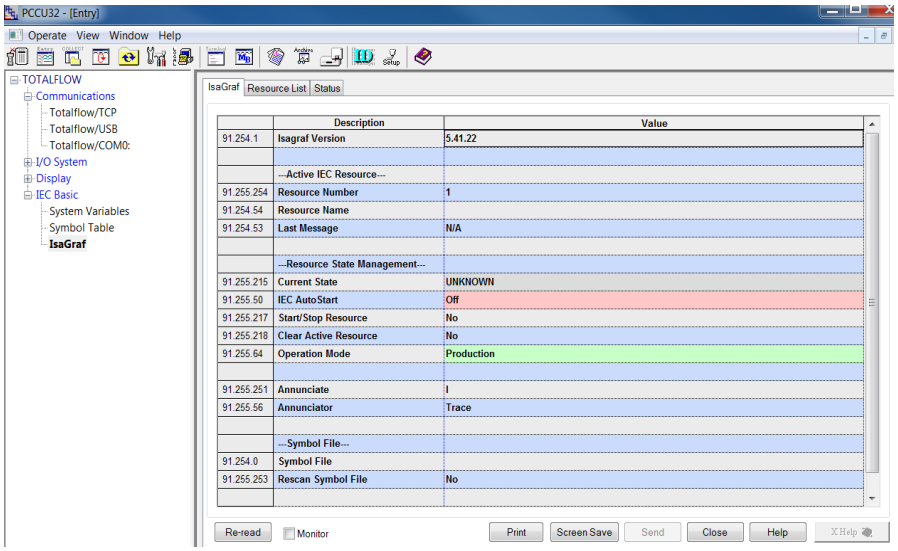
ABB MEASUREMENT & ANALYTICS

IEC 61131 Developer's Guide

ABB Totalflow G5 devices



Supporting IEC 61131 applications for custom automation and control programming

Measurement made easy

Table of Contents

List of figures

List of tables

Additional information

Table 1: Related Documentation

Documents	Document number
IEC Installer Wrapper Guide for ISaGRAF® IDE	2105856
IEC INI programmer's guide	2105858

Cyber security

The Digital Oilfield application integrates Totalflow products which are designed to be connected, and communicate information and data, via a network interface. All Totalflow products should be connected to a secure network. It is the customer's sole responsibility to provide, and continuously ensure, a secure connection between the product(s) and the customer network as well as a secured and controlled physical access to the hardware equipment, or any other network (as the case may be). The customer shall establish and maintain appropriate measures (such as, but not limited to, the installation of firewalls, the application of authentication measures, encryption of data, installation of antivirus programs, etc.) to protect the products, the network, its system and its interfaces against any kind of security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information. ABB Inc. and its affiliates are not liable for damages and/or losses related to security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information.

Although ABB provides functionality testing on the products and updates it releases, the customer should institute its own testing program for any product updates or other major system updates (to include, but not limited to, code changes, configuration file changes, third party software updates or patches, hardware change-out, etc.) to ensure that the security measures the customer has implemented have not been compromised and that the system functions in the customer's environment as expected.

Safety

Observe warning signs on packaging and on the devices, prior to software configuration.

Safety symbol conventions

"NOTICE" hazards are associated with equipment or property damage, it must be understood that under certain operating conditions, operating damaged equipment can result in degraded system / process performance leading to serious or life-threatening injuries. Therefore, compliance with all "NOTICE" hazards is required at all times.

NOTICE – Equipment damage or loss of data. This symbol indicates a potential for equipment damage, loss of data or other unintended outcome. Failure to observe this information may result in damage to or destruction of the product and / or other system components.

IMPORTANT NOTE: This symbol indicates operator tips, particularly useful information, or important information about the product or its further uses.

Potential safety hazards

IEC 61131 applications execute and interface with other applications in ABB Totalflow products. IEC application developers must be advanced users and be familiar with the operation of the product where they plan to install the applications.

Avoid disruption of commissioned devices. Carefully conduct and plan installation, activation and verification of the IEC application.

Review and follow all health and safety recommendations for the device described in the user manuals or startup guides when installing and running an IEC application during a device's field installation.

1 Description

This guide describes the development of IEC 6113 applications, installation, and activation in the ABB Totalflow G5 devices.

1.1 IEC 61131 General specifications

The IEC 61131 standard specification, developed by the International Electrotechnical Commission (IEC), provides a generic programming environment for the PLC industry. In conjunction with ISaGRAF®, ABB Totalflow has adapted their IEC compiler to work with the ABB Totalflow product line. This alliance has greatly enhanced the functionality of the products by providing the tools with which to build custom applications.

Different user applications may require different programming languages. One application may lend itself to a graphical language (Function Block Diagram) while another program may be best addressed using ladder logic (Ladder Diagram). The IEC 61131 ISaGRAF IDE supports the following programming techniques for G5 devices:

- ST: Structured Text
- FBD: Function Block Diagram
- LD: Ladder Diagram
- SFC: Sequential Function Chart

Using one of the above programming techniques, or using them in combination, should provide enough flexibility to address any special application problems that the field environment might present.

ABB Totalflow G5 devices, flash version 2.0.0 or greater, can run up to 10 IEC 61131 applications. Each IEC 61131 application requires an application credit.

- One IEC 61131 Interface Turned On = One ISaGRAF Resource
- One IEC 61131 application credit supports one instance of the IEC 61131 interface

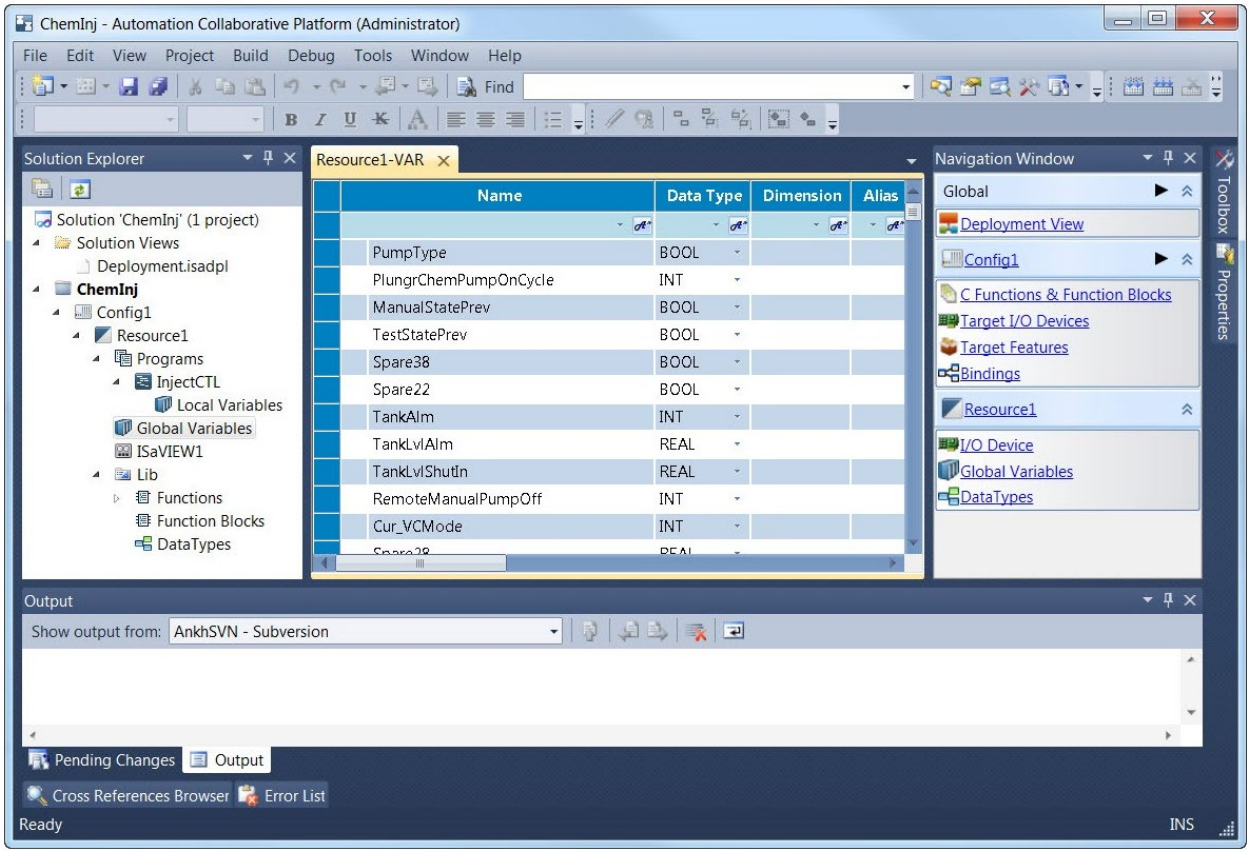
These application credits can be applied to the flow computer at the factory or by purchasing a credit key (secure flash drive) with the appropriate credits installed.

1.2 ISaGRAF integrated development environment (IDE)

IMPORTANT NOTE: The information included in this section does not discuss the ISaGRAF environment beyond the interface to ABB Totalflow products. Consult the ISaGRAF documentation for general-purpose information regarding IDE and project organization. For specific information about the ISaGRAF interface with Totalflow, use the details in this guide.

Figure 1 shows the ISaGRAF screen after opening a pre-existing project. In this view, several windows are visible. In the Solution Explorer window, the project file and its contents are organized in a hierarchical tree. A window with variable definitions and information displays in the middle of the screen. The Navigation window provides access to additional utilities.

Figure 1: ISaGRAF screen displaying pre-existing project



1.3 Activating ISaGRAF IDE

The ISaGRAF Workbench software (2104764) is required to build IEC applications for Totalflow controllers. This free software is available from order entry on a USB flash drive.

To order, see Contact us on the last page of this guide. Request 2104764 SOFTWARE, ISAGRAF WORKBENCH IEC DEVELOPMENT.

The USB flash drive includes:

- ISaGRAF Workbench 6.6
- Totalflow Packager for building IEC packages that can be loaded on RMC
- RMC and XRC target files
- Documentation for building IEC applications on RMC and XRC

For best results, verify the latest OS and flash software is loaded on the RMC or XRC.

1.3.1 Activate ISaGRAF Workbench

Registration Keys are required to activate the ISaGRAF work bench. Order Entry at ABB Totalflow will provide the registration keys. Use the following steps to find the information needed.

1. Open the License Info window: **ISaGRAF Automation Collaborative Platform.**
2. Click **Help.**
3. Click **Licensing CAM 5**. The Workbench registration window opens.
4. Call ABB Totalflow order entry (see Contact us).
5. Read the user codes from the Workbench registration window.

