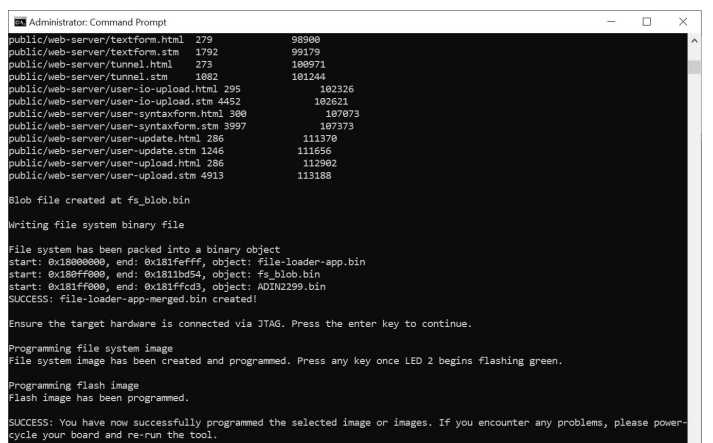
```
Administrator: Command Prompt
C:\Users\cstelmar\Desktop\RPG2_Customer_Products\Customer_Products_11_17\Easy_Load\Python rapid-load.py --file-system file-system --board-config ADIN2299.bin --flash-image rapid-ethernetip-software.bin
JLink has been installed and has been added to system path

File System Blobber for file-loader-app

Files to be blobbed are in file-system
Blob file to be located at fs_blob.bin

Files to be blobbed:
File Name                               Size (bytes)  Offset (bytes)
file-config.bin                         4044          10996
link-config.txt                         125           15040
private/pki/ca.crt                     1486          15165
private/pki/server.crt                 1195          16651
private/pki/server-key                  231           17846
public/web-server/admin-passwords.html 300           18077
public/web-server/admin-passwords.stm 1883          18377
public/web-server/chkboxform.html      285           20260
public/web-server/chkboxform.stm       4183          20845
public/web-server/display.html          276           24648
public/web-server/display.stm          1369          24924
public/web-server/dropform.html         279           26293
public/web-server/dropform.stm         2036          26572
public/web-server/favicon.ico           318           28608
public/web-server/glad.png              13230         28926
public/web-server/happy.png             3544          42156
public/web-server/index.html            268           45780
public/web-server/index.stm             2948          45968
public/web-server/ip-scripts.stm        293           48916
public/web-server/ip-settings.html       286           49209
public/web-server/ip-settings.stm       2132          49495
public/web-server/legel.png             23725         51627
public/web-server/navbar.html           1052          75352
public/web-server/radioform.html        282           76404
public/web-server/radioform.stm         1974          76686
public/web-server/sad.png                12997         78660
public/web-server/stats.html            278           91657
public/web-server/stats.stm             4784          91927
public/web-server/style.css             2189          96711
```

Figure 163. Python Script Success Messages Part 1



```
Administrator: Command Prompt
public/web-server/textform.html         279           98980
public/web-server/textform.stm          1792          99179
public/web-server/tunnel.html           273           100971
public/web-server/tunnel.stm            1082          101244
public/web-server/user-io-upload.html    295           102526
public/web-server/user-io-upload.stm    4452          102621
public/web-server/user-syntaxform.html   300           107073
public/web-server/user-syntaxform.stm   3997          107373
public/web-server/user-update.html       286           111370
public/web-server/user-update.stm        1246          111656
public/web-server/user-upload.html       286           112902
public/web-server/user-upload.stm        4913          113188

Blob file created at fs_blob.bin

Writing file system binary file

File system has been packed into a binary object
start: 0x1800000, end: 0x181aff, object: file-loader-app.bin
start: 0x180ff000, end: 0x1811bd54, object: fs_blob.bin
start: 0x181ff000, end: 0x181ffcd3, object: ADIN2299.bin
SUCCESS: file-loader-app-merged.bin created!

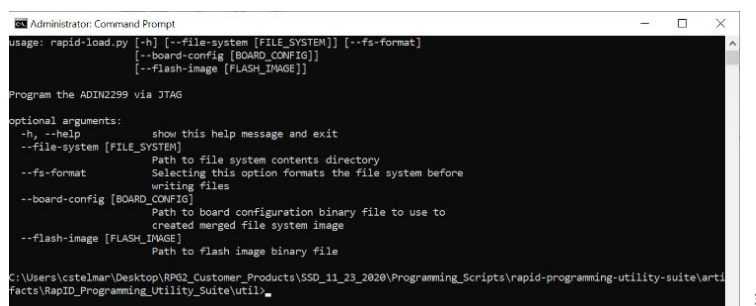
Ensure the target hardware is connected via JTAG. Press the enter key to continue.

Programming file system image
File system image has been created and programmed. Press any key once LED 2 begins flashing green.

Programming flash image
Flash image has been programmed.

SUCCESS: You have now successfully programmed the selected image or images. If you encounter any problems, please power-cycle your board and re-run the tool.
```

Figure 164. Python Script Success Messages Part 2



```
Administrator: Command Prompt
usage: rapid-load.py [-h] [--file-system [FILE_SYSTEM]] [--fs-format]
                    [--board-config [BOARD_CONFIG]]
                    [--flash-image [FLASH_IMAGE]]

Program the ADIN2299 via JTAG

optional arguments:
  -h, --help            show this help message and exit
  --file-system [FILE_SYSTEM]
                        Path to file system contents directory
  --fs-format            Selecting this option formats the file system before
                        writing files
  --board-config [BOARD_CONFIG]
                        Path to board configuration binary file to use to
                        create merged file system image
  --flash-image [FLASH_IMAGE]
                        Path to flash image binary file

C:\Users\cstelmar\Desktop\RPG2_Customer_Products\SSD_11_23_2020\Programming_Scripts\rapid-programming-utility-suite\artifacts\
RapidID_Programming_Utility_Suite\util>
```

Figure 165. RPG2 Python 3 Help Screen

PROGRAMMING METHODS BY USE CASE

When programming the RPG2 solution, there are different use cases that must also be considered. The three use cases include the evaluation kit, the module, and the embedded reference design.

For an RPG2 embedded reference design, the only way to program initially is by means of JTAG. It is the responsibility of the user to obtain a J-Link license for production purposes. After the embedded design has been programmed with a flash image and file system by means of

RPG2 PROGRAMMING USER GUIDE

JTAG, the user can use the API to configure the design, if desired. For maintenance purposes, the web server can be used for this use case as well.

Table 41. Different Systems

Use Case	Method	Supported Out of the Box	Notes
Evaluation Kit	JTAG	Yes	Not applicable
	API	Yes	Not applicable
	Web server	Yes	Not applicable
Module	JTAG	Yes	Need JTAG connector on design
	API	No	Not applicable
	Web server	Yes	Default method
Embedded Reference Design	JTAG	Yes	Only initial method
	API	No	Not applicable
	Web server	No	Not applicable

PROGRAMMING METHODS BY STATE OF DESIGN

Evaluation

During the evaluation phase, there are several methods to program and reprogram the RGP2 evaluation kit. The most straightforward way is with the **ni-example-app.exe** file. There is a reprogram option that can be entered from the command line that allows a user to change the I/O configuration data or upload a merged image containing what is required. Refer to [Programming by Means of the API](#) section for additional information.

The embedded web server can also be used as long as the device is assigned an IP address, which is generally not something that is expected of the customer during the evaluation phase. Once the device has an IP address, follow the [Programming by Means of the API](#) section for additional information.

JTAG can be used, and there are predistributed Python 3 scripts that load and reload the required flash element image along with the evaluation data element images. See the [Programming by Means of JTAG](#) section for further information.

Table 42. Method of Programming by Stage of Design

Stage in Design	JTAG	API	Web Server
Evaluation	Yes	Yes	Unlikely
Development	Yes (for flash elements)	Yes	Unlikely
Production	Yes (for flash elements)	Yes	Unlikely
Maintenance	Very unlikely	Yes	Yes

Software Development

Changing the API Link Type

All of the evaluation kits are shipped with no link configuration file. Therefore meaning the LT pins will be used to determine link type. If this is not desirable and the user wants to make the link type(SPI, UART or Ethernet) ignore the selector pins, then this need is determined by the [link configuration file](#).

When changing the link configuration file, users must consider that the file is no longer accessible via the original parameters defined in the link configuration file. In other words, if a user changes the API from Ethernet to UART, the API no longer communicates via Ethernet, and the only way to interact with the RPG2 module is to use the newly programmed link type. Note that the link type change takes immediate effect. [Figure 166](#) shows an example command of how to change the link type from Ethernet to UART.

RPG2 PROGRAMMING USER GUIDE

Production Programming Considerations

During production, some methods of programming are more useful than others.

For initial loading of the flash memory, a user may require the use of JTAG. The user must load the required flash element image onto the embedded reference design, which then allows interaction with the RPG2 solution and determines the proper bring up from there.

Reprogramming the link configuration file and the I/O configuration file for the embedded design with JTAG involves putting these files in the file system.

Table 43. Configuring the Link Type for the RPG2 Solution

API Type	New Link Configuration Required?	Configurable Board Configuration Parameters
Ethernet	No	Off
UART	Yes	Baud rate, parity, stop bit
SPI	Yes	Speed, clock phase, clock polarity

```
C:\Users\cste1mar\Desktop>ni-example-app.exe -l ETH -n {ACF4A8A7-264E-4C73-81AF-4C2616AE5830} --file-save "link-config.txt" "C:\Users\cste1mar\Desktop\link-config_UART.txt"
```

Figure 166. Command for Changing Link Type

Maintenance Programming Considerations

For maintenance purposes, it is assumed for many cases that the user does not have JTAG access for the end device of the user. Therefore, when the user gets to a situation where maintenance programming must happen, the user likely uses the web server for updating out in the field.

Some users may store update files on the API for the purpose of updating devices in the field, and it is up to the user how this is done. See the [RPG2 Unified Interface User Guide](#) section for more additional information.

RPG2 HARDWARE DESIGN INTEGRATION GUIDE

FEATURES

- ▶ Customizable network input and output configuration
- ▶ Customizable link types
 - ▶ Ethernet
 - ▶ SPI
 - ▶ UART

EVALUATION KIT CONTENTS

- ▶ 1 baseboard with the RPG2 module installed with the protocol specific software preloaded
- ▶ 1 wall mount AC adapter, 90 V ac to 264 V ac to 12 V dc, 12 W, 1 A
- ▶ 4 power supply plug adapters (Type A, Type C, Type G, and Type I)
- ▶ 1 USB A male to USB Micro B male cable
- ▶ 1 Ethernet cable

EQUIPMENT NEEDED

- ▶ JTAG programmer: J-Link LITE or J-Link BASE
- ▶ 1 Ethernet cable

DOCUMENTS NEEDED

- ▶ The following documents are available for download from the [main product page](#).
 - ▶ Embedded design materials
 - ▶ Schematics
 - ▶ Bill of materials
 - ▶ Example module layout

SOFTWARE NEEDED

- ▶ The following files are available for download on the ADIN2299 [main product page](#).
 - ▶ [Link Configuration](#) file
 - ▶ [I/O Configuration](#) file
 - ▶ [Network Configuration](#) file (EDS, GSDML, or ESI, which can be found in the IO_Configuration_Utility.zip file within the [IO Configuration](#) file)

GENERAL DESCRIPTION

The RapID Platform Generation 2 (RPG2) hardware design integration guide provides the engineering details to integrate the hardware components needed to interface with the [RPG2 Embedded Reference Design](#) zip file. The software within is used to configure the hardware. As shown in [Figure 167](#), this user guide requires the [EV-RPG2](#) evaluation kit set up to run the example application.

DEVELOPMENT OVERVIEW

Note that all of these documents are self contained in this reference manual.