Figure 2: Automation Collaborative Platform

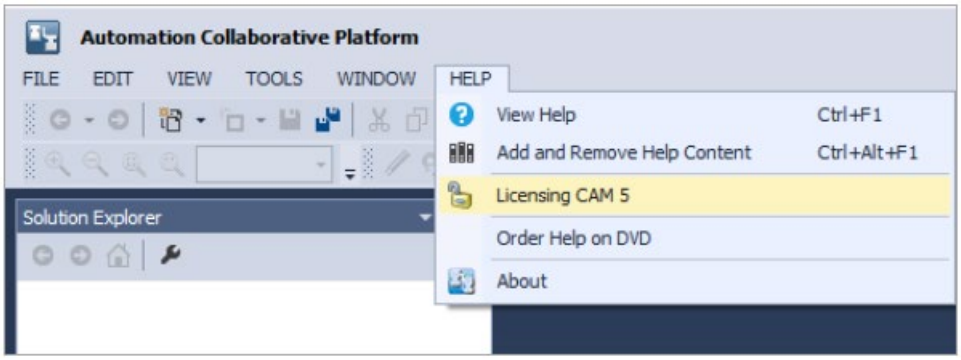


Figure 3: License Info screen

2 Develop and run an IEC 61131 application

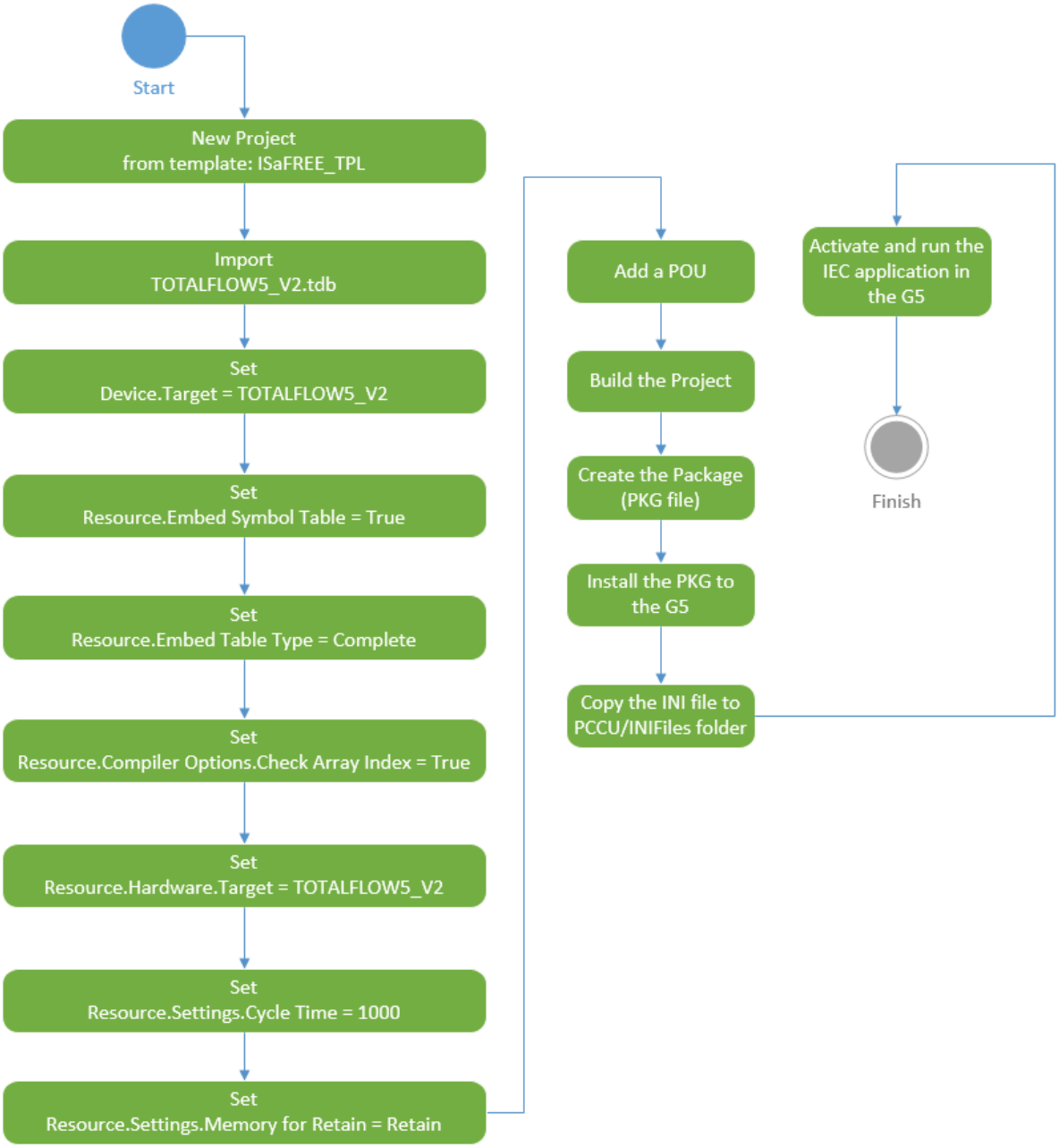
The procedures in this section describe how to create, install, and run an IEC 61131 application in an ABB Totalflow device. The workflow overview provides the end-to-end high-level view of the major tasks required to complete development and run the application.

IMPORTANT NOTE: This manual refers to the embedded Totalflow application, which supports IEC applications, as simply the "IEC interface". IEC 61131 application is shortened to the "IEC application".

2.1 Workflow overview

Figure 4 provides the workflow required to create, install, activate, and run an IEC application in an ABB Totalflow device. Review each procedure and follow the detailed instructions in each corresponding section in the order presented.

Figure 4: Workflow overview

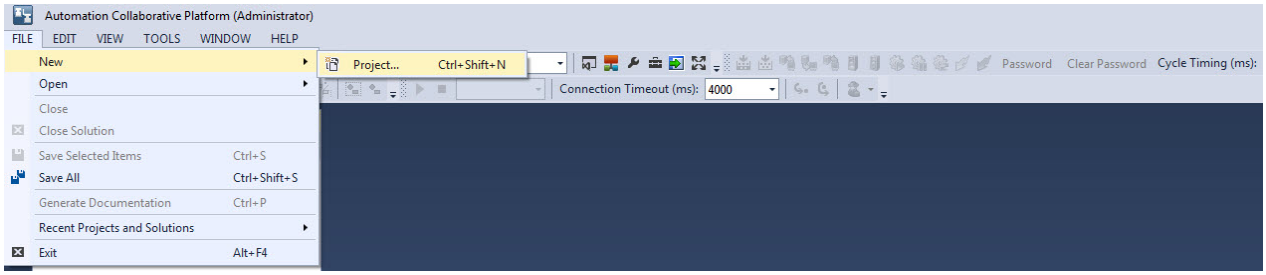


2.2 New Project from template: ISaGRAF_TPL

To create a new project using ISaGRAF:

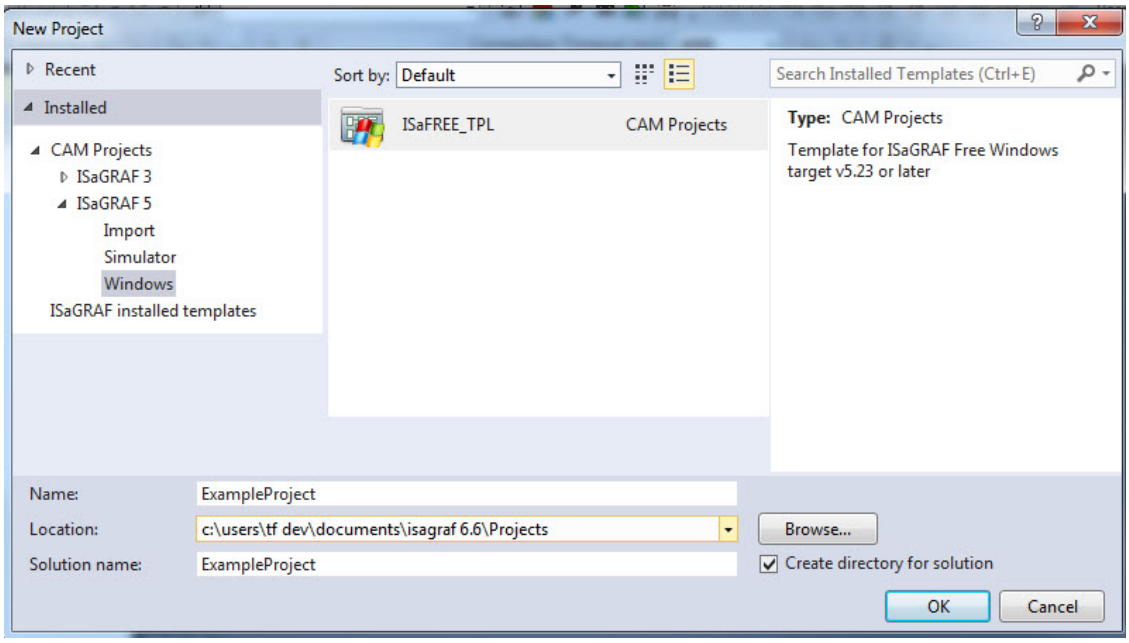
1. Open the ISaGRAF IDE.
2. Click **File**> **New**> **Project** (Figure 5).

Figure 5: Main ISaGRAF IDE screen



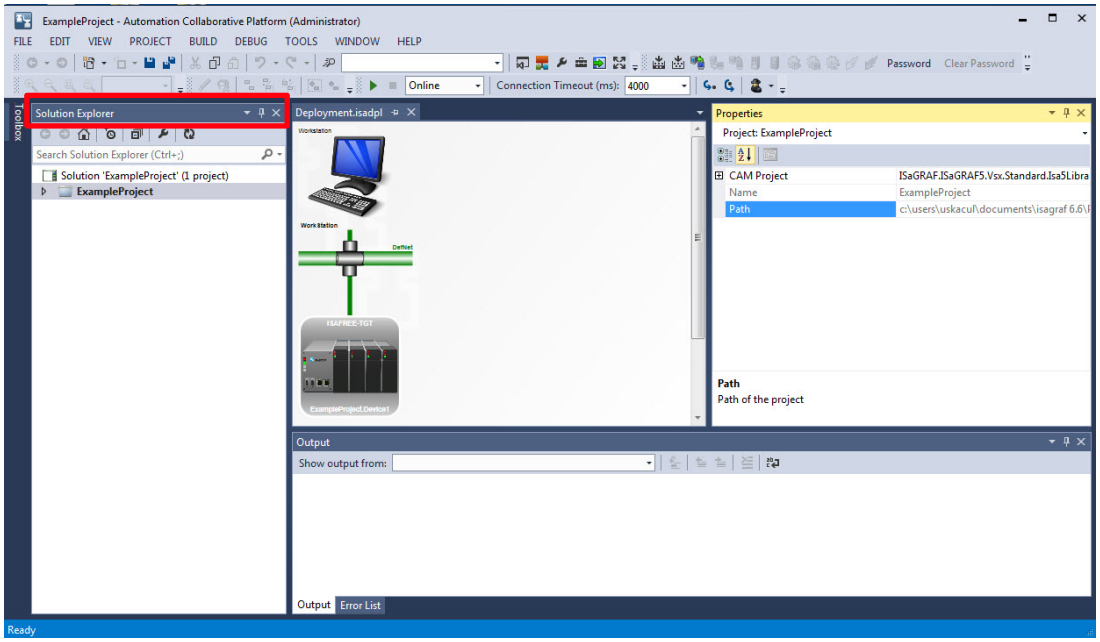
3. At the New Project dialog box (Figure 6), select the ISaFREE_TPL template from the list.
4. Type the name of the project in the Name field.
5. Click **OK** to use the default location folder to save the project file, or click **Browse** for another location folder.

Figure 6: New project dialog box



Verify that the project just created displays in the Solution Explorer window (Figure 7).

Figure 7: Solution Explorer window



2.3 Import TOTALFLOW5_V2.tdb file

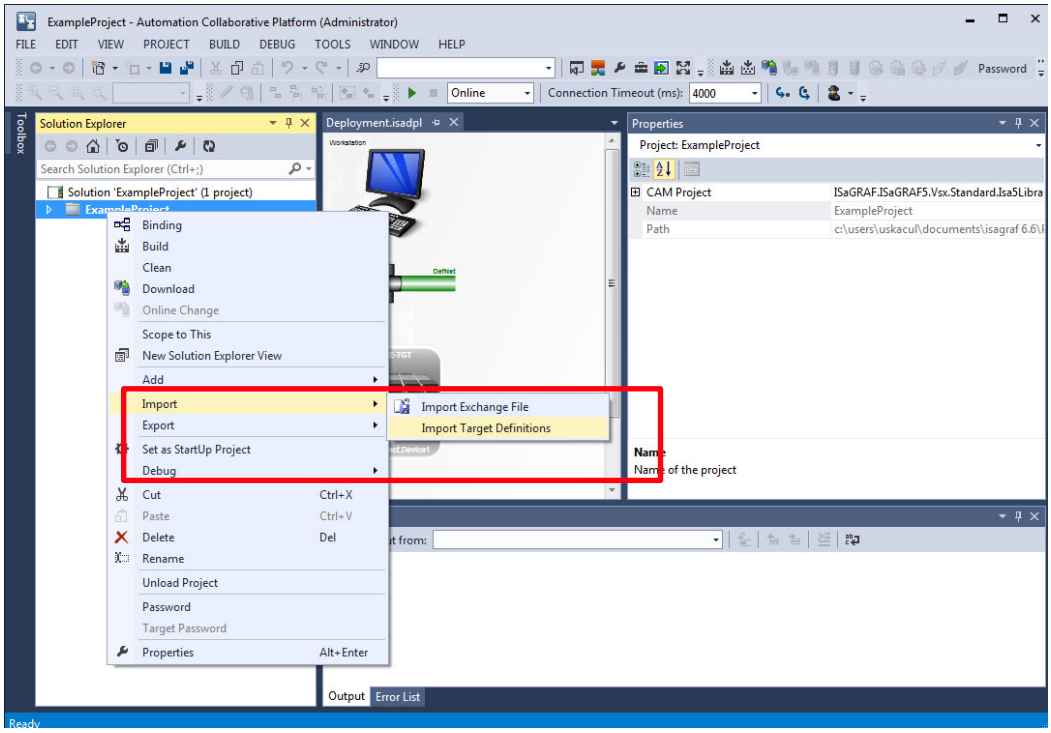
ABB Totalflow provides a definition file (TOTALFLOW5_V2.tdb) that enables the ISaGRAF environment to specifically target the ABB Totalflow G5 device.