RPG2 HARDWARE DESIGN INTEGRATION GUIDE

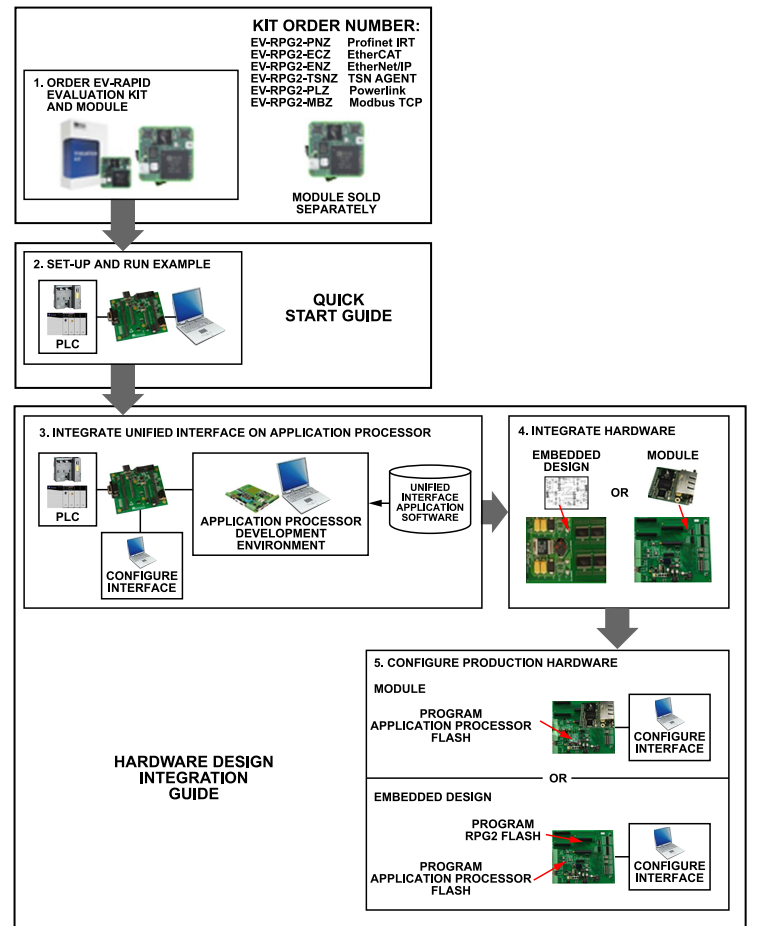


Figure 167. Development Overview with the Associated Documentation

INTRODUCTION

The [RPG2 EtherCAT Quickstart Guide](#) section, [RPG2 EtherNet IP Quickstart Guide](#) section, and the [RPG2 Profinet Quickstart Guide](#) section of this document provide instructions to connect the items in the [EV-RAPID](#) evaluation kit and run an application that demonstrates end-to-end communication from an application processor to a programmable logic controller (PLC). For the example described in these previous quickstart guides, the application processor is a PC that communicates with the module via an Ethernet network interface card (NIC). Users can switch between the three link types, universal asynchronous receiver transmitter (UART), Ethernet, or serial peripheral interface (SPI), by following the steps outlined in the [Obtaining and Modifying the Link Configuration File](#) section. A PLC or a controller simulator running on a PC can be used.

When the application is running, it is possible to switch out the PC for the actual application processor and verify end to end communication from the application processor to a PLC or controller simulator. Update the board configuration file and program the board to use the desired link type. The next step in the configuration process is to modify the example code to work with the final application software.

See the [Software Integration](#) section for a description on how to integrate the example code into the application processor, as well as how to use the [link configuration](#) file that is located in the [RapID Software Downloads](#). (Note that the usage of this file is detailed in the [Obtaining and Modifying the Link Configuration File](#) section). In addition, see the [RPG2 I/O Configuration Tool User Guide](#) section of this document for additional information on how to generate the input and output configuration data, and for details on how data is organized in an ESI, XDD, GSDML, EDS, or ModbusTCP.

Take the following steps to integrate the field device application software:

1. Modify the application processor software.
2. Create an application processor software .bin to program the embedded reference design. Refer to the [Considerations for Production and Maintenance](#) section of the Programming section.
3. Execute the examples shown in the quickstart guides with your own embedded reference design.

RPG2 HARDWARE DESIGN INTEGRATION GUIDE

See the [Hardware Integration](#) section for a description on how to embed the design into the circuit card of the final application hardware.

Take the following steps to integrate the field device application hardware:

1. Create an electronic network configuration file. Refer to the [Network Configuration File and the I/O Configuration](#) section and the [RPG2 I/O Configuration Tool User Guide](#) section of this document for additional information.
2. Create and program the I/O configuration file. Refer to the [Network Configuration File and the I/O Configuration](#) section and the [RPG2 I/O Configuration Tool User Guide](#) section of this document for additional information.
3. Configure the application processor link type using the board configuration file. Refer to the [Obtaining and Modifying the Link Configuration File](#) section.

See the [Communication Interfaces Between the Application Processor and RPG2](#) section for information on how to load the RPG2 module with the output of the I/O configuration utility (**IO_Configuration_Utility.exe**). If the RPG2 module requires an update to a new version of the EtherNet/IP, PROFINET or EtherCAT software, the .bin files are contained in the software downloads:

- ▶ [RapID PROFINET Software Suite](#)
- ▶ [RapID EtherNetIP Software Suite](#)
- ▶ [RapID EtherCAT Software Suite](#)

These links download the following zip files that contain the proper .bin file for RPG2:

- ▶ RapID_PROFINET_Software_Suite.zip, which is found in the **bin/rapid_profinet_software_XXXXX.bin** location, where XXXXX is the current version of the software.
- ▶ RapID_EtherNetIP_Software_Suite.zip, which is found in the **bin/rapid_ethernetip_software_XXXXX.bin** location, where XXXXX is the current version of the software.
- ▶ RapID_EtherCAT_Software_Suite.zip, which is found in the **bin/rapid_ethercat_software_XXXXX.bin** location, where XXXXX is the current version of the software.

See the [RPG2 Programming User Guide](#) section for how to reprogram the RPG2 module with the binaries previously listed. In addition to instructions on how to reprogram RPG2 with the binaries previously listed, this guide describes how to load the file system with the output of the I/O configuration utility (**IO_Configuration_Utility.exe**, which can be found in the zip file downloaded under the [RPG2 I/O Configuration Tool User Guide](#) section of this document and the [main product page](#)), the link configuration file (see the [RPG2 I/O Configuration Tool User Guide](#) section for additional information), and the application processor interface (API) software.

All documentation and required files for the application described in this user guide are located on the [main product page](#) for the ADIN2299.

Tools and Project Phases

Developing a field device with the RPG2 module requires different tools throughout the phases of the life cycle of a device. [Figure 168](#) shows the tools that are required for each particular phase of the life cycle of the device. Not all tools, such as those required to design the circuit card assemblies, are shown because tools are application specific.

In the evaluation phase, the [EV-RAPID](#) or the RPG2 module is required. To run the end to end communication example, a PLC or controller simulator is required, as well as a PC to run the Network Interface Example Application Processor (**ni-example-app.exe**). The Network Interface Example Application Processor simulator is located in the main product page under the [Network Interface Example Application Suite](#) link. Click **Network_Interface_Example_Application_Suite** to download the Network_Interface_Example_Application_Suite.zip file, which contains the **ni-example-app.exe** executable file under **ni-example-app\projectvsni-example-app\Release**.

In the development phase, the application processor development environment replaces the Network Interface Example Application Suite (**ni-example-app.exe**). If the user intends to modify the I/O configuration data as well as the link type, refer to the [I/O Configuration Utility Suite](#) and the [link configuration file](#) (see the [RPG2 I/O Configuration Tool User Guide](#) section and the [Obtaining and Modifying the Link Configuration File](#) section for additional information).

RPG2 HARDWARE DESIGN INTEGRATION GUIDE

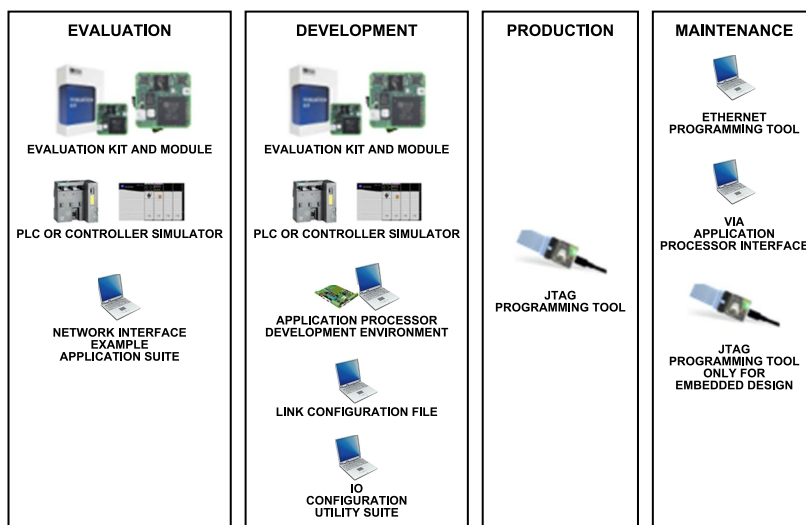


Figure 168. Tools Associated with the Development Phases of a Field Device

HARDWARE INTEGRATION

Embedded Design

The RPG2 solution can be embedded in the application hardware of a field device. RPG2 reference schematics are included to assist in the design process and are provided on the [main product page](#). Refer to the [RPG2 Embedded Reference Design](#) schematic along with the detailed design information in this reference manual when embedding the RPG2 solution into a design. Users are required to follow the component selection listed in the bill of materials of the [RPG2 Embedded Reference Design](#) zip folder.

When the RPG2 Embedded Reference Design schematic is implemented directly on the customer board, the only option for programming out of the box is through a Joint Test Action Group (JTAG) connection using the SEGGER Embedded Studio toolchain. Note that a serial wire debug connection is not supported. See the [RPG2 Programming User Guide](#) section of for more information.

The recommended JTAG wigglers include the following:

- ▶ J-Link LITE
- ▶ J-Link BASE

GPIO Requirements of the RPG2 Embedded Reference Design

The GPIO_x pins must have a maximum sourcing and sinking current of 4 mA.

Power Requirements of the RPG2 Embedded Reference Design

The embedded design consumes approximately 1.8 W, supplied from a single 3.3 V supply. The minimum power supply requirements for voltage and current are 3.3 V \pm 10% and 800 mA. It is best practice to use a 3.3 V and 1 A power supply for this application. The 3.3 V power is supplied to the +3V3 signal, and ground is connected to the GND signal.

Reset Requirements of the RPG2 Embedded Reference Design

The embedded design requires a power monitoring reset, supervisor chip with a 2.9 V threshold. This design must be connected to the SYS_HWRST (K02 pin of [ADSP-CM409CBCZ-AF](#), which is the processor for the RPG2 Embedded Reference Design). SYS_HWRST must be driven and cannot be left floating.

Regardless of how the embedded design is reset, it is not available for communication until the software has initialized the embedded design and entered the operating state (1.0 sec after the receipt of a valid reset pulse or after power is valid if no external reset source is used).

RPG2 HARDWARE DESIGN INTEGRATION GUIDE

Thermal Management Guidelines of the RPG2 Embedded Reference Design

The printed circuit board (PCB) plane layer structure is critical for thermal management. Generally, each power supply must have a dedicated plane layer. The embedded design utilizes a single 3.3 V power supply. The 3.3 V supply must have a dedicated plane layer, and the ground must have at least one dedicated plane layer. It is recommended to use an even number of plane layers to maintain PCB flatness. Note that the 2.5 V power supply is embedded in the embedded design.

Minimize the number of vias used. Each via puts a hole in the copper planes that adversely affects electrical and thermal performance. Perforating the plane layer with too many poorly placed vias makes it impossible to completely isolate the central power connections of a CSP_BGA.

Balance via minimized via usage with enough vias on the power supply connections for good electrical and thermal performance. Power supply vias are a significant factor in removing heat from the IC.

Avoid routing signals on the plane layers to prevent cutting up the plane and inhibiting the flow of heat and current.

The following design considerations reduce the power consumption of the design. Low dropout (LDO) type voltage regulators are inherently inefficient and must be avoided. It is recommended to use switching power supplies that can be implemented in a relatively small footprint.

Communication Interfaces Between the Application Processor and RPG2

UART Application Processor Interface

If the UART interface is selected, connect the application processor interface to the [RPG2 Embedded Reference Design](#) as shown in [Table 44](#) using the pins indicated within. The UART is configured for 115,200 bps with 8 data bits, no parity, and one start and one stop bit.

Table 44. UART Application Processor Interface Pins

Mnemonic	Direction	Description
UART0_TX	Output	UART0 transmit output
UART0_RX	Input	UART0 receive input

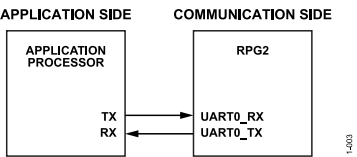


Figure 169. UART Application Processor Interface

SPI Application Processor Interface

If the SPI interface is selected, connect the application processor interface to the RPG2 module as shown in [Figure 170](#) using the pins indicated in [Table 45](#). The SPI follower connection supports a 10 MHz maximum clock rate with the clock phase (CPHA) = 0 so that data is captured on leading edge and the clock polarity (CPOL) = 0 with the leading clock edge rising. The Unified Interface protocol is described in the [RPG2 Unified Interface User Guide](#) section of this document.

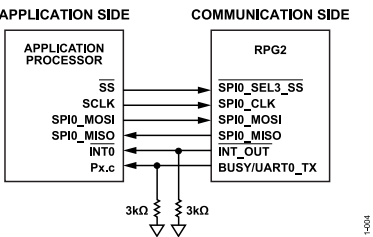


Figure 170. SPI Application Processor Interface

RPG2 HARDWARE DESIGN INTEGRATION GUIDE

Table 45. SPI Application Processor Interface Pins

Mnemonic	Direction	Description
SPI0_SEL3_SS	Input	SPI0 follower select input
SPI0_CLK	Input	SPI0 clock input
SPI0_MOSI	Input	SPI0 leader out, follower in
SPI0_MISO	Output	SPI0 leader in, follower out
INT_OUT	Output	Signal to notify the application side that a message is ready to be read
BUSY/UART0_TX	Output	Busy signal, indicating to the application side that the communication side is busy

Ethernet Application Processor Interface

The Ethernet link type can be either direct or indirect. A direct Ethernet link consists of two directly connected reduced media independent interfaces (RMII), one on the application side and one on the communication side. An indirect Ethernet link consists of two RMII connected through a physical layer (PHY) on either end or an Ethernet cable in between. The Unified Interface application software is agnostic to direct vs. indirect connections.