IMPORTANT NOTE: The TOTALFLOW5_V2.tdb file provides functions to read and write to the data types available in G5 devices. The definition file also defines the embedded OS, the processor type, memory mapping and the overall target device environment. The TOTALFLOW5_V2.tdb file definition is required for compatibility with Totalflow version 2.0.0 and on G5 devices. TOTALFLOW5_V1.tdb will no longer be supported.

To import the device definition:

1. From the ISaGRAF Solution Explorer window, right-click the project that was just created.
2. From the pop-up menu, select **Import**> **Import Target Definitions** (Figure 8).
3. At the Open dialog box, select **TOTALFLOW5_V2.tdb**.
4. Click **Open**.

Figure 8: Import the ABB Totalflow device definition file

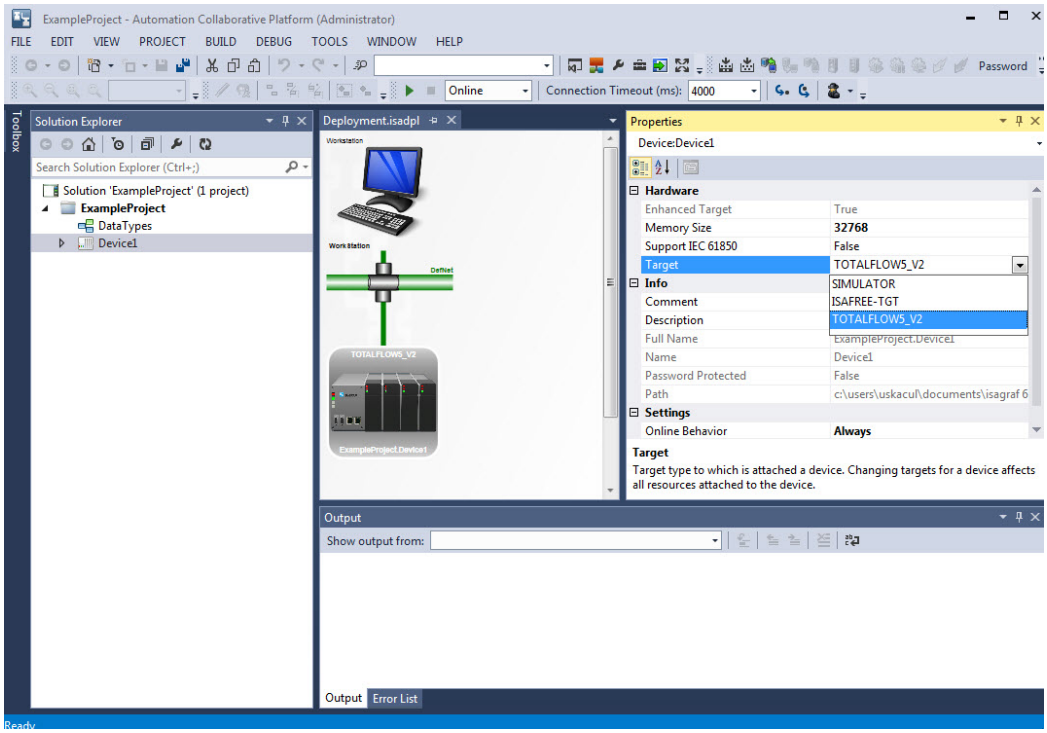


2.4 Set Device Target = TOTALFLOW5_V2

To set target priorities:

1. From the ISaGRAF IDE main top menu bar, select **View** > **Properties Window**.
2. After the TOTALFLOW5_V2.tdb is installed, expand the project in the Solution Explorer window (i.e., **ExampleProject**), and select **Device1** to view properties (Figure 9).

Figure 9: Set the device properties



3. In the Target property field, change the value to **TOTALFLOW5_V2**.
4. Click **Yes**.

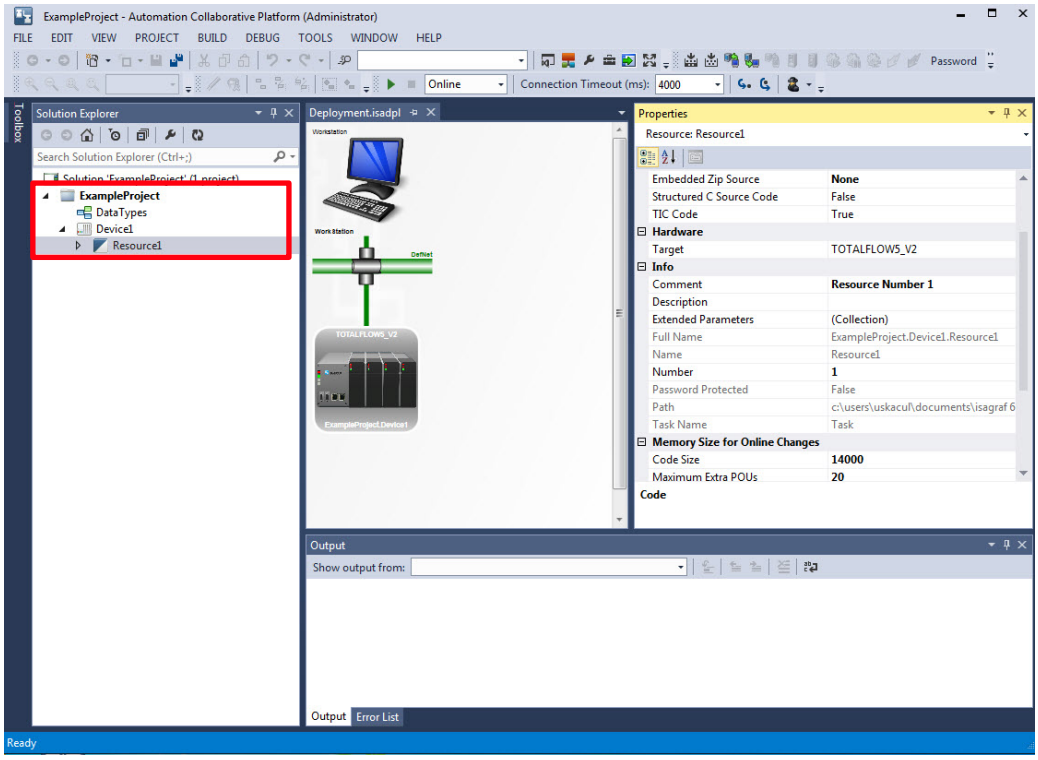
2.5 Set Resource Embed Symbol Table = True

The resource properties are set up at the Properties windows. Expand the different sections to locate specific properties as instructed in this procedure. Use the scroll bar if needed to navigate through the window to locate properties.

To set the Resource properties:

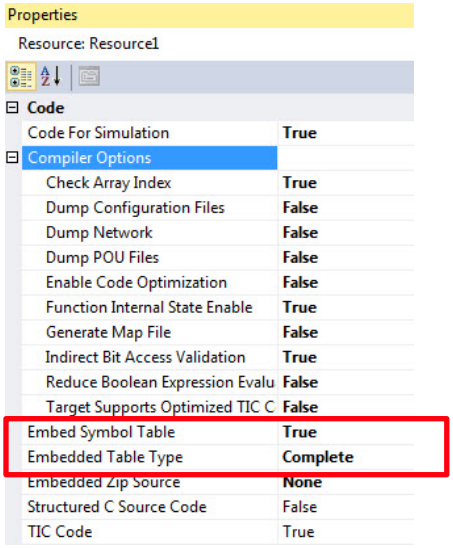
1. In the Solution Explorer window, expand **Device1**.
2. Click **Resource1** to view in the Properties Window (Figure 10).

Figure 10: View the resource properties



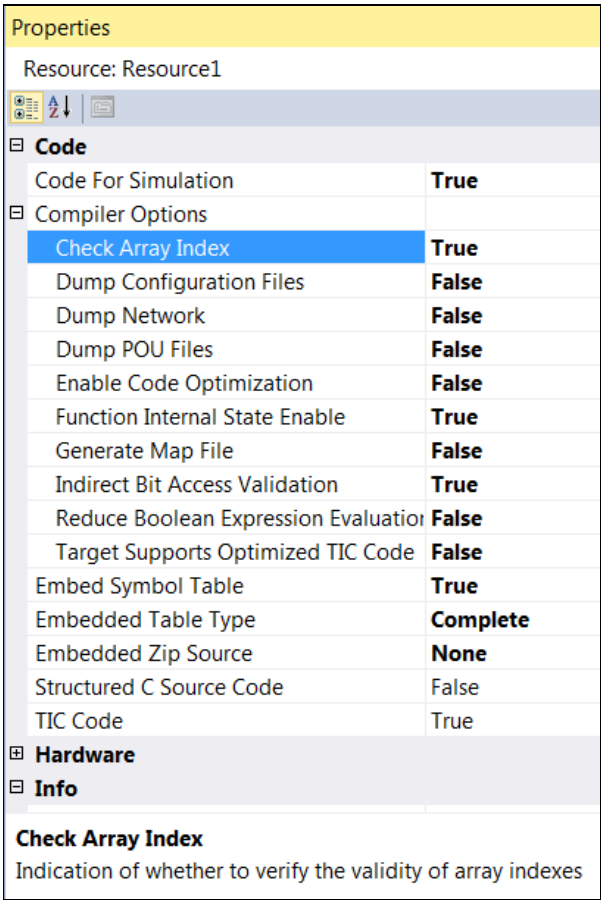
3. In the Properties window, select **Code** > **Compiler Options** (Figure 11):
   - Ensure that Embed Symbol Table property is set to **True**.
   - Ensure that Embedded Table Type property is set to **Complete**.

Figure 11: Verify the resource properties



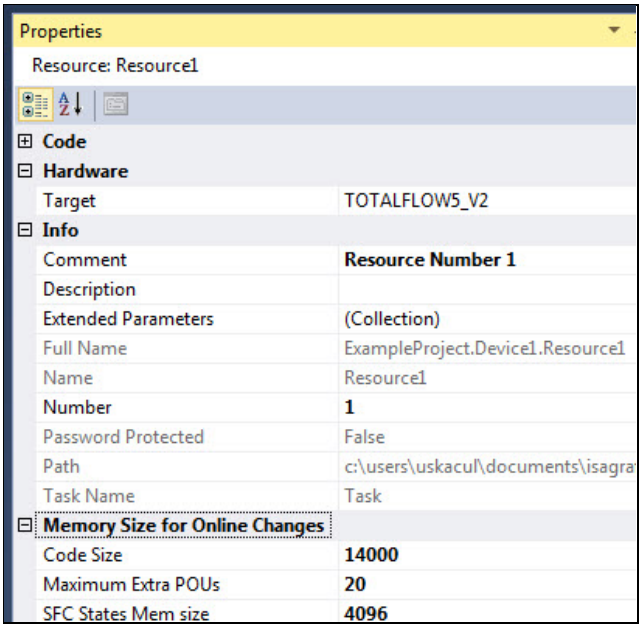
4. In the Properties window, expand **Compiler Options**, and set the Check Array Index property to **True**. See the figure below.