If the Ethernet interface is selected, connect the application processor interface to the RPG2 module as shown in Figure 171 or Figure 172 depending on whether you opt for a direct or an indirect connection using the pins indicated in Table 46.

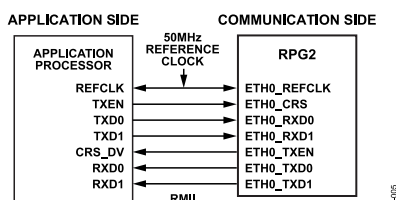


Figure 171. Direct Connection Through RMII

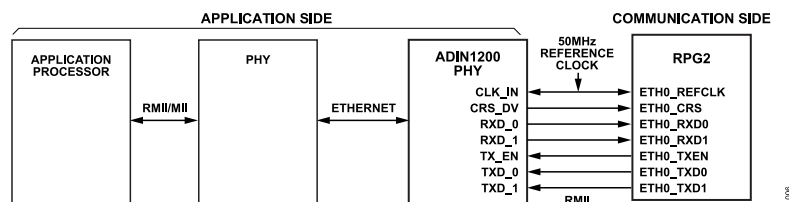


Figure 172. Indirect Connection

Table 46. Ethernet Application Processor Interface Pins

Mnemonic	Direction	Description
ETH0_REFCLK	Input	EMAC0 reference clock. Externally supplied Ethernet clock
ETH0_CRS/CRS/RXDV	Input	EMAC0 carrier sense (ETH0_CRS). Multiplexed on alternate clock cycles.
	Input	CRS is asserted by the PHY when either the transmit or receive medium is not idle, and CRS is deasserted when both are idle (CRS).
	Input	RXDV is asserted by the PHY when the data on ETH0_RXD_0 is valid (RXDV).
ETH0_RXD0	Input	EMAC0 Receive Data 0. Receive data bus.
ETH0_RXD1	Input	EMAC0 Receive Data 1. Receive data bus.
ETH0_TXEN	Input/output	EMAC0 transmit enable. When asserted, this pin indicates that the data on TXD_0 is valid.
ETH0_TXD0	Output	EMAC0 Transmit Data 0. Transmit data bus.
ETH0_TXD1	Output	EMAC0 Transmit Data 1. Transmit data bus.

RPG2 HARDWARE DESIGN INTEGRATION GUIDE

Required Support Circuitry

Table 47 lists the required external components for the [RPG2 Embedded Reference Design](#).

Table 47. External Components

Component	Description	Manufacturer	Part Number
Capacitors	Decouple each voltage rail (3.3 V) with a 47 μ F capacitor and a 0.1 μ F capacitor.	User specified	User specified
LEDs	MOD, NET, dual color LEDs, green and red	Kingbright	KPBA-3010ESGC
Resistors	470 Ω resistors, LED, current limit	User defined	User Defined
Ferrite Bead	1.5 A, 120 Ω ferrite bead at 100 MHz	Murata	BLM15EG121SN1D
MagJack	RJ45 jack, module connector with integrated magnetics and LEDs	Würth Electronics	7499010121A
Reset Supervisor	2.9 V power monitoring reset supervisor chip	Analog Devices	AMD708SARZ

SOFTWARE INTEGRATION

Software integration of the application processor is the same, regardless of the protocol. The information in this section applies to all protocol versions of the RPG2 including the following:

- ▶ PROFINET
- ▶ EtherNet/IP
- ▶ EtherCAT

To modify the example code, see the [RPG2 Unified Interface User Guide](#) section of this document.

Network Configuration File and the I/O Configuration File Installation and Modification

To install and modify the network configuration file (in EDS, ESI, GSDML, XDD, or Modbus/TCP format) and the I/O configuration file, refer to the [RPG2 I/O Configuration Tool User Guide](#) section of this document. Once you have modified both files, refer to the [RPG2 Programming User Guide](#) section of this document to program the board with the input and output configuration data.

Obtaining and Modifying the Link Configuration File

Go to the ADIN2299 [main product page](#) and download a software package to find the link configuration file.

Note that the link configuration file must be constructed as follows:

LINK TYPE: ETH

SPI CLOCK POLARITY: 0

SPI CLOCK PHASE: 0

UART BAUD RATE: 115200

UART PARITY: None

The link configuration file is used as a .txt file and then programmed into the ADIN2299.

Table 48 details all the possible link configuration file parameters.

Table 48. Application Processor Interface Link Type Parameters

Parameter	Usable Variables	Default	Interpreted As	Notes
LINK TYPE	Ethernet (ETH), SPI, UART, PINS	ETH	STRING	If the string entered does not match exactly what is in the Usable Variables window, RPG2 defaults to ETH and, therefore, is an Ethernet application processor interface.
SPI CLOCK POLARITY	0, 1	0 1	Unsigned 32-bit decimal integer Unsigned 32-bit decimal integer	Ignored unless the SPI link type is selected (by the file or by the selection pins). If an integer value greater than 1 is used, it is ignored, and 0 is used instead.

RPG2 HARDWARE DESIGN INTEGRATION GUIDE

Table 48. Application Processor Interface Link Type Parameters (Continued)

Parameter	Usable Variables	Default	Interpreted As	Notes
SPI CLOCK PHASE	0, 1	0 1	Unsigned 32-bit decimal integer Unsigned 32-bit decimal integer	Ignored unless the SPI link type is selected (by the file or by the selection pins). If an integer value greater than 1 is used, it is ignored, and 0 is used instead.
UART BAUD RATE	9600 to 1000000	115200	Unsigned 32-bit decimal integer	The communication side performs minimum and maximum baud rate verification. If an out of range baud rate is selected, the default is used. Certain baud rates are not practical or possible. Other baud rates can cause an unacceptably high error percentage on either the application or communication side. It is up to the customer to select a practical and feasible baud rate that yields an acceptable error percentage on both the application and communication sides.
UART PARITY	NONE, ODD, or EVEN	NONE	String	If a string is detected that is not a member of this list, the default is used.
UART STOP BITS	1	1	7	Reserved

See the [RPG2 Programming User Guide](#) section for more information on how to load the link configuration file, which is the same way that the I/O configuration file is loaded onto the RPG2.

Strapping Option Method

Refer to this section if the **PINS** option is selected in the [link configuration](#) file.

A user can also elect to use the three strapping pins to pick the desired application processor link type. [Table 49](#) details what signals must be logic high and logic low to select a given application processor link type.

Note that the default link type in the board configuration file is what is strapped from the pins. If the user wants a specific applications processor interface, then they can load a link configuration file as part of their integration.

Table 49. Application Processor Interface Strapping Options

Link Type Selector				
Pin 2	Pin 1	Pin 0	Link Type Code	Unified Interface Link Type
0	0	0	0	Ethernet, default
0	0	1	1	Reserved
0	1	0	2	SPI
0	1	1	3	Reserved
1	0	0	4	UART0
1	0	1	5	Reserved
1	1	0	6	Reserved
1	1	1	7	Reserved

CONSIDERATIONS FOR PRODUCTION AND MAINTENANCE

There are three methods for configuring and programming the RPG2 during the production of end use systems:

- ▶ JTAG
- ▶ Ethernet
- ▶ Programming via the application processor interface

When implementing the [Embedded Design](#) the only option for programming out of the box is through a JTAG connection using the SEGGER Embedded Studio toolchain. A serial wire debug connection is not supported. See the [Embedded Design](#) section for more information.

RPG2 HARDWARE DESIGN INTEGRATION GUIDE

JTAG Method

The JTAG method is used during production and maintenance when it comes to the embedded design. The JTAG method is used only during production when used directly with the RPG2 module. Refer to the [RPG2 Programming User Guide](#) section to program the module or the embedded design using this method.

Ethernet Method

The Ethernet method is used only during maintenance when used with a module or an embedded design. The Ethernet method is used primarily for field updates. Refer to the [RPG2 Programming User Guide](#) section of this reference manual to program the embedded design or a module using this method.

Application Processor Interface Method

The application processor interface method is used only during maintenance when used with a module or an embedded design. The application processor interface method is used primarily for field updates. Refer to the [RPG2 Programming User Guide](#) section to program the embedded design or a module using this method.

CONNECTING TIMERS FOR RPG2 EMBEDDED REFERENCE DESIGN

Eight timers in the embedded reference design are connected in the [RPG2 Embedded Reference Design](#). Some timers are used by protocols for different purposes, and some timers are left in the design for future use. The inclusion of the timers is detailed in [Table 50](#).

Table 50. API Strapping Options

Timer	Protocol	Used	Notes
0	EtherNet/IP	No	Unused.
1	EtherNet/IP	No	Unused.
2	EtherNet/IP	No	Unused.
3	EtherNet/IP	No	Unused.
4	EtherNet/IP	No	Unused.
5	EtherNet/IP	No	Unused.
6	EtherNet/IP	No	Unused.
7	EtherNet/IP	No	Unused.
0	PROFINET	No	Unused.
1	PROFINET	No	Unused.
2	PROFINET	No	Unused.
3	PROFINET	No	Unused.
4	PROFINET	Yes	This timer shows synchronization for PROFINET isochronous real time (IRT). For conformance, this timer must be connected to an external test point to verify synchronization.
5	PROFINET	No	Unused.
6	PROFINET	No	Unused.
7	PROFINET	No	Unused.
0	EtherCAT	No	Unused.
1	EtherCAT	No	Unused.
2	EtherCAT	No	Unused.
3	EtherCAT	No	Unused.
4	EtherCAT	No	Unused.
5	EtherCAT	No	Unused.
6	EtherCAT	Yes	Sync 1 pulse for EtherCAT distributed clock functionality.
7	EtherCAT	Yes	Sync 0 pulse for EtherCAT distributed clock functionality.

RPG2 HARDWARE DESIGN INTEGRATION GUIDE

LED BEHAVIOR

The LED behavior is dictated by the protocol or the application. [Figure 173](#) shows the LEDs in the [EV-RAPID](#) used to demonstrate protocol and application behavior. The LEDs from the [RPG2 Embedded Reference Design](#), **LED1/2** and **LED3/4**, are referred to as **MOD** and **NET**, respectively, for this guide and by the ODVA.

LED Color

The LEDs are a specific color in the RPG2 evaluation kits. See [Table 51](#) for the specific LEDx colors.

Table 51. NET Behavior Descriptions EtherNet/IP Protocol

LED Number	Color
LED1	Green
LED2	Red
LED3	Green
LED4	Red
LED5	Green
LED6	Red
LED7	Green
LED8	Red

Deviating from the color scheme shown in [Table 51](#) can cause conformance failures for some protocols.

Industrial Protocol Specific Behavior

LEDs and EtherNet/IP

The LED behavior of the RPG2 module is specified by the protocol. If the solution is embedded into the hardware design and the LEDs are connected as shown in the reference schematics available on the [main product page](#), the protocol software in use controls the behavior of the LEDs.

Table 52. MOD Behavior Descriptions for EtherNet/IP Protocol

MOD Indicator	System Status
Off	No power
Steady Green	Device operational
Flashing Green	Standby
Steady Red	Major fault
Flashing Red	Minor fault
Flashing Red or Green	Self test

Table 53. NET Behavior Descriptions EtherNet/IP Protocol

NET Indicator	System Status
Off	Not powered or no IP address
Steady Green	Connected
Flashing Green	No connections
Steady Red	Duplicate IP
Flashing Red	Connection timeout
Flashing Red or Green	Self test

LEDs and PROFINET

The LED behavior of the RPG2 module is specified by the protocol. If the solution is embedded into the hardware design and the LEDs are connected as shown in the reference schematics available on the [main product page](#), the protocol software in use controls the behavior of the LEDs. Profibus International has no requirements for labeling or naming LEDs. For the purposes of this guide, the LEDs are referred to as **NET** and **MOD**.

RPG2 HARDWARE DESIGN INTEGRATION GUIDE

Table 54. MOD Behavior Descriptions for PROFINET Protocol

MOD Indicator	System Status
Off	Startup
Blinking Green	Blink test
Solid Green	Configuration complete

Table 55. NET Behavior Descriptions for PROFINET Protocol

NET Indicator	System Status
Off	Startup
Blinking Red	No PLC connection
Solid Green	PLC connected

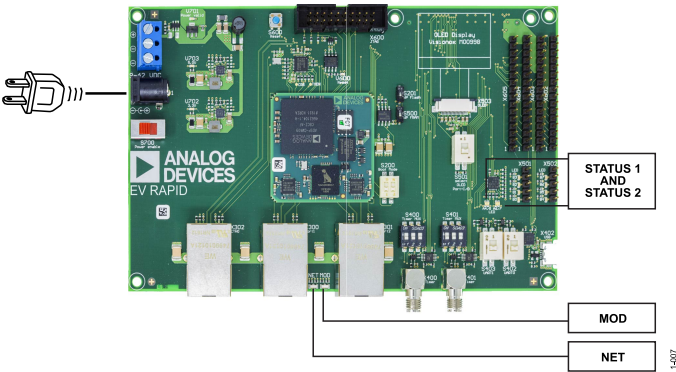


Figure 173. Labels of LEDs in EV-RAPID

LEDs and EtherCAT

The LED behavior of the RPG2 module is specified by the protocol. If the solution is embedded into the hardware design and the LEDs are connected as shown in the reference schematics available on the [main product page](#), the protocol software in use controls the behavior of the LEDs.

For EtherCAT, the term used for the devices are RUN and ERR as opposed to NET and MOD. **LED1/2** and **LED3/4** are referred to as **RUN** and **ERR**, respectively, for this guide and by ETG.

Table 56. NET Behavior Descriptions for the EtherCAT Protocol

RUN Indicator	System Status
Off	Initialization
Blinking	Preoperational
Single Flash	Safe operational
Double Flash	Not applicable
On	Operational

Table 57. MOD Behavior Descriptions for the EtherCAT Protocol

ERR Indicator	System Status
Off	No error
Blinking	Invalid configuration
Single Flash	Local error
Double Flash	Process data watchdog timeout or EtherCAT watchdog timeout
On	Application controller failure

RPG2 HARDWARE DESIGN INTEGRATION GUIDE

LEDs and Modbus/TCP

The LED behavior of the RPG2 module is specified by the protocol. If the solution is embedded into the hardware design and the LEDs are connected as shown in the reference schematics available in the [main product page](#), the protocol software in use controls the behavior of the LEDs. There are no LED labeling requirements for a Modbus/TCP device. The LEDs are referred to as **NET** and **MOD**, respectively.

Table 58. MOD Behavior Descriptions for Modbus/TCP Protocol

MOD Indicator	System Status
Off	Configuration
Red On	Connecting (IP not OK, link down)
Red On	Connecting (IP not OK, link up)
Green Flashing	Run (IP OK, link down)
Green Flashing	Run (IP OK, link up)

Table 59. NET Behavior Descriptions for Modbus/TCP Protocol

NET Indicator	System Status
Off	Configuration
Red On	Connecting (IP not OK, link down)
Off	Connecting (IP not OK, link up)
Red On	Run (IP OK, link down)
Green On	Run (IP OK, link up)

LEDs and POWERLINK

The LED behavior of the RPG2 module is specified by the protocol. If the solution is embedded into the hardware design and the LEDs are connected as shown in the reference schematics available on the [main product page](#), the protocol software in use controls the behavior of the LEDs. The LED indicators have no relationship to the Ethernet Powerlink Standardization Group (EPSG) specification. The LEDs are described as **NET** and **MOD**, respectively, for this guide, however, there are no labeling requirements as such for a POWERLINK device.

Table 60. MOD Behavior Descriptions for POWERLINK Protocol

MOD Indicator	System Status
Off	Initialization
Solid Green	Unified interface initialization complete
Solid Red	Error initializing the unified interface

Table 61. NET Behavior Descriptions for POWERLINK Protocol

NET Indicator	System Status
Off	No PLC connection
Solid Green	Connected to a PLC
Solid Red	Error or disconnect from a PLC

LEDs Bootloader and System Startup

If the solution is embedded into the hardware design and the LEDs are connected as shown in the reference schematics available in the [main product page](#), the application processor software in use controls the behavior of the LEDs. Note that the Status 1 and Status 2 LEDs map directly to LED4 and LED5 and LED6 and LED7 and are optional for a user. The LEDx behavior has meaning during startup and not during protocol operation as the protocol specific LED behavior detailed in [Table 62](#) to [Table 66](#). The ultimate effect is that if the startup happened and there are no errors on the RPG2 platform, Status 1 and Status 2 are solid green in a steady state and ready for communication from the application processor.

Optional Status LEDs

There are four LEDs on the [RPG2 Embedded Reference Design](#) that have some meaning and will behave a certain way when controlled by the software. The inclusion of these LEDs is optional.

In [Table 66](#), Status 1 corresponds to LED4 and LED5 and Status 2 corresponds to LED6 and LED7.

RPG2 HARDWARE DESIGN INTEGRATION GUIDE

Table 62. Security Status LED Behavior Descriptions for Bootloader

Bootloader Error Status	MOD	NET	Status 1	Status 2
Invalid I/O Signature	Green	Yellow	Off	Yellow
Invalid Net Signature	Yellow	Green	Off	Yellow
Invalid I/O Header	Red	Off	Off	Yellow
Invalid Net Header	Off	Red	Off	Yellow
Invalid Public Key	Yellow	Yellow	Off	Yellow
Invalid Secure Boot Header	Yellow	Red	Off	Yellow
Invalid Image Version	Red	Yellow	Off	Yellow
I/O, Hardware, or Platform Error	Red	Red	Off	Yellow

Table 63. LED Behavior Descriptions for Application Error Status

Bootloader Step Progress Status	MOD	NET	Status 1	Status 2
Invalid Board Configuration (Must Be Steady State)	Off	Off	Off	Off
Invalid Hardware	Red	Red	Red	Red
Hardware Driver Error	Off	Off	Off	Red
Invalid Network Application	Red	Off	Off	Red
Initialization Error	Off	Red	Off	Red

Table 64. LED Behavior Unified Interface Status

Bootloader Step Progress Status	MOD	NET	Status 1	Status 2
Startup Done Ready for Messages (Follows the Signal SDONE)	Off	Off	Green	Off
Fatal Internal Error	Off	Off	Red	Off

Table 65. LED Behavior Protocol Execution Status

Bootloader Step Progress Status	MOD	NET	Status 1	Status 2
Update Found, Authentication in Progress	Off	Off	Off	Green
Application Verification and Download in Progress	Off	Off	Off	Green
Boot Process Complete	Off	Off	Off	Green

Table 66. Status LED Behavior Descriptions for the Bootloader

Bootloader Step Progress Status	Status 1	Status 2
Update Found, Authentication in Progress	Green	Not applicable
Application Verification and Download in Progress	Not applicable	Green
Boot Process Complete	Green	Green

RPG2 WEB SERVER USER GUIDE

FEATURES

- ▶ TLS with a maximum of two client connections at a time
- ▶ CGI using SSI directives
- ▶ HTTP Digest Authentication
- ▶ Firmware upgrade

EQUIPMENT NEEDED

- ▶ [RPG2-EVK-PNZ](#)
- ▶ [RPG2-EVK-ENZ](#)
- ▶ [RPG2-EVK-ECZ](#)

DOCUMENTS NEEDED

- ▶ [RPG2 Unified Interface User Guide](#) section

SOFTWARE NEEDED

- ▶ [RPG2 Profinet software](#)
- ▶ [RPG2 EtherNet/IP software](#)
- ▶ [RPG2 EtherCAT software](#)

GENERAL DESCRIPTION

To simplify interfacing with multiple industrial automation protocols, the RapID Platform Generation 2.0 (RPG2) solution uses Unified Interface, which is a protocol between the module and the host processor that isolates the host from network protocol details. The Unified Interface consists of a set of tools that allows users to define the common interface (CI) data envelope used by the device. This results in CI configuration data that, when loaded into the RPG2 solution, allows the RPG2 to expose the data to an industrial automation network. The CI item types supported include input and output data, configuration data, and diagnostic data.

The data associated with these item types is also accessed in the RPG2 Web Server. In addition to the CI data types, the RPG2 Web Server can also access nonCI data types, such as the module IP address and subnet mask. The RPG2 Web Server uses a simple item naming syntax to read and write the data associated with each item, whether CI or nonCI. The user provided web content uses this item naming syntax combined with a simple set of common gateway interface (CGI) functions to operate the data.

To address the need for security, the RPG2 Web Server provides the following two levels of password protection: administrator (admin) and user. Password protection can protect any particular web page. In addition, to enhance the web experience, the HTML content can include cascading style sheets or JavaScript elements.

INTRODUCTION

The RPG2 supports an embedded RPG2 Web Server with the following features:

- ▶ Transport layer security (TLS) with a maximum of two client connections at a time
- ▶ CGI using server side include (SSI) directives
- ▶ HTTP Digest Authentication
- ▶ A firmware upgrade

When a browser establishes an initial connection to the RPG2 Web Server, there is usually no specific file name requested. In this case, the RPG2 Web Server uses the default file name of index.html. Therefore, when a user creates and loads actual web content onto the RPG2 module, the user must have a file with the index.html name. (If users do not load any web content into the RPG2 module, the web server serves a simple built-in version.) If users have loaded specific web content and that content includes an **index.html** file, that file is served, and the built-in version is not used. Note that the RPG2 Web Server files are stored in the **/public/web-server/** folder in the flash file system. Certificates and keys are stored in the **/private/pki/** folder. Refer to the [Loading RPG2 Web Server Files](#) section for additional information on how to load the RPG2 Web Server files to the file system. The file system contains the private and public directories.

The CGI functions within the web browser use specific SSI directives to perform the reading and writing of the device data work. These directives take the form of special text strings that are embedded within the HTML content stored within the RPG2 module. The directives look

RPG2 WEB SERVER USER GUIDE

like a simply comment to any HTML parser (that is the browser of the user). However, the text of these directives is normally never encountered by the browser because these directives are intercepted by the RPG2 Web Server before these directives are transmitted to the browser. In fact, as the RPG2 Web Server is transmitting the web content, it is also examining the stream of text transmitted looking for these directives to perform the work requested.

Note that the SSI directive examination process is only performed if the web content file name has an .shtml or .stm file extension. If the file has any other file extension (such as .html), the file transmits without this examination, resulting in any embedded SSI directives being ignored. Any file that contains SSI directives must have a filename that ends in .shtml or .stm for the SSI directives to work properly, which means SSI directives cannot be used for homepages. However, if users must use SSI directives on the first page that is served by the RPG2 Web Server, users can use the automatic redirection feature. When using this feature, the normal home page file (**index.html**), when requested, causes the browser to redirect to another page, for example, **index.stm**, where the SSI directives can be used.

The HTML text for this redirection follows:

```
<html>
<head>
<meta http-equiv="refresh"
content="0;URL='/index.stm'">
</head>
<body>
Redirecting index.html to index.stm </body>
</html>
```

It is also possible to make any page reload or refresh itself by using the following code inside the head tag:

```
<meta http-equiv="refresh" content="5">
```

RPG2 WEB SERVER SECURITY

User and Administrative Level Security

The RPG2 Web Server provides two permission levels: admin and user, with a distinct, 8-character password for each level. The admin permission level allows access to all web content (that is, insecure pages, user level pages, and admin level pages). However, the user permission level only allows access to insecure and user level pages. Note that access to admin level pages is not provided to users. The passwords are stored in a flash file system, and these passwords are used to allow the creation of secure web pages. To create a secure web page, use a prefix on the file name. Use the **admin-** prefix to secure a web page at the admin level or the **user-** prefix to secure a web page at the user level.

Each time the RPG2 Web Server is asked to serve a page, it looks at the file name to see if it has one of the special security prefixes. If it does, the RPG2 Web Server forces the browser to ask for a username and password. To obtain access to a secured page, the user must enter a username that is either user or admin and a password. Note that the usernames are fixed. The password that is entered must match the password that is stored in the flash file system for that security level. The default passwords for both levels are password. However, passwords can be changed by using the **ReadPoint.cgi** function. The **ReadPoint.cgi** function is invoked by using the proper SSI code embedded into a page. See the [ReadPoint.cgi Function](#) section and [Table 72](#) for additional information on the **ReadPoint.cgi** function. Note that this function must only be done using a secure web page with admin level security.

If the user has forgotten the passwords, it is possible to reset the passwords back to their defaults. Resetting the passwords can be handled by deleting the password file (**/private/admin-passwd.txt**). It is up to the specific application on how to reset the password.

TLS Level Security

Certificate Authorization

TLS mutual authentication is not supported because the RPG2 Web Server does not verify the certificate of the client. The web browser checks the authenticity of the web server by verifying the certificate of the web server sent during the TLS handshake.

RPG2 WEB SERVER USER GUIDE

For a device certificate to be trusted by web browsers, the certificate must be authentic and issued by a trusted certificate authority which is typically embedded in the trusted store of the browser. If the browser does not support a given root certificate authority (CA), it must be installed in the browser. The certificate of the device can be signed by the root CA or one or more intermediate CAs.

A device specific certificate must be placed in the respective location in the device where the mbed TLS fetches and validates from. (mbed TLS is an implementation of the TLS and SSL protocols, and the respective cryptographic algorithms and support code required.)

Private Key

Private key with passphrase is not supported in the present implementation. Private key size restrictions (if any) as per the standard do apply.

For the Rivest–Shamir–Adleman (RSA) public key cryptosystem, standard key sizes are 1024, 2048, 4096, and 8192 bits. For an Elliptic curve cryptosystem, the standard sizes are 192, 224, 256, 384, 512, and 521 bits. These are all the sizes supported by mbed TLS and, therefore, this solution as well (mbed TLS, Version 2.5.1). There are no special limitations on the private key size in the firmware.