Figure 12: Set Resource Compiler Options Check Array Index = True



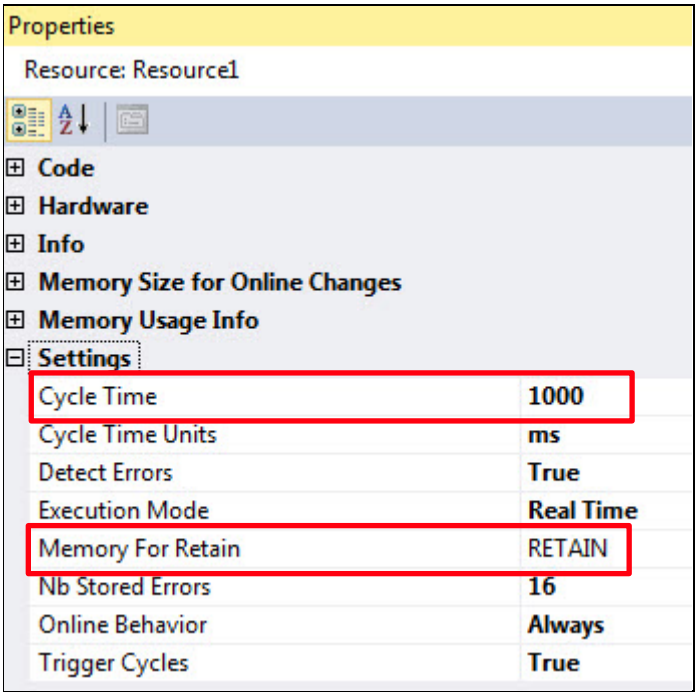
5. In the Properties window, expand **Hardware** and ensure the Target property is set to **TOTALFLOW5_V2**. See figure below.

Figure 13: Set Resource Hardware target = TOTALFLOW5_V2



6. In the Properties window, expand **Settings**. See Figure 14.
7. Change the Cycle Time property value to **1000**.
8. Ensure the Memory for Retain property is set to **RETAIN**.

Figure 14: Set Resource Settings Cycle Time = 1000



2.6 Create a Program Organizational Unit (POU) in the project

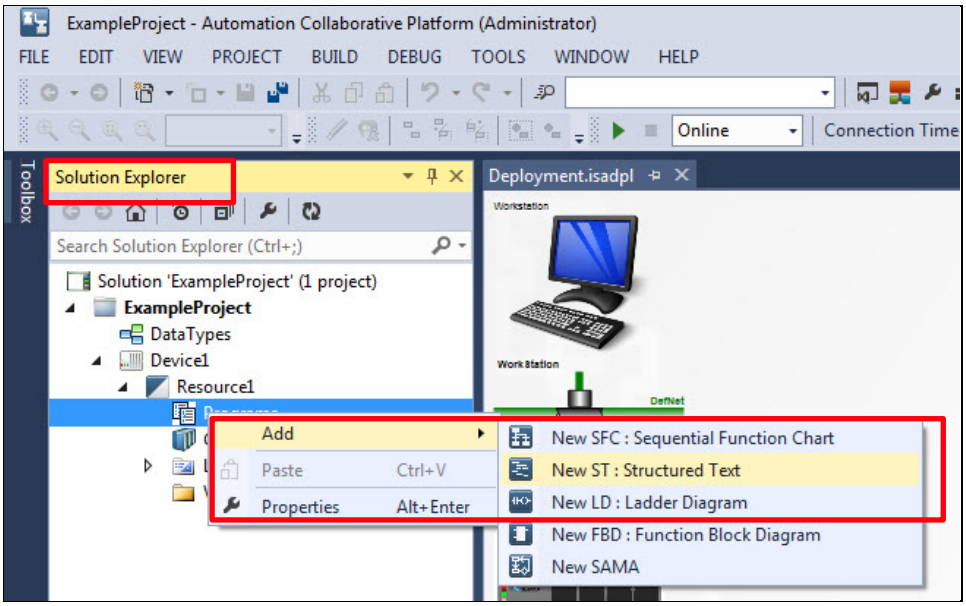
This section includes procedures to create a simple program (POU). The logic for an IEC application is contained in a POU. A POU belongs to a Resource and may use the various languages available within ISaGRAF. The example used in the following steps uses Structured Text (ST) language.

2.6.1 Add a POU

To add a POU to the Resource:

1. In the Solution Explorer window, Right-click **Resource1** and select **Add** > **New ST**: **Structured Text**. See the figure below.

Figure 15: Add Structured Text

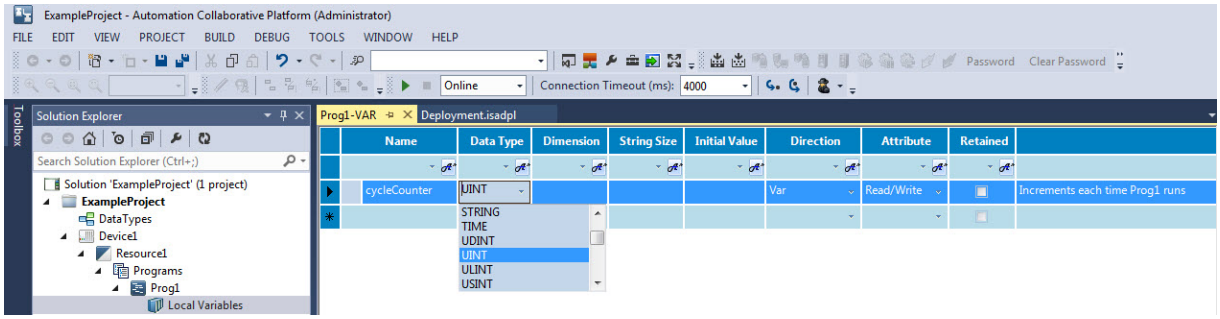


2.6.2 Add POU variables

To add local variables to the POU:

1. Expand the program or POU previously added. In the Solution Explorer window, expand **Prog1**.
2. Under Prog1, Double-click **Local Variables**. A table to define variables displays. See Figure 16.
3. Add a Local variable. For example: Name = cycleCounter, Data Type = UINT, Initial Value = 0, Direction = Var, Attribute = Read/Write, and Comment = Increments each time Prog1 runs.

Figure 16: Add POU variables

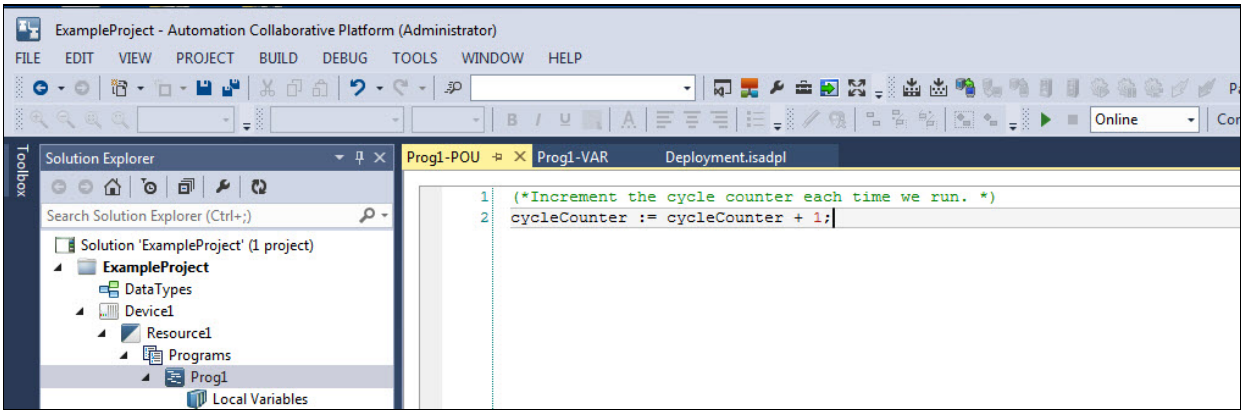


2.6.3 Add POU ST code

To add code to the POU:

1. Under Programs, double-click **Prog1**. See Figure 17.
2. Add the ST code. In this example, this code increments the cycleCounter variable.

Figure 17: Add structured text (ST) code

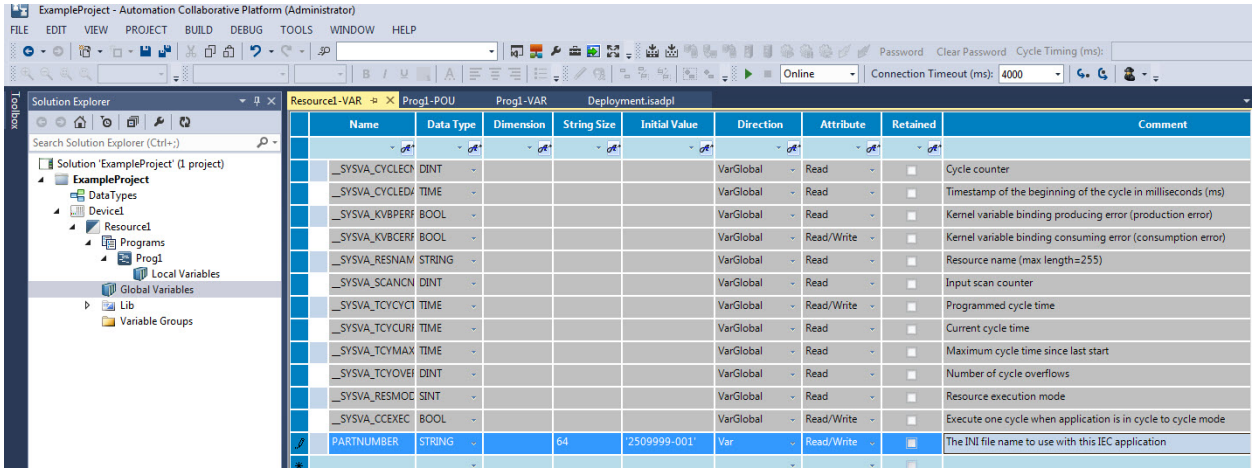


2.6.4 Add global variables

To add global variables to the Resource:

1. In the Solution Explorer window, expand **Device1** and **Resource1**.
2. Under Resource1, double-click **Global Variables**. See Figure 18.
3. Add the Global variable as shown. For example:
   - Name = PARTNUMBER
   - Data Type = STRING
   - String Size = 64
   - Initial Value = 2509999-001
   - Direction = Var
   - Attribute = Read/Write
   - Comment = the INI file name to use with this IEC application

Figure 18: Add resource global variables



4. In Solution Explorer, right-click **Resource1**. Select **Rename**, enter the name of the project, and Press **Enter**.

2.6.5 Add proprietary ABB Totalflow variables

There are two special variables that ABB Totalflow uses to help the developer:

- PARTNUMBER: Used by PCCU32 to determine the INI file name for the selected IEC program.
- NumRegisters: Set the size of the indirect address array. For details, see section 3: Interface with the IEC application.

IMPORTANT NOTE: ABB Totalflow variables are case sensitive.

2.7 Build and package the IEC application

After the application has been completed, use the following procedures to run the application on the Totalflow G5 device.

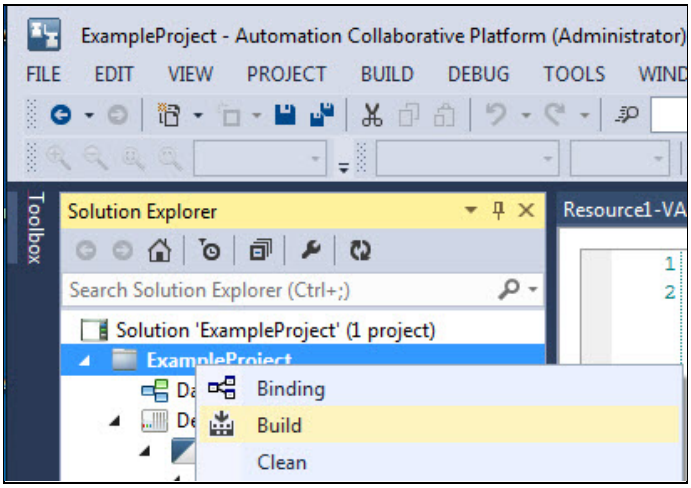
- Build the project
- Package the IEC application

2.7.1 Build the project

To build the project code:

1. Right-click on the project.
2. Click **Build** (Figure 17). The IEC code is created.

Figure 19: Build the IEC project



2.7.2 Create the package

The packager can create a package file or PKG (a file with extension .pkg) for installation in a Totalflow G5 device.