Device Specific Certificate and Key Files

Place the device certificate and the private key files in the flash file system of the device in the **/private/pki** directory. The RPG2 Web Server identifies the certificate with the **server.crt** file name and the key with the **server.key** file name. Use any offline method to copy these files to the file system, such as a file loader app.

When the certificate must be changed, for example, a different certificate or a present certificate expires, both the key files and the certificate must be generated for successful authentication, not just the certificate alone.

The new certificate is effective only if any of the following actions are performed:

- ▶ The application calls the application programming interface (API), **CI_WebserverCertificateReload()**.
- ▶ Reboot of the board.

This solution supports certificates signed by intermediate CA. The device where the browser is running must install the intermediate CA, which signed the device under test (DUT) certificate, to properly verify the certificate chain.

TLS Connection

This solution supports only TLS Version 1.2.

The following cipher suites are supported for TLS handshake:

- ▶ TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- ▶ TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- ▶ TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- ▶ TLS_ECDHE_ECDSA_WITH_AES_128_CCM
- ▶ TLS_DHE_RSA_WITH_AES_128_CCM
- ▶ TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- ▶ TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- ▶ TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- ▶ TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- ▶ TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- ▶ TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- ▶ TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8
- ▶ TLS_DHE_RSA_WITH_AES_128_CCM_8

The elliptic curves with the following order are supported for Elliptic Curve Diffie Hellman (ECDHE) key exchange:

- ▶ Secp192k1
- ▶ Secp192r1
- ▶ Secp224k1

RPG2 WEB SERVER USER GUIDE

- ▶ Secp224r1
- ▶ BrainpoolP256r1,
- ▶ Secp256k1
- ▶ Secp256r1
- ▶ BrainpoolP384r1
- ▶ Secp384r1
- ▶ BrainpoolP512r1
- ▶ Secp521r1

Currently, TLS is configured to support a maximum of two concurrent TLS sessions.

The present implementation uses 30 seconds as timeout inactivity on the TLS session to expire. After timeout, the session terminates, and the client must perform a new TLS connection, if required.

CI ITEM, NAME REFERENCE SYNTAX

To the RPG2 module, the CI data is a simple byte stream, and the data has no inherent format. However, to the user, the data is typically highly structured, which means that the RPG2 Web Server must be provided with a way to know how to reference these structured data items within the input and output byte stream. The RPG2 Web Server understands a simple item naming syntax. To demonstrate this, the example configuration data provided with the RPG2 module can be used, which is found in the [main product page](#) under the **IO Configuration Tool** download. The item data can be zoomed into to access just the desired byte(s) or bit(s). The example configuration that is provided with the RPG2 module shows a typical device with the following input and output data format:

- ▶ 16 discrete digital input and output lines identified by the Common Interface ID Number 500.
- ▶ A pair of 16-bit scaled analog input and output signals identified by the Common Interface ID Number 501.
- ▶ A single 16-bit control register (sent to the RPG2 module, not provided by it) identified by the Common Interface ID Number 502.

Note that although it is not specifically described as such in this example, the control register can represent device configuration information (any particular input item can represent device diagnostic data).

To the RPG2 module, the digital data item is simply a pair of bytes exchanged with the host, and the analog data is simply a series of four such bytes. If using the Unified Interface, the host sees this data packaged together in a single package, or if the user is a developer solution user, code of the user reads each CI item by its CI item handle. Either way, the host (or developer solution code) knows how to interpret the data in exact detail. The RPG2 Web Server, however, runs inside the RPG2 module and has no additional information about the meaning of the data. Therefore, the question becomes how does the RPG2 Web Server reference a single, arbitrary bit from the 16 bits of digital data used by Item 500? Or, how does the RPG2 Web Server reference the second analog value from the four bytes used by Item 501? The answers to these questions are provided in the item naming syntax that follows.

The RPG2 Web Server can support bit, bit range, byte, word, and double word (dword) data formats and can reference any of these formats from any portion of the input and output data. The HTML content provided by the user to the RPG2 Web Server uses the SSI directives to provide the CGI functions with a CI item name using the following basic structure:

```
point_vvv_xx_y_szl
```

This name is composed of a series of fields delimited by the underscore (`_`) character. The fields and associated descriptions are detailed in [Table 67](#).

Table 67. Bits and Fields

Field	Description
point	The point field is a fixed string that begins the item name.
vvv	The vvv field is the CI item ID number (for example, 500).
xx	The xx field is the CI item instance number. If the user wanted to reference the second instance of the vvv item, the user uses 01. If the user is only using a single instance of the vvv item, this field must always be 00.
y	The y field is the CI item data direction. Use the i character for the input data associated with the vvv item or the o character for the output data associated with the vvv item.

RPG2 WEB SERVER USER GUIDE

Table 67. Bits and Fields (Continued)

Field	Description
s	The s field is an optional prefix to indicate that the referenced data is signed. If this field is omitted, the data is considered unsigned.
z	The z field defines the specific bit, bit range, byte, word, or dword of the data to use. See the following text and Table 68 for additional information on this field.
l	The l field is an optional suffix that causes the data to be interpreted as little Endian. If this field is omitted, the data is interpreted as big Endian.

The input and output direction can be confusing. Here, the inputs are defined as data that is produced by the host (that is, input to the network) and the outputs as data that is consumed by the host (that is, output from the network.)

The z field is where users can specifically describe which part of the data the user wants to reference. This field can be a single character, such as b, to simply specify the first byte of the data, or this field can be a multiple character string, such as d3.24-26 to denote a specific range of bits (Bits[26:24] within a specific dword (the third dword) in the data).

Specifically, the z field begins with the data size designator followed up by the data reference specifics. The first (and possibly only) character is one of the characters listed in Table 68, and it denotes the base size of the referenced data.

Table 68. Z Field

Character	Description
b	Byte (8 bits)
w	Word (16 bits)
d	Dword (32 bits)
f	Float (IEEE 754, 32-bit single precision)
l	Double (IEEE 754, 64-bit double precision)

If there is only one character in the z field, the resulting data is a simple byte, word, dword, or floating point value, and no more explanation is needed. The following characters, however, if present, can be used to select a more specific part of the data. For example, using a b as the data size designator allows the options shown in Table 69.

For the other data size designators (w or d), the result is similar, except that the base data being referenced is more than a single byte.

Table 69 list the f or l size designator options.

Table 69. Base Data Larger Than a Byte

Size Designator	Description
f or l	References a floating point value starting at the first byte of the item data
fn or ln	References a floating point value starting at the nth byte of the item data
f.x-y or l.x-y	References a floating point value starting at the first byte of the item data and format (for reading) using an x character field with y digits to the right of the decimal point
fn.x-y or ln.x-y	References a floating point value starting at the nth byte of the item data and format (for reading) using an x character field with y digits to the right of the decimal point

Use the f for single precision values or the l for double precision values.

Also, the s prefix and/or the l suffix (described in Table 67) can be included to further refine the resulting data used by the RPG2 Web Server.

Because this method can reference any data item defined within the CI, it can be used to reference any of the internal data types such as input and output and configuration or diagnostic data. Therefore, this technique allows the returned data to be

- ▶ Extracted (read) from or inserted (written) into the item data in a variety of ways
- ▶ Shifted right, if necessary (to become zero based)
- ▶ Interpreted as either signed or unsigned
- ▶ Interpreted as either big Endian or little Endian

These items can be accessed using the same CGI functions as the CI data items. Note that some items are read only.

RPG2 WEB SERVER USER GUIDE

Table 70. Bit(s) Reference

Designator	Reference	Shift	Signed	Returns	Conditions for x and y
bn	References the nth byte of the item data	Not applicable	Signed or unsigned	Returns 0 to +255 if unsigned or -128 to +127 if signed	Not applicable
b.x	References the xth bit of the first byte of the item data	Not applicable	Not applicable	Returns 0 or 1	x must be less than 8
b.x-y	References Bit x through Bit y of the first byte of the item data	Shifted right to be returned as a zero based number	Never signed	Not applicable	x must be less than y, and both x and y must be less than 8
bn.x-y	References Bit x through Bit y of the nth byte of the item data	Shifted right to be returned as a zero based number	Never signed	Not applicable	x must be less than y, and both x and y must be less than 8
br.x-y	Bit range (Bits[x: y]) of the first byte of the item data without shifting the data	Not applicable	Never signed	Not applicable	x must be less than y, and both x and y must be less than 8
brn.x-y	Bit range (Bits[x:y]) of the nth byte of the item data without shifting the data	Not applicable	Never signed	Not applicable	x must be less than y, and both x and y must be less than 8

NONCI ITEM, NAME SYNTAX

In addition to the CI data items (input and output, and configuration or diagnostic), the RPG2 Web Server must access other nonCI data within the RPG2 module such as the IP address and subnet mask of the device. Accessing this data is much simpler than the CI data because the format is inherently defined by the data type. For example, accessing the IP address of the device always results in a set of four bytes and has a typical presentation as a dotted string surrounded by quotes (“192.168.21.12”). A list of the nonCI data items including the RPG2 Web Server reference item name is provided in [Table 71](#).

Table 71. Item Name Reference Table

Data Item	RPG2 Web Server Name	Format	Comment
Admin Password	apassword	Text	Maximum length 8 characters
User Password	upassword	Text	Maximum length 8 characters
IP Address	ipaddress	Dotted text	For example, xxx.xxx.xxx.xxx
Subnet Mask	subnet	Dotted text	For example, xxx.xxx.xxx.xxx
Gateway	gateway	Dotted text	For example, xxx.xxx.xxx.xxx
DNS Server 1	dns1	Dotted text	For example, xxx.xxx.xxx.xxx
DNS Server 2	dns2	Dotted text	For example, xxx.xxx.xxx.xxx
Hostname	hostname	Text	Maximum length 63 characters
Domain Name	domainname	Text	Maximum length 48 characters
Dynamic Host Configuration Protocol (DHCP)/Static Flag	dhcpenable	Text flag	1 = enable 0 = disable
Port 1 Ethernet Link Speed	eth1speed	Text flag	1 = 100 Mbps 0 = 10 Mbps
Port 1 Ethernet Link Duplex	eth1duplex	Text flag	1 = full 0 = half
Port 1 Ethernet Link Autonegotiation Enable	eth1auto	Text flag	1 = enable 0 = disable
Port 2 Ethernet Link Speed	eth2speed	Text flag	1 = 100 Mbps 0 = 10 Mbps
Port 2 Ethernet Link Duplex	eth2duplex	Text flag	1 = full 0 = half
Port 2 Ethernet Link Autonegotiation Enable	eth2auto	Text flag	1 = enable 0 = disable
Industrial Ethernet Protocol in Use	protocol	Text (read only)	Maximum length = 32 characters
NI Version	niversion	Text (read only)	Maximum length = 8 characters

RPG2 WEB SERVER USER GUIDE

Table 71. Item Name Reference Table (Continued)

Data Item	RPG2 Web Server Name	Format	Comment
SW Version	swversion	Text (read only)	Maximum length = 8 characters
MAC Address	macaddress	Colon delimited text (read only)	For example, xx:xx:xx:xx:xx:xx
Device Serial Number	serialnum	Text (read only)	Maximum length = 8 characters
Device Hardware Revision	hwrevision	Text (read only)	Maximum length = 8 characters

SSI DIRECTIVES

To allow the RPG2 Web Server to distinguish between the normal HTML content and the commands the web designer intends it to internally process, use SSI directives. These directives have a simple syntax, which follows:

```
<!--#directive="value"-->
```

Note that the syntax does not allow spaces between the leading **<!--** sequence and the **#directive** or between the **"value"** and the trailing **-->** sequence. In addition, the quotation marks are required.

While some commercial web servers (such as Apache) support a wide variety of SSI directives, the embedded RPG2 Web Server in the RPG2 module only supports two: include and exec. The **include** directive is used to include a common piece of HTML code within an HTML file to simplify things like HTML navigation bars. An example of the syntax for an **include** directive follows:

```
<!--#include="/navbar.html"-->
```

The **exec** directive is used to execute a CGI function that reads or writes a (CI or nonCI) data item, opens the HTML Tunnel (that is described as follows), or requests a system reboot. An example of the syntax for the **exec** directive follows:

```
<!--#exec="/ReadPoint.cgi?  
point_500_00_o_b.0"-->
```

or

```
<!--#exec="/WritePoints.cgi?  
point_500_00_o_b.0&value=0n"-->
```

The value portion of the SSI is further delimited into subfields by a question mark (?) and/or by an ampersand (&) symbol. The question mark separates the CGI function name from its arguments, and the ampersand separates the arguments into multiple subarguments, which is shown in the previous examples.

Making use of these directives is simply a matter of embedding the directives into the appropriate location within the HTML file so that the directives can accomplish their intended purpose. The **include** directive causes the specified file to be inserted in the HTML stream, and the **exec** directive causes the specified CGI function to be executed treating the data after the question mark (?) as an argument. The **ReadPoint.cgi** function causes the referenced data item to be read and its value inserted into the HTML stream, while the **WritePoints.cgi** function causes one or more items to be written to the provided value(s). See the [ReadPoint.cgi Function](#) section and the [WritePoints.cgi Function](#) section for additional information on these functions.

RPG2 WEB SERVER USER GUIDE

CGI FUNCTIONS

The following four CGI functions are provided by the RPG2 Web Server:

- To read a CI or nonCI data item, use **/ReadPoint.cgi**.

```
<!--#exec="/ReadPoint.cgi?
point_500_00_o_b.0&option=0"-->
```

- To write one or more CI or nonCI data items, use **/WritePoints.cgi**.

```
<!--#exec="/WritePoints.cgi"-->
```

- To open the HTML Tunnel, use **/HTML_Tunnel.cgi**.

```
<!--#exec="/HTML_Tunnel.cgi"-->
```

- To request a system reboot, use **/reset.cgi**.

```
<!--#exec="/reset.cgi"-->
```

Note that the slash character (/) is an integral part of the CGI function name and must be present.

The user can register specific CGI handlers using the **CI_registerCgiEntry** API. Only call this function after the configuration is complete (**CI_ConfigComplete()**) in the **NI_ConfigComplete** (see the [RPG2 Unified Interface User Guide](#) section for more detail on this API function).

CI_registerCgiEntry API

This function shows how to register a CGI entry using the RPG2 Web Server functionality.

```
int32_t CI_registerCgiEntry(CI_cgiEntry_t *entry_p, uint32_t numEntries)
```

The entry_p argument is a CI_cgiEntry_t struct as follows.

```
typedef struct {
    const char *url_p;
    CI_cgiHandler_tp handler_p;
} CI_cgiEntry_t;
```

Where the CG handler, CI_cgiHandler_tp, is defined as the following:

```
typedef void (*CI_cgiHandler_tp)(CI_cgiRequest_t, CI_cgiResponse_t *);
```

The CGI request structure, CI_cgiRequest_t, is defined as the following:

```
typedef struct {
    CI_cgiMetaVariable_t metaVar;
    const char *msgBody_p; /*Request message body */
} CI_cgiRequest_t;
/*CGI meta variables structure*/
```

RPG2 WEB SERVER USER GUIDE

```
typedef struct {
    int contentLength; /* Content-Length: size of message body */
    int serverPort; /* Server's tcp port number */
    int remotePort; /* client tcp port number */
    const char *authType_p; /* Authentication scheme(Basic, Digest) */
    const char *contentType_p; /* Requested message content type */
    const char *documentArgs_p; /* Query string of the active SSI document */
    const char *queryString_p; /* URL encoded search or parameter string */
    const char *remoteUser_p; /* User identification
    string supplied by as part of the user authentication
    */
    const char *requestMethod_p; /* HTTP request
    method ('GET', 'POST', 'HEAD', etc., )*/
    const char *scriptName_p; /* URI path which
    identify the CGI script*/
    const char *remoteAddress_p; /* User identification
    string supplied by client as part of user
    authentication */
} CI_cgiMetaVariable_t;
```

The CGI response structure, `CI_cgiResponse_t`, is defined as the following:

```
typedef struct {
    char *msgBody_p; /* Response buffer */
    int msgBodyMaxLen; /* Maximum length of
    response buffer */
    const char *contentType_p; /* content type of the
    response */
    int msgBodyLen; /*Length of the response */
} CI_cgiResponse_t;
```

Example Code

The following is the example code for CGI handlers:

```
CI_cgiEntry_t cgiTable_g[] = {
    { "/Example1.cgi", CgiHandler1 },
    { "/ Example2.cgi", CgiHandler2 },
}
/* CGI Handler */
void CgiHandler1(CI_cgiRequest_t req, CI_cgiResponse_t *resp_p)
{
    char* handlerResp_p = "Example response";
    if (resp_p) {
        resp_p->msgBodyLen = snprintf(resp_p->msgBody_p,
        resp_p->msgBodyMaxLen,
        "%s",
        handlerResp_p);
    }
}
/* Register CGI handlers */
int32_t RegisterCgiHandlers(void)
{
    {
```

RPG2 WEB SERVER USER GUIDE

```
uint32_t numEntries = sizeof(cgiTable_g)/sizeof(CI_cgiEntry_t);
return CI_RegisterCgiEntry(cgiTable_g, numEntries);
}
```

ReadPoint.cgi Function

An example of the use of the **ReadPoint.cgi** function follows:

```
<!--#exec="/ReadPoint.cgi?
point_500_00_o_b.0&option=0"-->
```

Table 72 shows how the **ReadPoint.cgi** function is broken down.

The purpose of this CGI function is to read a CI or a nonCI data item and to produce the appropriate text for the HTML form or page being transmitted. The text generated by the **ReadPoint.cgi** function is inserted directly into the HTML stream being transmitted and can simply display the value of the item or be used within a form element to cause that element to be modified in a useful way. This function can cause a radio button to become pushed or a check box to become checked. It has the CGI function name of **/ReadPoint.cgi** and always has a single CI or nonCI item name reference as an argument. Additional arguments may or may not be present. If other arguments are, these arguments are delimited by an ampersand (&) character.

Note that additional arguments are optional, and intended uses for additional arguments are detailed in Table 73.

If no argument is present, the **ReadPoint.cgi** function simply reads the item and directly returns its value as a quoted string.

Table 72. ReadPoint.cgi Function Breakdown

Description	Text
SSI Directive Start	<!--#exec="
CGI Function Name	/ReadPoint.cgi
CGI Function Name and Argument Delimiter	?
CGI Function Arguments Delimiter (the & Character)	point_500_00_o_b.0&option=0
SSI Directive End	"-->

Table 73. ReadPoint.cgi Additional Arguments

Argument Text	Used With	Effect
radio_On	Radio buttons used to control single-bit values	Modifies the ReadPoint.cgi function to return the text checked = checked if the referenced bit is a 1.
radio_Off	Radio buttons used to control single-bit values	Modifies the ReadPoint.cgi function to return the text checked = checked if the referenced bit is a 0.
checkbox	Check boxes used to control single-bit values	Modifies the ReadPoint.cgi function to return the text checked = checked if the referenced bit is a 1.
option=xxx	Dropdown lists used to present enumerated options	Modifies the ReadPoint.cgi function to return the text selected if the referenced item is equal to the option value (that is, the xxx).

WritePoints.cgi Function

See the following for the **WritePoints.cgi** function example:

```
<!--#exec="/WritePoints.cgi?
point_500_00_o_b.0&value=On"-->
```

Table 74 details how the **WritePoints.cgi** function is broken down.

The purpose of this function is to write data to one or more CI or nonCI items. The arguments are always in pairs: name&value=xxx (delimited by the ampersand (&) character.) More than one pair of arguments can be (and usually are) present. Each pair is the name of an item followed by the value to be written to that item. The **WritePoints.cgi** function can be used directly. However, this function is more commonly used to

RPG2 WEB SERVER USER GUIDE

process the data submitted by an HTML form. To process this data, place the **WritePoints.cgi** function into an .stm or .shtml file (usually at the top of the <body> section) without arguments.

```
<!--#exec="/WritePoints.cgi"-->
```

This way, when the web browser submits the HTML form results, the **WritePoints.cgi** function immediately receives this data and can perform the requested item write operations.

HTML_Tunnel.cgi Function

See the following for the **HTML_Tunnel.cgi** function example:

```
<!--#exec="/HTML_Tunnel.cgi"-->
```

Table 75 details how this function is broken down.

The following things happen when the **HTML_Tunnel.cgi** function is called:

- ▶ The HTML Tunnel opens so the host can directly produce HTML content.
- ▶ The HTML request data is received from the web browser and provided to the host.
- ▶ The HTML content is received from the host and provided to the web browser.

When the RPG2 Web Server encounters a request for a page that contains the **HTML_Tunnel.cgi** function, it looks to see if any data was posted to the RPG2 Web Server (that is, the user hit a submit button), and if so, the RPG2 Web Server retrieves this data and sends the data to the host. If not, this function simply sends the URL of the web page to the host. The expected response from the host is for some sort of HTML data to display in place of the **HTML_Tunnel.cgi** function. Once the data is received from the host, the **HTML_Tunnel.cgi** simply sends this data directly to the web browser. It is possible to display custom HTML forms and to interact with the user in complex ways using this feature. See Section 4 of the example RPG2 Web Server content for full details including example code.

Table 74. WritePoints.cgi Function Breakdown

Description	Text
SSI Directive Start	<!--#exec="
CGI Function Name	/WritePoints.cgi
CGI Function Name and Argument Delimiter	?
CGI Function Arguments Delimiter (the & Character)	point_500_00_o_b.0&value=On
SSI Directive End	"-->

Table 75. HTML Tunnel CGI Function Breakdown

Description	Text
SSI Directive Start	<!--#exec="
CGI Function Name	/HTML_Tunnel.cgi
SSI Directive End	"-->

Reset.cgi Function

See the following for the **rest.cgi** function example:

```
<!--#exec="/reset.cgi"-->
```

Table 76 details how the **rest.cgi** function is broken down.

The purpose of the **rest.cgi** function is to request a reset (or a reboot) of RPG2. Like the others, the **rest.cgi** function immediately executes when encountered by the RPG2 Web Server. Users simply place this function onto a page, and when that page is requested, the system

RPG2 WEB SERVER USER GUIDE

automatically and unconditionally resets. If used this simply, this feature may not be that useful. This function is more useful if the browser is directed to ask the user if a reset is really desired.

The key to providing this functionality is to use JavaScript within a standard HTML web page. It is this JavaScript (running within the browser of the user) that asks the user for confirmation, and if the user allows it, triggers the CGI function by requesting the web page that contains the **rest.cgi** function. Two files must be added to the RPG2 Web Server content. One file that has a standard .html file naming extension, contains the JavaScript, and is directly exposed to the user, and one file that has an .stm or .shtml file naming extension, contains the SSI directive, and is not directly exposed to the user. For a complete example of how this is accomplished, see the [Example RPG2 Web Server Content](#) section.

Table 76. Reset CGI Function Breakdown

Description	Text
SSI Directive Start	<!--#exec=
CGI Function Name	/reset.cgi
SSI Directive End	-->

EXAMPLE RPG2 WEB SERVER CONTENT

The following block of code places the SSI directive to call the CGI function, **WritePoints.cgi**, directly into the HTML with no relation to anything else within the HTML. For example, the code block is not used as a part of an input element in a form. Moreover, the code block is placed in the RPG2 Web Server code without any arguments (to **WritePoints.cgi**). At first glance, this may seem confusing and unnecessary, but code block is there for a purpose. Because the pages also have HTML forms, this SSI directive is provided to capture the posted data (when the submit button is pressed).

This function works because these pages all contain what can be referred to as a self referential HTML form. The action attribute of the form tag is set to refer to the same page that contains the form itself. For example, within the radio button form, users see the following:

```
...
<!-- ----- A form to allow outputs to be displayed and modified ----- -->
<form action="/radioform.stm" method="post">
```

Note that the text **action="/radioform.stm"** is contained within the form tag, and this bolded text is what is referred to as the self referential part. That is, as long as the HTML content is in a file named **"radioform.stm"**, submitting the form requests the RPG2 Web Server resource **"radioform.stm"** that redisplayes the same form submitted.

The following is an example use case for input points as these points apply to the **ReadPoint.cgi** function:

```
<html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso8859-1">
<title>RapID Platform Generation 2.0 Input Item Display Page</title>
</head>
<body>
<hr> <h1><b> RapID Platform Generation 2.0 Form To Display Input Points</b></h1> <hr>
Item ID 500, input bit 0 (point_500_00_i_b.0) is
<b><!--#exec="/ReadPoint.cgi?point_500_00_i_b.0"--></b>
<br>
Item ID 500, input bit 1 (point_500_00_i_b.1) is
<b><!--#exec="/ReadPoint.cgi?point_500_00_i_b.1"--></b>
<hr>
<a href="/index.html">Home</a>
</body>
</html>
```

Users may wonder why this seemingly unnecessary call to **WritePoints.cgi** was added. The answer is that this is the CGI function that captures data being posted to the web server when the submit button is pressed.

RPG2 WEB SERVER USER GUIDE

Note that even if users do not use a self referential page, to successfully write to the CI and nonCI data items, the page used to capture and process submitted data must contain this SSI directive.

Display Only Pages (Display Form)

The display only pages display the data item values only (not to change them), shown as **Display Form** in Figure 174. In this case, no HTML forms are required, and the **ReadPoint.cgi** function is the only one used. The SSI directive for this CGI function is simple, consisting of the **ReadPoint.cgi** function and a single item name reference to read as the only argument. For example, see the following:

```
<!--#exec="/ReadPoint.cgi?
point_500_00_i_b.0"-->
```

Each time this code is encountered, the **ReadPoint.cgi** function is called, the referenced item is read, and the resulting HTML text provided is substituted into the HTML stream. The net result is the value of the item inserted into the web page.

The example HTML for such a page can be found in the **RPG2 Sample Web Server Content Suite** found within the ADIN2299 main product page at www.analog.com/adin2299. In the zip folder of this suite, the file name is **display.stm**.