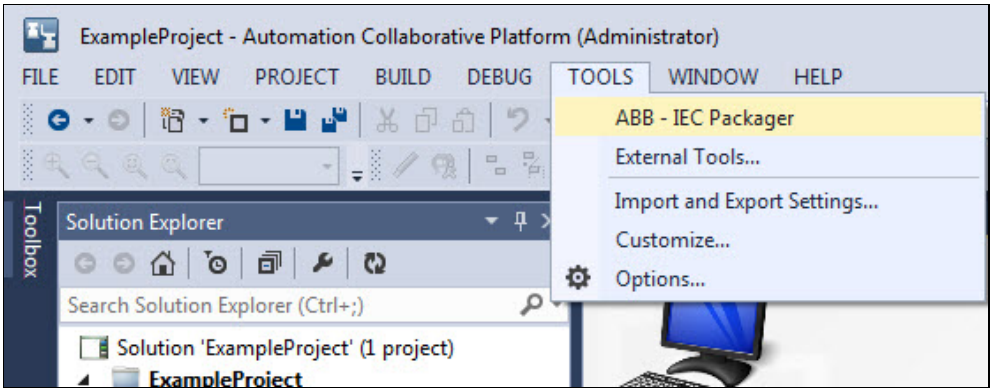
Table 2: IEC packager options

IEC Resource	Select the Resource for packaging from the IEC Resource dropdown menu if the IEC project contains multiple Resources.
IEC Package Name	The name of the package file to create. This is the name that displays in PCCU under Available IEC Resources after installation of the IEC application. The default is the name of the resource within the ISaGRAF IDE.
Description Tag	The description displays in the Totalflow device loader when installing the IEC application (PKG file). The default is the comment property of the resource within the ISaGRAF IDE.
INI Name	The name of the INI file to generate. The default is the value of the PARTNUMBER string variable within the ISaGRAF IDE. **IMPORTANT NOTE:** The PARTNUMBER is the name of the INI that PCCU will load when this IEC application is running in the G5 devices. The PARTNUMBER is automatically retrieved from the Global Variables.
Create PKG File	Option to create a package file for installing to the G5 device
Create INI File	Option to create an INI file for interfacing to the IEC application within PCCU
Show PKG/INI file in file explorer	When checked, file explorer will open to show the new PKG and INI file.

To package the IEC application:

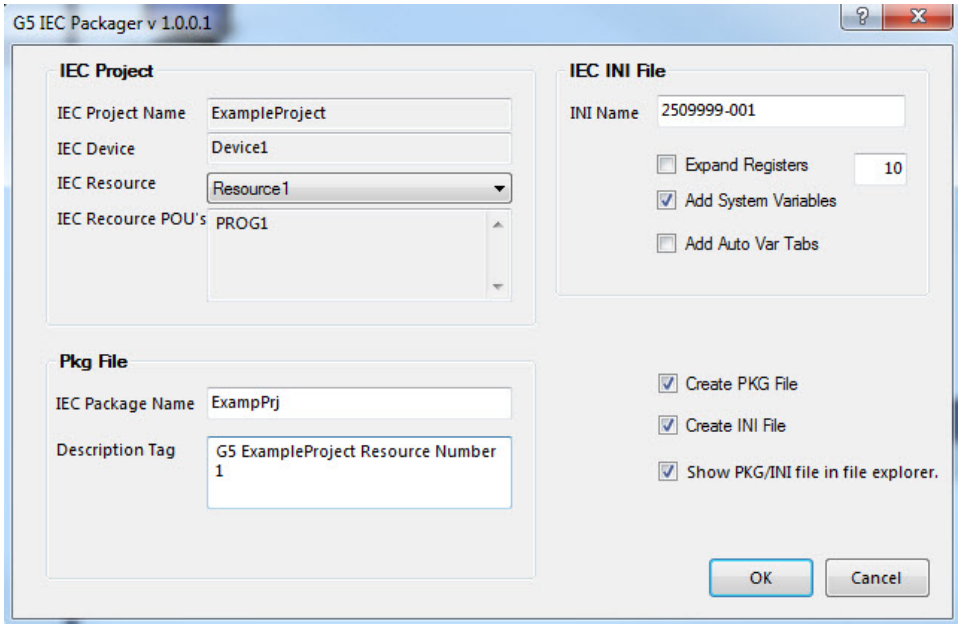
1. Select the project in Solution Explorer.
2. Select **Tools** on the top main menu (Figure 20).
3. Select **ABB - IEC Packager**.

Figure 20: Packaging the IEC project



4. In the G5 IEC Packager window, select **Options**. See the figure below.

Figure 21: G5 IEC Packager options



5. Click **OK** to create the package and INI file.

File explorer opens to show the location of the new files. Take note of the location of the files. This is required to complete the installation.

2.7.3 Instantiate the IEC interface from PCCU32

The IEC Interface is the ABB Totalflow application required to support and run IEC applications. The IEC Interface is the link between the ISaGRAF application and the embedded Totalflow software in the G5 device.

To instantiate the IEC interface:

1. Connect to the target device using PCCU.
2. In Entry mode, locate and select the meter ID at the top of the PCCU32 tree view.
3. Click the **Application/License Management** tab.

Under **Device Credits**, verify that there is at least one unused basic IEC credit available per instance. Any available credits display under the Surplus/Deficit column (highlighted in green).

4. If there are no basic IEC credits available, insert a credit key and add the required credits. Click **Help** for more information.

IMPORTANT NOTE: The IEC Interface application requires a basic IEC credit. To add a basic IEC credit, refer to the Application and License Management help topics in PCCU.

5. After the credit is available, click **Add**.
6. Click the **Application to add** drop-down list.
7. Select the **IEC Interface**. A default application number is automatically configured.
8. Click **OK**. In the application table, verify that the IEC Interface application is displayed.

IMPORTANT NOTE: If there are no credits available, the Basic IEC application may still show in the list, but it will not be possible to enable the application.

9. Click **Send**.

Verify that the IEC Interface application displays in the table and in the PCCU tree view.

2.8 Install the PKG and copy the INI file

After building the code and packaging the IEC application files, install the application in the target device and copy the INI file to the PCCU directory. The procedures in this section describe how to install the IEC application package and its INI file. The IEC application must be installed in the device to be activated, and the INI file must be in the PCCU directory to be able to display the custom screens designed for the application.

IMPORTANT NOTE: For more details, see the help topics for IEC 61131 in PCCU.

2.8.1 Copy the INI file to PCCU/INIFiles folder

IMPORTANT NOTE: During development, it may be advantageous to use the built-in automatic INI until most variables are defined.

To copy the INI file:

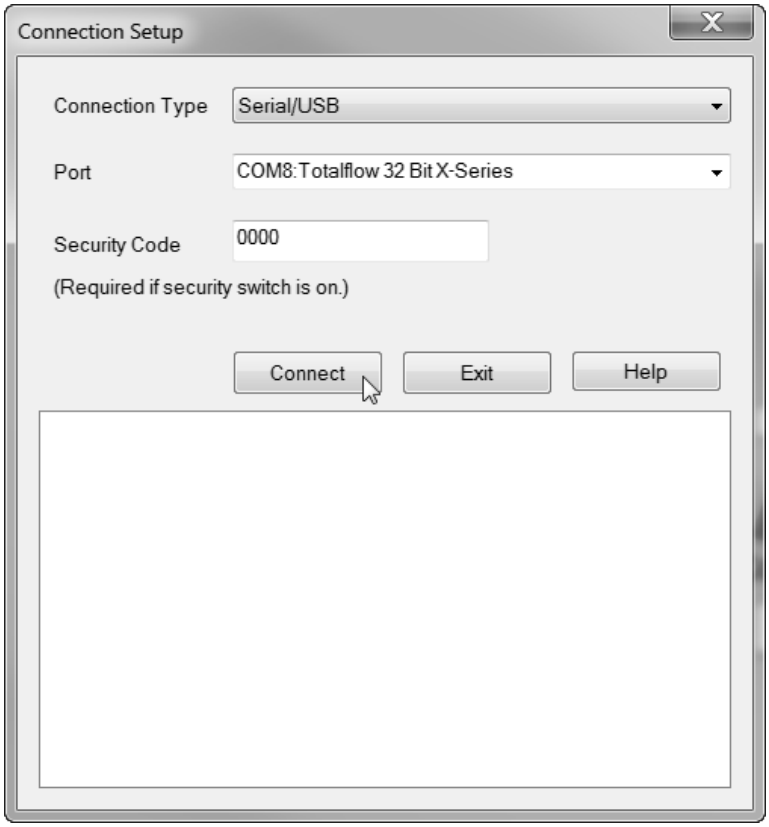
1. Locate the INI file created. The default location is in the ISaGRAF installation directory shown by the G5 IEC Packager, as described in section 2.7.2. Create the package
2. Copy the INI file to the IniFiles folder in the PCCU installation directory.

2.8.2 Install the IEC application in the target device

IMPORTANT NOTE: Use the following procedure for G5 devices and the RMC. For more information about the device loader, see the G5 Device Loader help files.

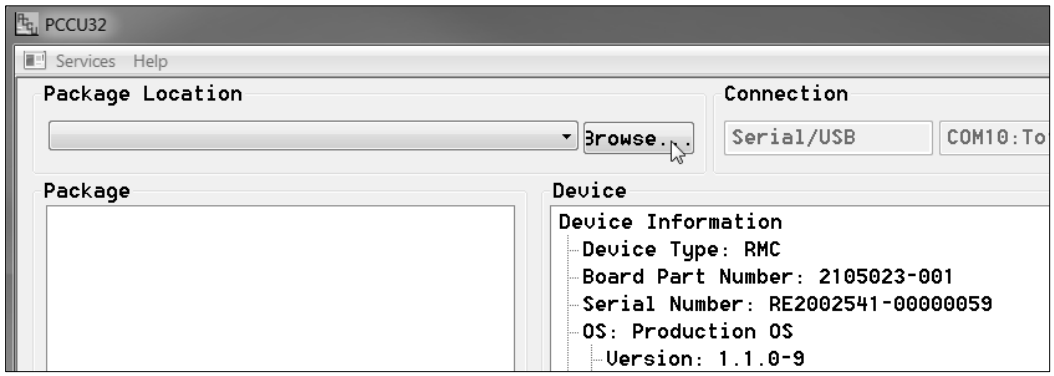
1. Start PCCU.
2. On the top tool bar, select the **32-bit Loader** icon.
3. If you have backed up the device data and configuration, click **OK**.
4. At the Connection Setup window, click **Connect**. See the figure below.

Figure 22: Connection Setup for the device loader



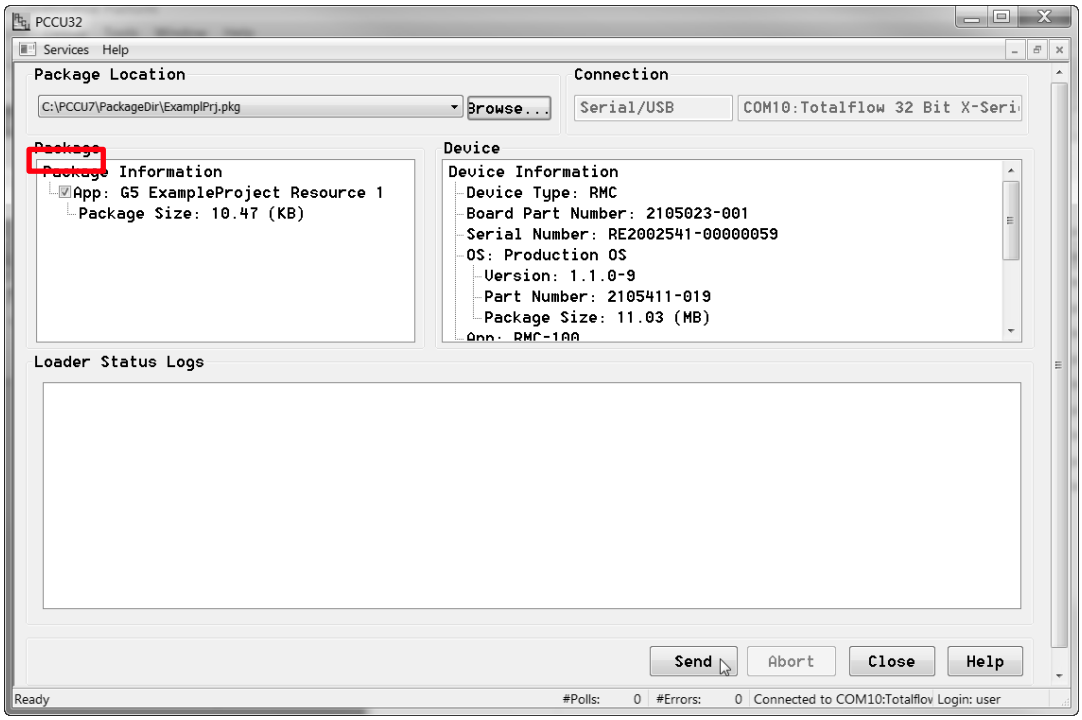
5. On the device loader screen (Figure 23), under Package Location, click **Browse** to locate the IEC application package file. It is shown by the G5 IEC Packager (see section 2.7.2 Create the package).