Radio Forms

Radio buttons are used to display and control the value of discrete (that is, single-bit) items. To allow the user to make changes and submit the changes, an HTML form is required. The form uses radio buttons as the input elements, two for each item (bit) controlled. One radio button must specify a value attribute of **On**, and the other attribute must specify a value attribute of **Off**. Each radio button must have its name attribute set to the desired item name reference. In addition, each radio button must include the SSI directive for the **ReadPoint.cgi** function and provide two arguments. (Note that, the argument(s) to the CGI function are the text beginning after the CGI function name delimited by a question mark (?).) The arguments are separated by an ampersand (&) character and are the CI item name with either the **radio_On** or **radio_Off** text. Use the former for the on value radio button (**radio_On**), and use the latter for the off valued radio button (**radio_Off**).

When the web page displays, the **ReadPoint.cgi** function is called twice for each radio button pair. The first time getting **<item name>&radio_On** as an argument, and the second time getting **<item name>&radio_Off** as an argument. The **radio_On** argument causes the function to look at the value of the item and provides **checked="checked"** if the value of the item is true. The function supplies an empty string if the data of the item is false. The **radio_Off** argument causes the function to look at the value of the item and provides **checked="checked"** if the value of the item is false, and an empty string if the value of the item is true, which ensures that the correct radio button is initially selected.

When the web page is submitted for each pair (on/off) of radio buttons, the **WritePoints.cgi** function gets either **<item name>=On** or **<item name>=Off** depending on which radio button is selected.

The HTML for such a page follows. To use this example HTML with the example homepage, it is recommended to name the file **radioform.stm**.

```
<html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso8859-1">
<title> RapID Platform Generation 2.0 Write Form</title>
</head>
<body>
<!-- this line does the item writing by acting on the posted form submission data -->
<!--#exec="/WritePoints.cgi"-->
<hr>
<h1><b> RapID Platform Generation 2.0 Form To Demonstrate Radio Buttons</b></h1>
<hr>
<!-- ----- A form to allow outputs to be displayed and modified ----- -->
<form action="/radioform.stm" method="post">
Item ID 500, output bit 0 (point_500_00_o_b.0):
<input type="radio" name="point_500_00_o_b.0" value="On"
<!--#exec="/ReadPoint.cgi?point_500_00_o_b.0&radio_On"--> > <b>On</b>
<input type="radio" name="point_500_00_o_b.0" value="Off"
```

RPG2 WEB SERVER USER GUIDE

```

<!--#exec="/ReadPoint.cgi?point_500_00_o_b.0&radio_Off"--> > <b>Off</b>
<br>
Item ID 500, output bit 1 (point_500_00_o_b.1):
<input type="radio" name="point_500_00_o_b.1" value="On"
<!--#exec="/ReadPoint.cgi?point_500_00_o_b.1&radio_On"--> > <b>On</b>
<input type="radio" name="point_500_00_o_b.1" value="Off"
<!--#exec="/ReadPoint.cgi?point_500_00_o_b.1&radio_Off"--> > <b>Off</b>
<br>
<input type="submit" value="Submit">
</form>
<hr>
<a href="/index.html">Home</a>
</body>
</html>

```

CheckBox Forms

Check boxes are used to display and control the value of discrete (single-bit) item data. To allow the user to make changes and submit these changes, an HTML form is required. The form uses check boxes as the input elements, two for each item (bit) controlled. One check box is displayed by the browser to accept the input of the user, and the other is present on the form but is not displayed to the user. This second check box is present to tell the **WritePoints.cgi** function which items are controlled by the form. The first check box (that is, the visible one) must specify a value attribute of **On**, and the other check box (that is, the hidden one) must specify a value attribute of **present**. The hidden check box must also specify that it is checked by including **checked="checked"**. Each check box must have its name attribute set to the desired item name. The visible check box must include the SSI directive for the **ReadPoint.cgi** function and provide two arguments. (Note that, the argument(s) to the CGI function is the text beginning after the CGI function name delimited by a question mark (?).) The arguments are separated by an ampersand (&) character and are the CI item name followed by the **checkbox** text.

When the web page displays, the **ReadPoint.cgi** function is called once for each check box pair (that is, for the visible one). When the **ReadPoint.cgi** function is called, the visible check box provides **<item name>&checkbox** as an argument. The argument **checkbox** causes the function to look at the value of the item and provide **checked="checked"** if the item is true. The function provides an empty string if the item is false, which ensures that the check box is initially selected or not selected according to the value of the bit.

When the form is submitted (using an example with two check boxes: A and B), the **WritePoints.cgi** function gets the following when only one check box is checked:

```
<item name A>=On<item name A>=Present<&item name B>=Present
```

If two check boxes are checked, the **WritePoints.cgi** function gets the following:

```
<item name A>=On<item name B>=On<item name A>=Present<&item name B>=Present
```

When the form is submitted, the net result is that the **present** argument is always provided by the hidden check boxes, which allows the function to build a list of what check boxes are present on the form. The function must do this because if a (visible) check box is not checked when the form is submitted, there is nothing in the data posted to the RPG2 Web Server to tell it that the unchecked check boxes are simply omitted from the posted data. This way, because this hidden check box is always checked (**present**), it always posts to the CGI function to let the RPG2 Web Server determine if a visible check box is omitted or not.

The HTML for such a page follows. To use this example HTML with the example homepage, it is recommended to name the file **chkboxform.stm**.

```

<html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso8859-1">

```

RPG2 WEB SERVER USER GUIDE

```

<title> RapID Platform Generation 2.0 Check Box Form</title>
</head>
<body>
<!-- this line does the item writing by acting on the posted form submission data -->
<!--#exec="/WritePoints.cgi"-->
<hr>
<h1><b> RapID Platform Generation 2.0 Form To Demonstrate Check Boxes</b></h1>
<hr>
<!-- ----- A form to allow outputs to be displayed and modified ----- -->
<form action="/chkboxform.stm" method="post">
Item ID 500, output bit 0 (point_500_00_o_b.0):
<input type="checkbox" name="point_500_00_o_b.0" value="On"
<!--#exec="/ReadPoint.cgi?point_500_00_o_b.0&checkbox"--> >
<input hidden style="visibility:hidden;" type="checkbox" name="point_500_00_o_b.0"
value="present" checked="checked">

```

Text Forms

Text boxes are used to display and control the value of numeric (floating point or integer) items. To allow the user to make changes and submit these changes, an HTML form is required. This form uses text boxes as the input elements, one for each item controlled. Each text box must set its name attribute to the desired item name reference. In addition, each text box must include the SSI directive for the **ReadPoint.cgi** function and provide a single argument. (Note that, the argument(s) to the CGI function is the text beginning after the CGI function name delimited by a question mark (?).) The argument is simply the CI item name.

When the web page displays, the **ReadPoint.cgi** function is called once for each text box. Because no additional arguments are provided to the **ReadPoint.cgi** function, this function simply returns the value of the item, which causes the text box to display it.

When the form is submitted, the **WritePoints.cgi** function gets the following:

```
<item name>=128
```

where 128 (as an example) is the numeric text entered by the user.

The HTML for such a page follows. To use this example HTML with the example homepage, it is recommended to name the file **textform.stm**.

```

<html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso8859-1">
<title> RapID Platform Generation 2.0 Text Box Form</title>
</head>
<body>
<!-- this line does the item writing by acting on the posted form submission data -->
<!--#exec="/WritePoints.cgi"-->
<hr>
<h1><b> RapID Platform Generation 2.0 Form To Demonstrate Text Fields</b></h1>
<hr>
<!-- ----- A form to allow outputs to be displayed and modified ----- -->
<form action="/textform.stm" method="post">
Item ID 500, output bit 0 (point_500_00_o_b.0):
<input type="text" name="point_500_00_o_b.0" size="4"
value=<!--#exec="/ReadPoint.cgi?point_500_00_o_b.0"--> >
<br>
Item ID 500, output bit 1 (point_500_00_o_b.1):
<input type="text" name="point_500_00_o_b.1" size="4"

```

RPG2 WEB SERVER USER GUIDE

```
value=<!--#exec="/ReadPoint.cgi?point_500_00_o_b.1"--> >
<br>    <input type="submit" value="Submit">
</form>
```

Pull-Down List Forms (Drop Form)

Pull-down lists (**Drop Form** shown in [Figure 174](#)) allow users to make a selection from an enumerated list. To allow the user to make changes and submit these changes, an HTML form is required. The form uses a select element with multiple option fields. The selected element must have its name attribute set to the desired item name reference. Each option field must have its name attribute set to the text choice of the option, and the value attribute to a numeric value assigned to that choice. In addition, this option must include the SSI directive for the **ReadPoint.cgi** function and provide two arguments. (Note that, the argument(s) to the CGI function is the text beginning after the CGI function name delimited by a question mark (?).) The arguments are separated by an ampersand (&) character and are the CI item name followed by the **option=n** text, where n is the numeric assignment value.

The following block of code is used for implementation of the dropdown menu in conjunction with the **WritePoints.cgi** function:

```
html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso8859-1">
<title> RapID Platform Generation 2.0 Drop Down Form</title>
</head>
<body>
<!-- this line does the item writing by acting on the posted form submission data -->
<!--#exec="/WritePoints.cgi"-->
<hr>
<h1><b> RapID Platform Generation 2.0 Form To Demonstrate Drop Down Menus</b></h1>
<hr>
<!-- ----- A form to allow outputs to be displayed and modified ----- -->
<form action="/dropform.stm" method="post">
Item ID 500, output bit 0 (point_500_00_o_b.0):
<select name="point_500_00_o_b.0">
<option name="Zero" value="0">
<!--#exec="/ReadPoint.cgi?point_500_00_o_b.0&option=0"--> >
Zero
</option>
<option name="One" value="1">
<!--#exec="/ReadPoint.cgi?point_500_00_o_b.0&option=1"--> >
One
</option>
</select>
<br>
Item ID 500, output bit 1 (point_500_00_o_b.1):
<select name="point_500_00_o_b.1">
<option name="Zero" value="0">
<!--#exec="/ReadPoint.cgi?point_500_00_o_b.1&option=0"--> >
Zero
</option>
<option name="One" value="1">
<!--#exec="/ReadPoint.cgi?point_500_00_o_b.1&option=1"--> >
One
</option>
</select>
<br>
<input type="submit" value="Submit">
</form>
```

RPG2 WEB SERVER USER GUIDE

```
<hr>
<a href="/index.html">Home</a>
</body>
</html>
```

When the web page displays, the **ReadPoint.cgi** function is called once for each possible option choice. The function gets the item name and the possible option as an argument each time. Because the argument has the keyword **option** added, the function compares the assigned numeric value of the option to the actual value of the item, and, if the numeric value and actual value of the item match the return selected. If the numeric value and actual value of the item do not match, nothing returns, which causes the correct option to be initially selected in the dropdown list.

When the web page is submitted, the **WritePoints.cgi** function gets the new value in this form:

```
<point_name>=n
```

where n is the assigned numeric value.

The example HTML for such a page can be found in the **RPG2 Sample Web Server Content Suite** found within the ADIN2299 main product page at www.analog.com/adin2299. In the zip folder of this suite, the file name is **dropform.stm**.

Password Protected Forms (Security Form Listed)

There is nothing special about a secure web page other than the file name prefix. Any HTML page (whether it has SSI content or has an .stm or .html file extension) can be protected. To create a secure web page, simply change the file name so that it starts with a special prefix. Administration level pages start with the prefix text **admin-**, and the user level pages start with the prefix **user-**. (See [Figure 174](#) for **Security Form Listed**.)

To allow the user to see and change the passwords, use a text box form. The item name reference follows the nonCI item naming syntax that follows. To change the administration level password, use **apassword**. To change the user level password, use **upassword**.

The HTML for such a page follows. To use this example HTML with the example homepage, it is recommended to name this file **admin-passwords.stm**.

```
<html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso8859-1">
<title> RapID Platform Generation 2.0 HTML Password Set Page</title>
</head>
<body>
<!-- this line does the item writing by acting on the posted form submission data -->
<!--#exec="/WritePoints.cgi"-->
<hr>
<h1><b> RapID Platform Generation 2.0 Secure Form To Allow Passwords to be Changed</b></h1>
<hr>
<b>Note: Access to this page requires admin level security.</b>
<hr>
<!-- ----- A form to allow non-CI items to be displayed and modified ----->
<form action="/admin-passwords.stm" method="post">
The <b>admin</b> level user name is <b>"admin"</b>, the password is
<input type="text" name="apassword" size="12"
value=<!--#exec="/ReadPoint.cgi?apassword"--> >
<br>
The <b>user</b> level user name is <b>"user"</b>, the password is
<input type="text" name="upassword" size="12"
value=<!--#exec="/ReadPoint.cgi?upassword"--> >
```

RPG2 WEB SERVER USER GUIDE

```
<br>
<input type="submit" value="Submit">
</form>
<hr>
<a href="/index.html">Home</a>
</body>
</html>
```

The HTML Tunnel

The purpose of the HTML Tunnel is to provide a way for the RPG2 Unified Interface to interact with the web browser in ways that are not immediately available through the built-in RPG2 Web Server and CGI functions (see the [RPG2 Unified Interface User Guide](#) section for additional information). The process starts when the web browser of the user issues a request for a web page containing the HTML Tunnel SSI directive. The HTML content within that page are streamed to the web browser of the user in the normal fashion, and then, when the RPG2 Web Server encounters the SSI directive, it calls the registered callback handler. The callback handler prepares the response and sends the response back to the RPG2 Web Server with the HTML Tunnel data. This response is assumed as valid HTML data and is inserted into the HTML stream by the RPG2 Web Server. Inserting data into the HTML stream can be thought of as replacing the SSI directive contained on the original HTML page with content provided by RPG2 Unified Interface (see the [RPG2 Unified Interface User Guide](#) section for additional information). After sending the HTML Tunnel content data, the RPG2 Web Server transmits the remainder of the original page content to complete the presentation of the web page originally requested.