Figure 23: G5 Device loader



6. From the browser window, select the file, and click **Open**. The example below uses ExamplePrj. Pkg.
7. As shown in Figure 24, verify that the file is listed in Package View and is classified as an App package. Select the **App checkbox.**

Figure 24: IEC application package selected in the G5 device loader



8. Click **Send**.

Monitor the Log view in the loader screen to verify that the file is sent successfully to the target device. After the package file transfer is complete, the Loader Status Logs shows the message "Device info updated".

9. Click **Close** to exit the device loader. Activate the application as described in the following section.

2.9 Activate and run the IEC application in the G5 device

After the application is installed in the device, the application must be activated to run or execute. The procedure included in this section describes how to activate and run the IEC application. For the example application used:

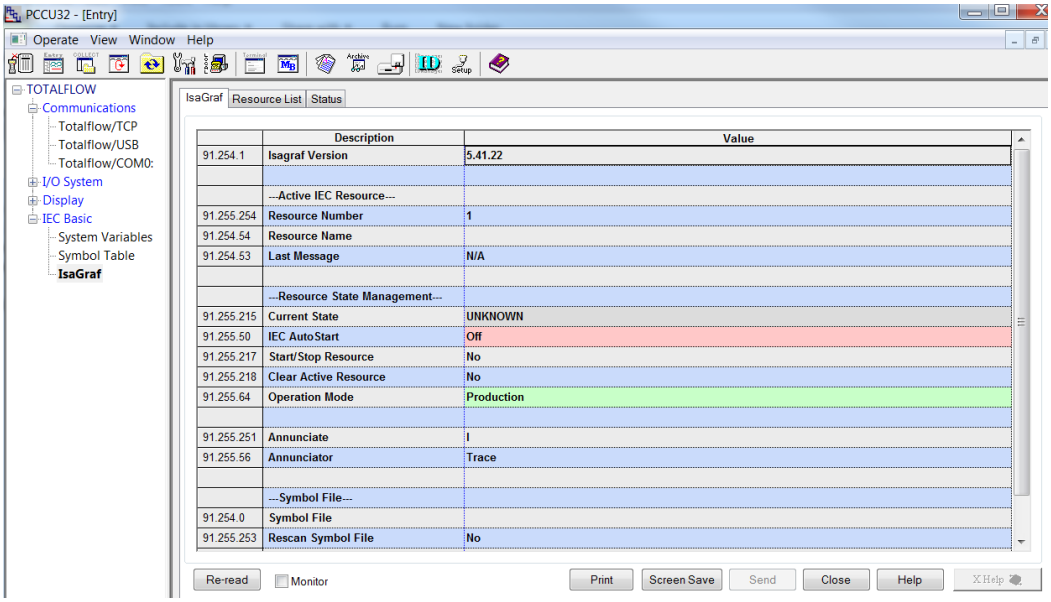
Change the ISaGRAF Active Resource to **Activate & Run** for the ExamplePrj resource.

View the cycleCounter variable in PCCU to verify that the ExamplePrj IEC application is running.

To activate and run the IEC application:

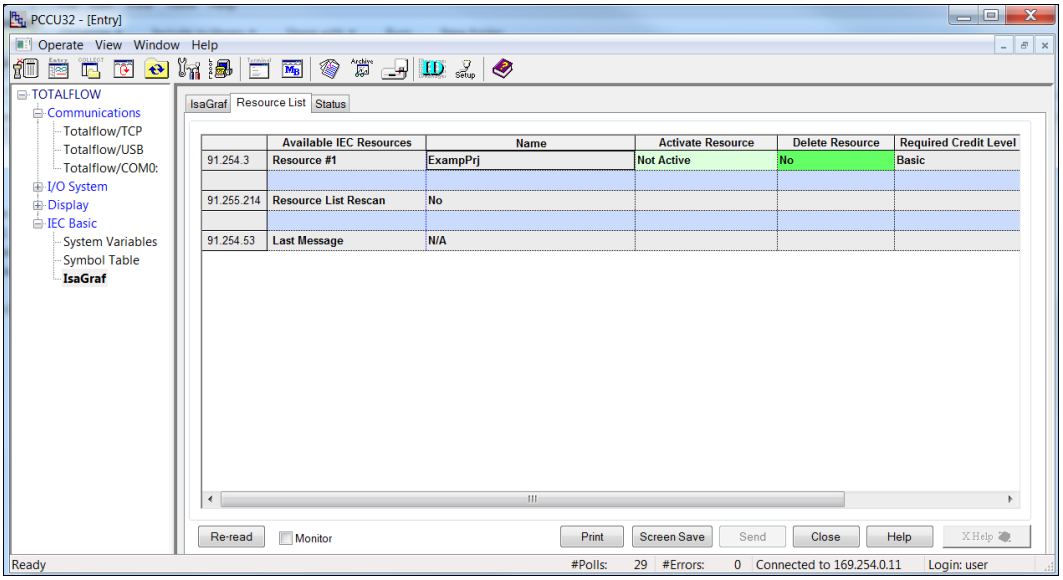
1. Connect to the device using PCCU and select **Entry** mode.
2. On the tree view, expand **IEC Basic**.
3. Under IEC Basic, select **ISaGRAF.**

Figure 25: ISaGRAF tab



4. Select the **Resource List** tab to display the installed application.

Figure 26: ISaGRAF Resource List tab



5. On the Resource List tab, select the **Activate Resource** drop-down menu (see Figure 27 and Figure 28).
6. Click **Select & Run**.
7. Click **Send**.

Figure 27: Activate Resource drop-down menu

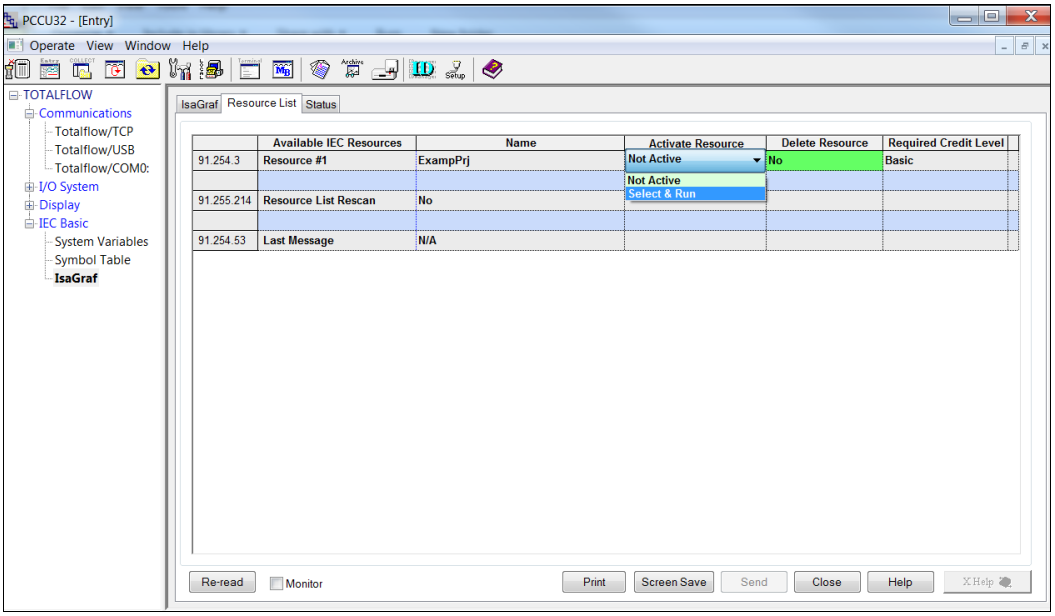
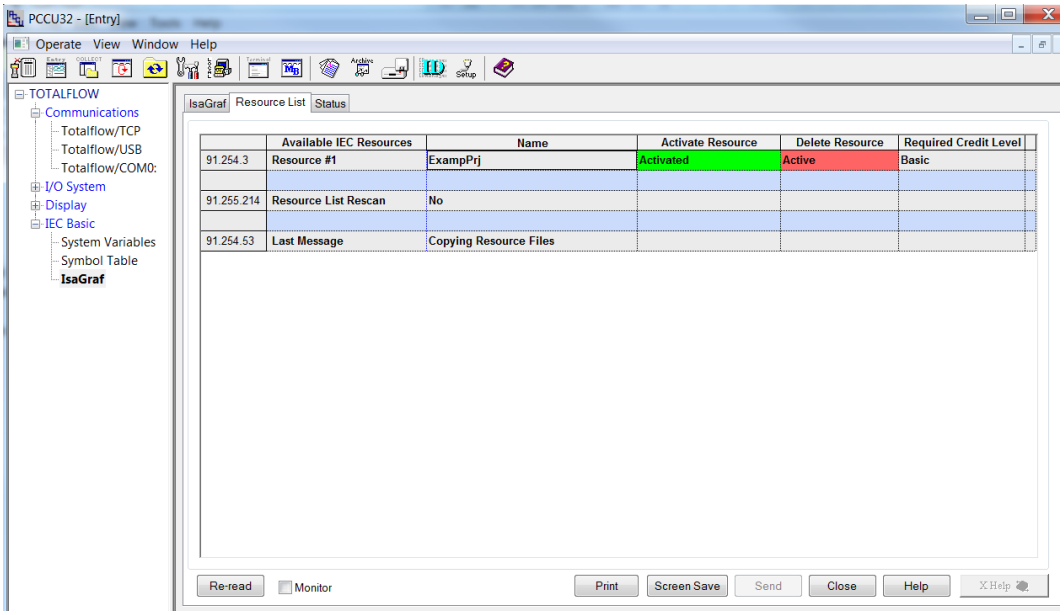
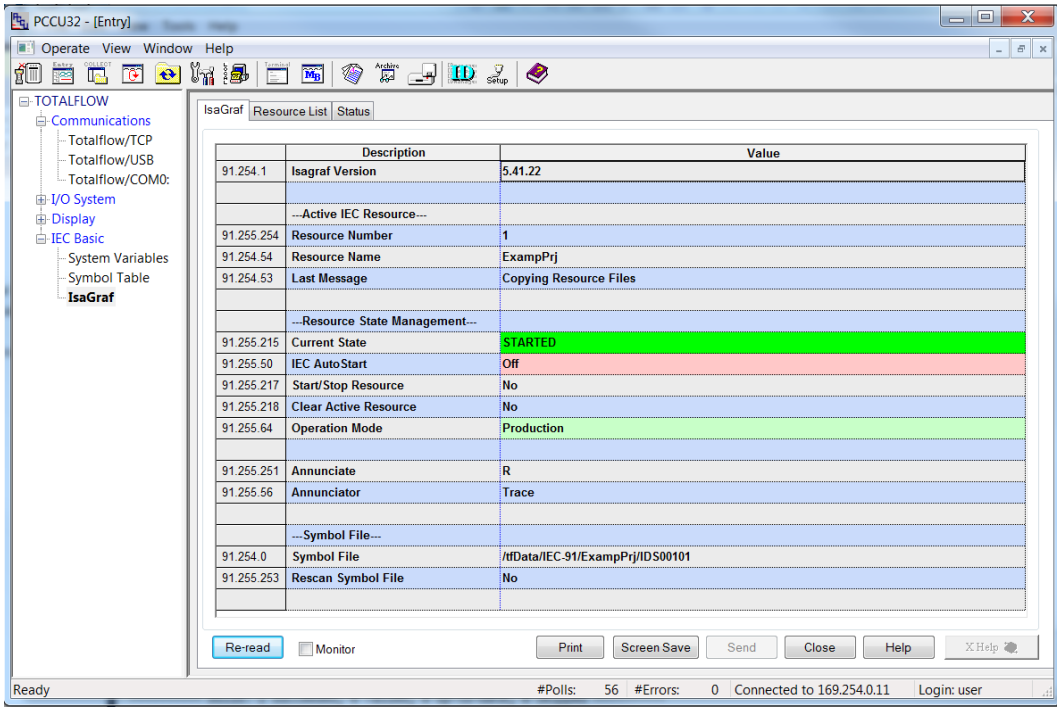


Figure 28: Activated Resource



After activation, return to the **ISaGRAF** tab. Under the Description Column, Current State (app 255.215), verify that the resource has started. To automatically start the IEC resource when the TOTALFLOW controller boots, set the IEC Auto Start = **ON** (see figure below). For more information, click the context-sensitive **Help**.

Figure 29: Activated ISaGRAF tab



IMPORTANT NOTE: For additional instructions to manage the IEC resource, click **Help** from the **ISaGRAF tab** screen. Or search for the **IEC 61131 Applications** topic within PCCU help.

2.9.1 Verify that the IEC application is running

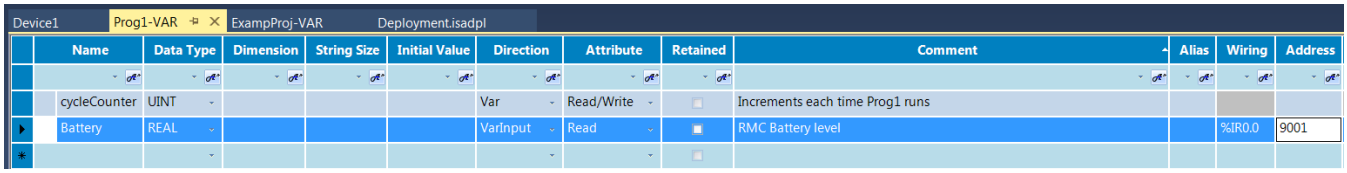
To verify that the IEC is running:

1. Select **IEC Basic** in the tree view. The defined tabs reflecting the IEC application custom INI should display (Figure 30).

IMPORTANT NOTE: The INI file is dependent on the specific definitions of the IEC application. Verifying that an IEC is running may require viewing results, providing input, etc.

2. For the example used, verify that the cycleCounter variable is incrementing as expected.

Figure 30: Verify the IEC application is running (example)



3 Interface with the IEC application

The procedures included in this section describe how to configure the IEC application to interface with other Totalflow applications.

3.1 Wire I/O registers between the IEC application and the Totalflow G5 device

This procedure describes how to wire an IEC variable to receive its value from a Totalflow register that is external to the IEC Interface application. For example, an IEC variable can reflect the value from the battery voltage at register address 7.3.99. For a complete explanation of I/O wiring, see the ISaGRAF help files.

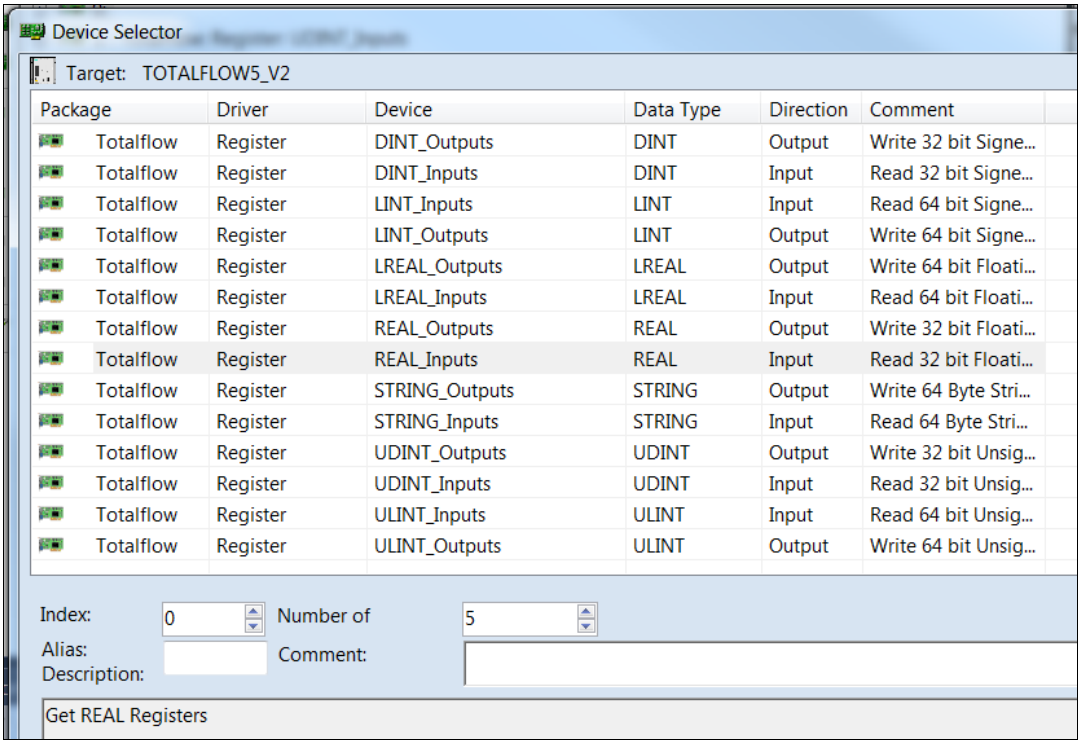
The following are simplified examples of how to:

1. Add an I/O device instance.
2. Wire an IEC variable to an I/O channel.
3. Assign the I/O channel to a Totalflow register address (App.Array.Index).
4. View the results in PCCU.

3.1.1 Add an I/O device instance

An I/O device is a group of I/O channels that are specific to a hardware platform (the G5 device in this case). The I/O channels have a data type and a direction (input to IEC, or output from IEC). The I/O device definitions are sourced from the imported TDB (TOTALFLOW5_V2.tdb). As such, the G5 TDB provides I/O devices for each supported data type (Figure 31).

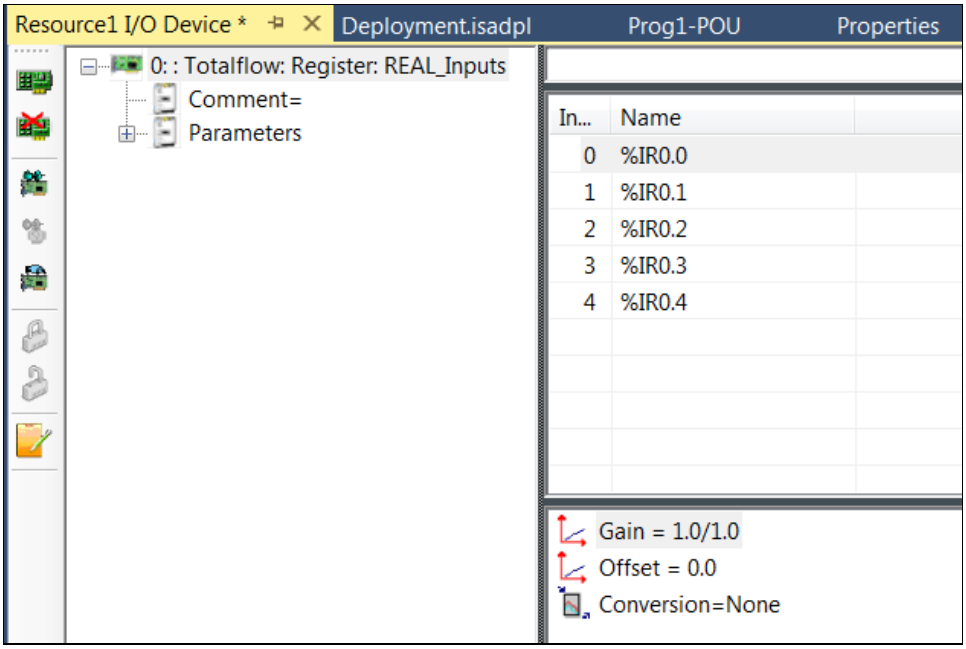
Figure 31: Device selector (available G5 I/O devices)



To add an I/O device:

1. From Solution Explorer, right-click on the resource, then click **I/O Device** to view the I/O Device workspace (Figure 32).

Figure 32: I/O Device workspace



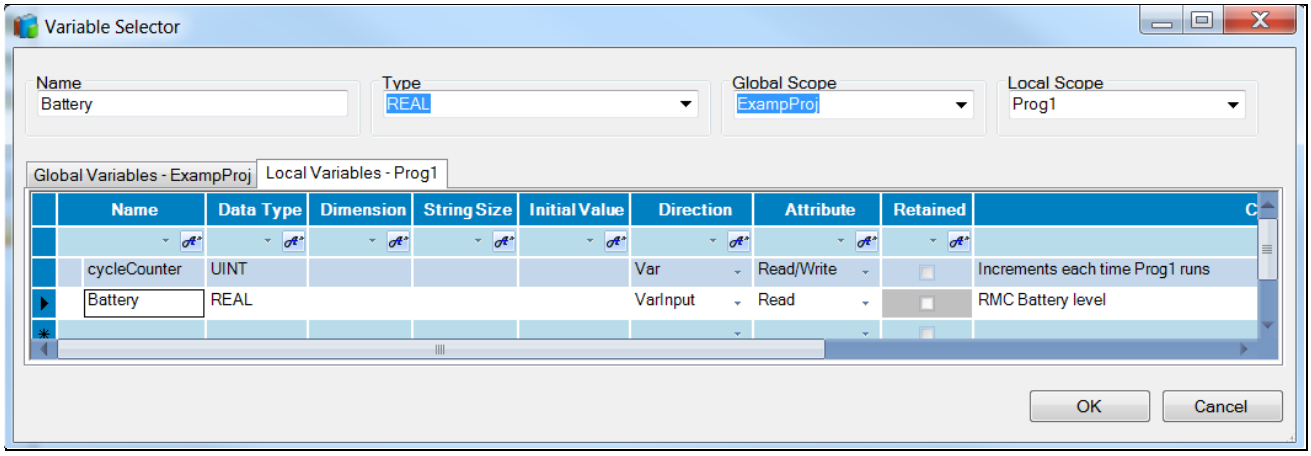
2. In the upper left of the I/O Device workspace, on the I/O Wiring toolbar, click the **Add Device** icon.
3. In the Device Selector dialog (Figure 32), select **REAL_Inputs** device.
4. Enter **5** in the Number of field (this is the number of channels to expose for wiring).
5. Click **OK**.
6. The I/O Device workspace shows the I/O Device instance and the 5 channels in the wiring grid.

3.1.2 Wire an IEC variable to an I/O channel

An I/O channel wires an IEC variable to an I/O device (specific data type, direction, register address). To wire an IEC variable to the first I/O channel:

1. Double-click **First Channel** in the Wiring Grid: %IR0.0. The Variable Selector dialog displays (Figure 33).

Figure 33: Variable Selector dialog box



2. Enter a new variable to wire. Click **OK**. The wired variable displays as %IR0.0=Battery@Prog1 in the wiring grid.

IMPORTANT NOTE: The wired variable name is coded to show %<direction><data type><I/O device index>.<channel index>. See the Table below.