The overall sequence of operations looks like the following:

- ▶ User sets up an HTML Tunnel handler function in the RPG2 Unified Interface (see the [RPG2 Unified Interface User Guide](#) section for additional information) by using **CI_SetHttpTunnelHandler(CI_httpTunnelHandler_tp handler_p)**.
- ▶ The web browser of the user requests a web page containing the HTML Tunnel SSI directive.
- ▶ The RPG2 Web Server detects the directive on the requested page and calls the registered tunnel handler.
- ▶ The handler function prepares the valid HTML Tunnel response data in the following steps:
 - ▶ Allocates a buffer to hold the response data.
 - ▶ Copies the response tunnel data into the allocated buffer.
- ▶ The RPG2 Web Server transmits the HTML response data to the web browser and then frees the buffer containing the response data, request data.

When the **HTML_Tunnel.cgi** function executes, it looks to see if any data was posted to the RPG2 Web Server (that is, the user hit a submit button). If so, this function retrieves this data and sends it (along with the URL that contains the **HTML_Tunnel.cgi** function) to the host. If not, the function sends the URL of the web page to the host. Either way, after the data is sent to the host, the function expects to get HTML data back from the host. When the data is provided, the function simply streams it out as a part of the HTML content. The HTML for such a page follows. To use this example HTML with the example homepage, it is recommended to name the file **tunnel.stm**.

```
<html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso8859-1">
<title> RapID Platform Generation 2.0 HTML Tunnel Test Page</title>
</head>
<body>
<hr> <h1><b> RapID Platform Generation 2.0 Form To Demonstrate the HTML Tunnel</b></h1> <hr>
The Host's content should appear below:
<hr> <!-- ----- The host will provide the content that goes here ----- -->
<!--#exec="/HTML_Tunnel.cgi"-->
<hr>
<a href="/index.html">Home</a>
</body>
```


RPG2 WEB SERVER USER GUIDE

```
</html>
```

Requesting a System Reboot

As previously discussed in the [Reset.cgi Function](#) section, it is possible to reset the system using an SSI directive. This example shows how to do this by using a JavaScript to ask for confirmation and then to trigger the `/reset.cgi` function. There are a few web page files required to do this. The first page is a standard HTML file that has the JavaScript, a button to trigger it, and no SSI directives, while the second page is the one that does the work by executing the CGI function. In addition, the first page is made directly available to the user (via an HTML link or other means), but the second page is not. This way, a system reset only occurs when the user requests the visible page, presses the reset button, runs the JavaScript, and confirms the request.

The HTML for such a page follows. To use this example HTML with the example homepage, it is recommended to name this file **reset.html**.

```
<html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso8859-1">
<title> RapID Platform Generation 2.0 Reset Test Page</title>
<script language="JavaScript">
<!-- Begin
function confirmation() {
var answer = confirm("Are you sure you want to RESET the system?")
if (answer)
window.location = "reset.stm";
}
// End -->
</script>
</head>
<body>
<hr>
<h1><b> RapID Platform Generation 2.0 Form To Test The Reset CGI Function</b></h1>
<hr>
<b>Press this button to request a reset:</b>
<form>
<input type="button" onclick="confirmation()" value="Reset the System" />
</form>
<hr>
<a href="/index.html">Home</a>
</body>
</html>
```

The HTML for the second page follows:

```
<html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso8859-1">
<title> RapID Platform Generation 2.0 Point Access Syntax Form</title>
</head>
<body>
<b><h2>System Reset In Progress</h2></b>
<br>
<br>
<hr>
Please wait a few seconds and then press the link below.
```

RPG2 WEB SERVER USER GUIDE

```

<hr>
<br>
<br>
<a href="index.html">[Home]</a>
<hr>
<!-- ----- This is what requests the system to reset ----- -->
<!--#exec="/reset.cgi"-->
</body>
</html>

```

Of course, any RPG2 Web Server content is incomplete without a home page. Therefore, a simple HTML page that can be used with the other examples provided follows. It is recommended to name this file **index.html**.

```

<html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso8859-1">
<title> RapID Platform Generation 2.0 Check Box Form</title>
</head>
<body>
<hr>
<h1><b> RapID Platform Generation 2.0 WEB Server Demonstration Pages</b></h1>
<hr>
<a href="/display.stm">Display Only Page</a><br>
<a href="/text.stm">Text Box Page</a><br>
<a href="/ radioform.stm">Radio Button Page</a><br>
<a href="/ chkboxform.stm">Check Box Page</a><br>
<a href="/ dropform.stm">Drop Down Menu Page</a><br>
<a href="/ admin-passwords.stm">Password Page</a><br>
<a href="/tunnel.stm">Tunnel Page</a><br>
<a href="/reset.html">Reset Page</a><br>
</body>
</html>

```

FIRMWARE UPGRADE

To update the firmware on the device, follow the procedures outlined in this section. Note that it is assumed that the final binary is built and available. The final binary includes io-app, network-app, bootloader, and board-config in a single binary file. The firmware update is carried out using the RPG2 Web Server on the device as follows. The network status (LED-1, NET) and module status (LED-2, MOD) indicate the firmware update status.

Performing a Secure Firmware Update to the Device

Take the following steps to perform a secure firmware update to the device:

1. Open the web page of the device with a web browser.
2. Click on the link in the left-hand column for **User Upload**.
3. Authenticate to the RPG2 Web Server with the username and password provided or set. Use the user account. See the [Password Protected Forms \(Security Form Listed\)](#) section for additional information.
4. Upload the **update.pkg** file that was created and provided by the vendor.
5. After the upload reaches 100%, click the **User Update** link. Select the **update.pkg** file that was uploaded in Step 4 from the list and click **Submit**.
6. The device then automatically reboots and performs a secure firmware update. Verify that the update is occurring by viewing the firmware update status LEDs.

RPG2 WEB SERVER USER GUIDE

Verifying the LEDs

To verify the LEDS, take the following steps:

1. The bootloader indicates which step of the authenticating and booting application process it is in with the NET and MOD LEDs.
2. After the bootloader finishes booting, it turns all LEDs off.

A chart of the LED behavior can be found in the [RPG2 Hardware Design Integration Guide](#) section.

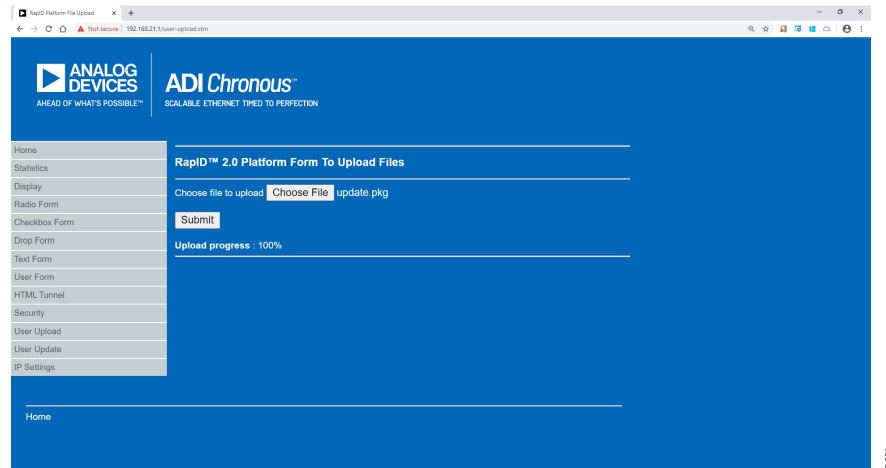


Figure 174. RPG2 Web Server Upload Page

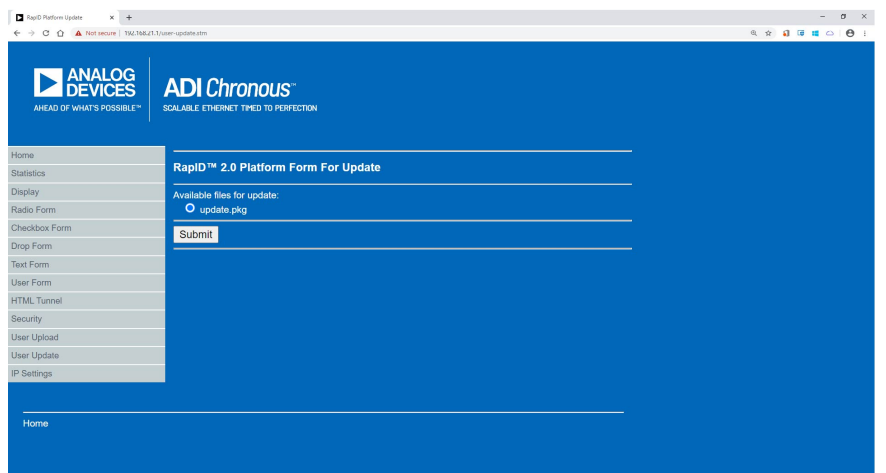


Figure 175. RPG2 Web Server Upload Submit Button

MBEDTLS

The RPG2 Web Server uses mbedTLS Version 2.5.1 for TLS communication. For mbedTLS declaration and definitions, refer to its source code, which can be downloaded from the ARM Limited, arm MBED website.

LOADING RPG2 WEB SERVER FILES

Application software is agnostic to the HTML pages loaded on to the system. The following procedure describes the high level steps to follow to load the device with the required HTML pages. The Analog Devices, Inc., software package (EtherNet/IP, Profinet, or EtherCAT) has example HTML pages for loading. These webpages can be replaced with customer specific HTML pages. This user guide also details how to create webpages with Analog Devices supported features throughout. This process can only be done if a user has access to the IAR Embedded Workbench for ARM, which can be obtained via the IAR Systems website. A user with a different applications processor must load up files onto the RPG2 file system by using JTAG or the API. See the [RPG2 Programming User Guide](#) section and the [RPG2 Unified Interface User Guide](#) section, respectively, for more information on how to do this.

RPG2 WEB SERVER USER GUIDE

The RPG2 solution supports the LittleFS file system. The device specific files must be copied to the LittleFS file system. The folder structure is depicted in [Figure 176](#).

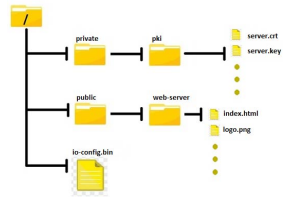


Figure 176. Web Server Content Structure Block Diagram

All RPG2 Web Server pages must be copied to **/public/web-server**. The **/private** folder typically includes the certificates and keys for secure connections for the different protocols. However, the **/private/pk1** folder includes the RPG2 Web Server specific certificate and key files.

Load Files to the Device

Take the following steps to load the files to the device:

1. Load or flash the application specific HTML pages to the flash file system of the device while the RPG2 Web Server fetches these pages when accessing the pages from the client (the web browser).

This loading procedure is carried out by using the **File Loader** application tool provided by Analog Devices, which is provided in the [software download packages](#) (see Step 2 for additional information on this application tool). This procedure is a one-time process where the HTML pages load. This procedure is not needed unless the product HTML pages must be updated.

2. Load files by using the **File Loader** application tool as follows:
 - a. Find the **File Loader** application file (**fs-blobber.exe**) in the **file-loader-app** project folder in the Analog Devices provided sample RPG2 Web Server content folder (see the [Example RPG2 Web Server Content](#) section for additional information).
 - b. Open the directory containing the **fs-blobber.exe** file in the **Command Prompt**.
 - c. Enter the command that follows:
fs-blobber.exe -i <input file path> (Note that the <input file path> must point to the folder path on the Windows® PC that consists of the files that must be loaded to file system.)
 - d. The generated blob file stores in the default directory location (**\$PROJ_DIR\$../toLoad/fs_blob.bin**).
 - e. Copy the **file-loader-app** file into the same folder where the input and output application was built.
 - f. Open the **File Loader** app project file (**file-loader-app.eww**) found in the **file-loader-appprojectiar** folder and build it.
 - g. When the build is successful, run the project.
 - h. Once loading of the files completes, the module status LED (LED-2, MOD) blinks green.
3. Stop and close the IAR project.

NOTES

**ESD Caution**

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

Legal Terms and Conditions

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use, nor for any infringements of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices. Trademarks and registered trademarks are the property of their respective owners. Information contained within this document is subject to change without notice. Software or hardware provided by Analog Devices may not be disassembled, decompiled or reverse engineered. Analog Devices' standard terms and conditions for products purchased from Analog Devices can be found at: http://www.analog.com/en/content/analog_devices_terms_and_conditions/fca.html