Table 3: ISaGRAF I/O codes

Direction	Code	Type	Code
INPUT	I	DINT / UDINT	D
OUTPUT	Q	REAL	R
		String	S
		LREAL	L
		LINT / ULINT	L

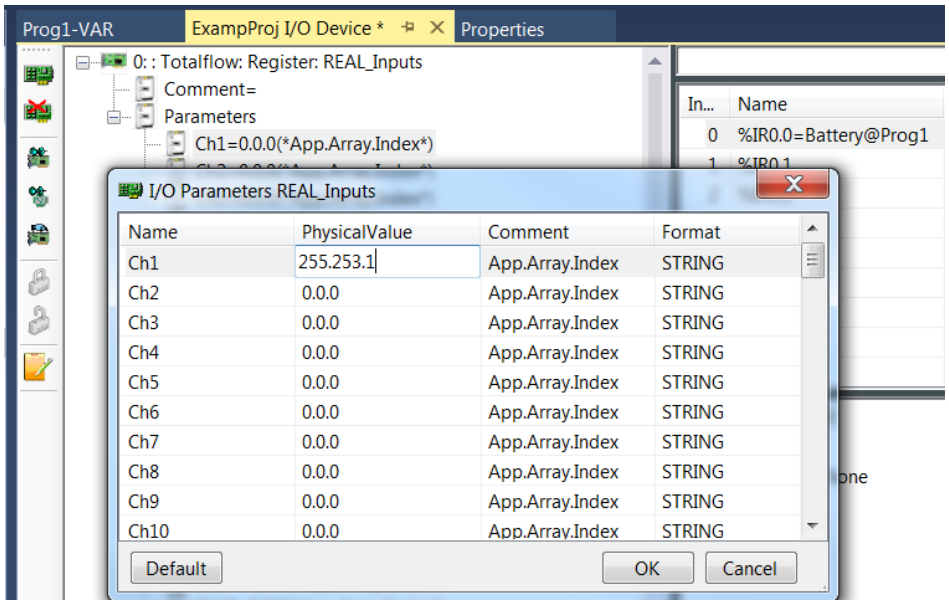
3.1.3 Assign a Totalflow register address to the I/O channel

An I/O channel wires an IEC variable to a Totalflow register address (App.Array.Index). The Totalflow register address is exposed in PCCU, allowing the user to enter an App.Array.Index. The live value at that App.Array.Index register address is then available in the wired IEC variable, thus interfacing and connecting external G5 device application variables to IEC application variables.

To assign a Totalflow register address to the I/O Channel:

1. Expand the I/O Device Instance (0:Totalflow:Register:REAL_Inputs).
2. Expand Parameters and double-click the **Ch1=0.0.0** item. The I/O Parameters REAL_Inputs dialog box displays. See the figure below.

Figure 34: I/O parameters for the I/O Device



3. In the Ch1 Physical Value column, type **255.253.1**.

IMPORTANT NOTE: An App value of 255 resolves to the IEC Interface app slot (91 by default). An Array value of 253 indicates the data type is Register, see Table 4: IEC application register numbers. An index value of 1 indicates to use the first register entry in the array. Therefore, with the IEC Interface Application at slot 91, the address resolves to 91.253.1 at runtime by the IEC interface app.

4. Click **OK** and note that the I/O Device 0:Totalflow:Register: REAL_Inputs, I/O Channel 1 under Parameters indicates the newly created register address.

IMPORTANT NOTE: The IEC developer must use the special IEC variable named NumRegisters to expose assigned register addresses in PCCU. The Initial Value of the NumRegisters variable (see Figure 35) will determine the range of addresses to create (255.253.1 to 255.253.<NumRegisters>).

The value given to NumRegisters must also be used in the G5 IEC Packager when generating an INI. The value should be entered into the Expand Registers field.

Figure 35: Example of the NumRegisters variable

Device1 Prog1-VAR ExampProj-VAR Deployment.isadpl

Name	Data Type	Dimension	String Size	Initial Value	Direction	Attribute	Retained	Comment
_SYSVA_RESMOD	SINT				VarGlobal	Read		Resource execution mode
_SYSVA_CCEXEC	BOOL				VarGlobal	Read/Write		Execute one cycle when application is in cycle to cycle mode
_IO_IR0_2	REAL				VarDirectly	Read		
_IO_IR0_3	REAL				VarDirectly	Read		
_IO_IR0_4	REAL				VarDirectly	Read		
_IO_IR0_1	REAL				VarDirectly	Read		
NUMRegisters	UDINT			5	Var	Read	☑	Number of indirect addresses

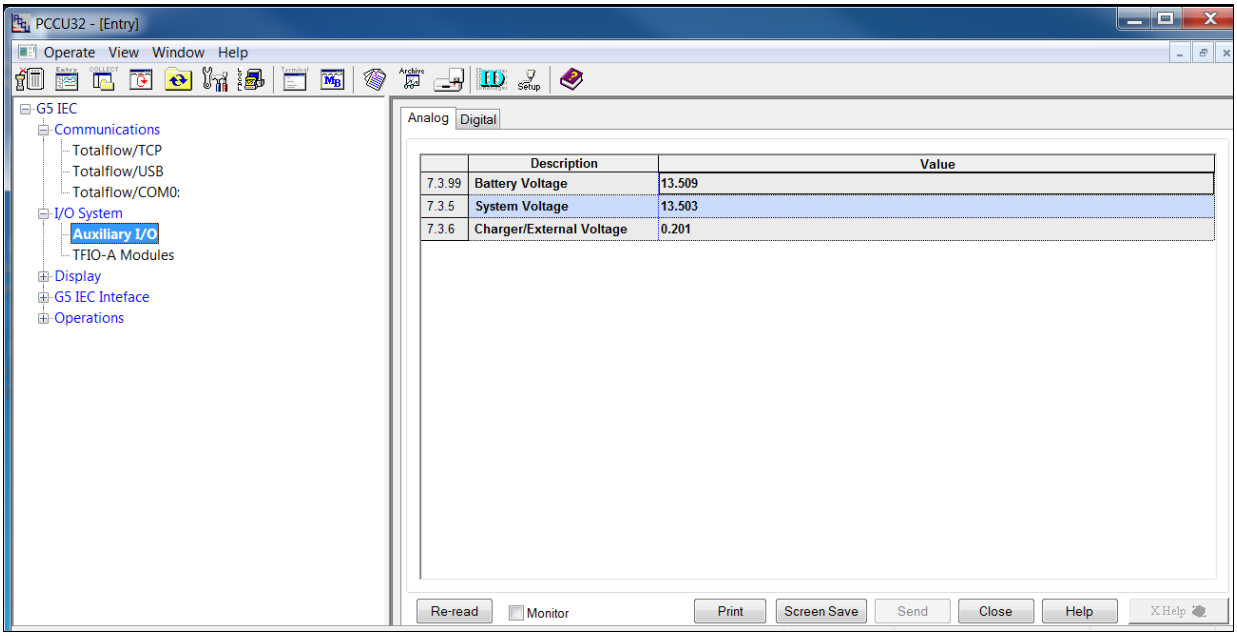
3.1.4 Verify the I/O wiring in PCCU

Follow this verification procedure once the IEC application has been built, packaged, and installed in the device as described in section 2, Develop and run an IEC 61131 application.

To verify that the newly created I/O wiring is functioning as expected:

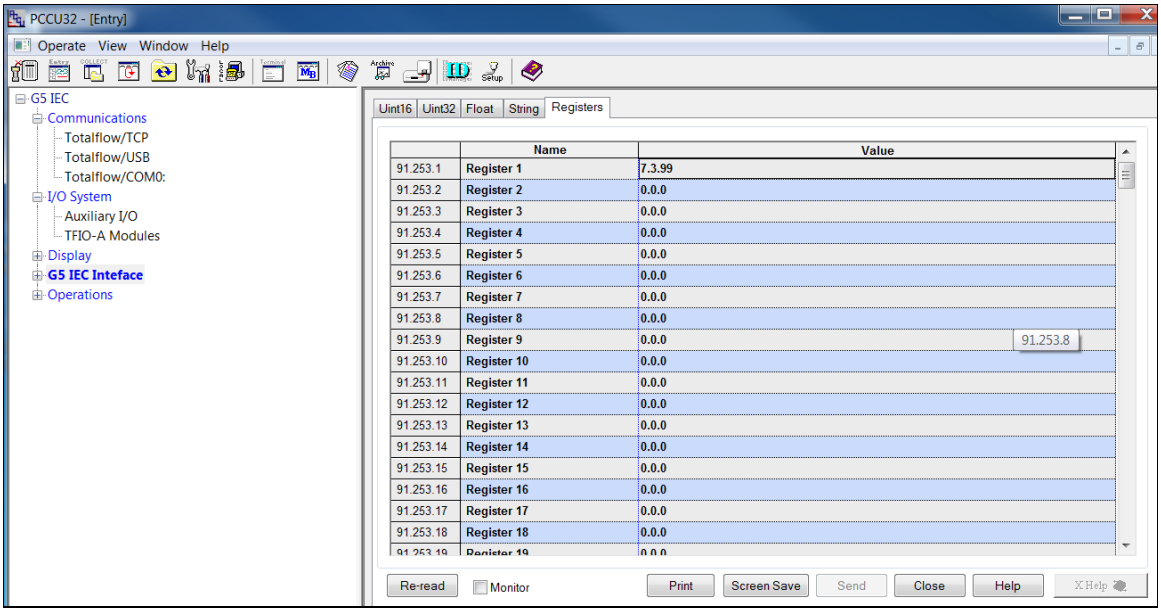
1. Activate and run the IEC application as described in section 2.9, Activate and run the IEC application in the G5 device.
2. On the tree view, expand the **I/O System** application.
3. Select **Auxiliary I/O**.
4. Take note of the Totalflow register address to wire to the IEC application. For the example here, the register for the value of the Battery Voltage is 7.3.99 (See the figure below).

Figure 36: Register address assigned to Battery Voltage



5. On the tree view, select **IEC Interface**.
6. Select the **Registers** tab.

Figure 37: Indirect addressing

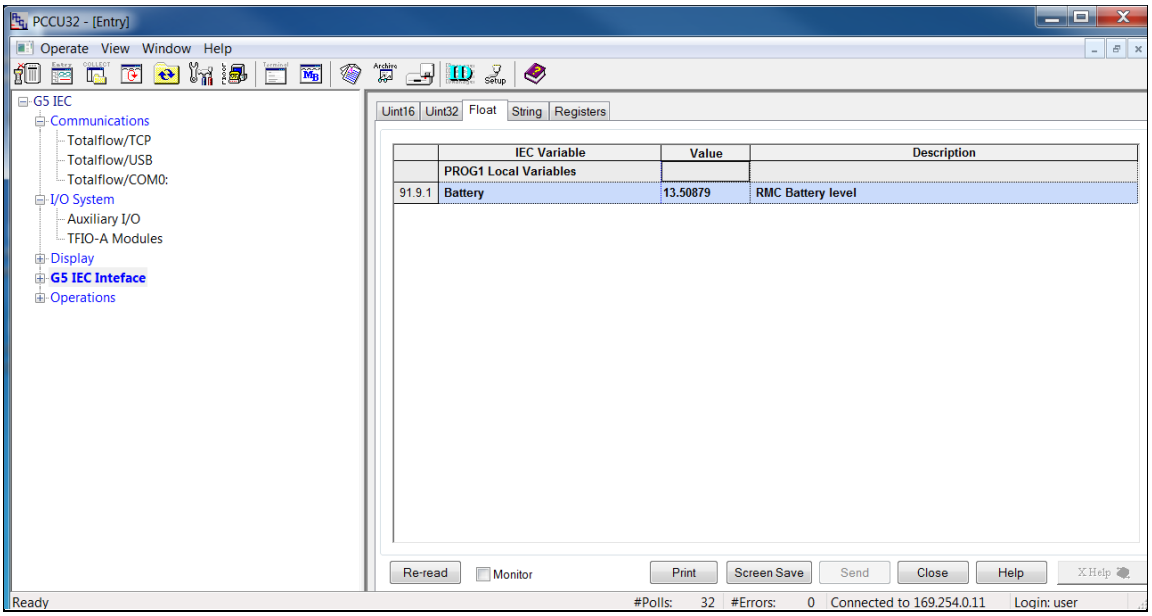


7. Type the register address **7.3.99** into the register 91.253.1, Register[1].

IMPORTANT NOTE: The register 91.253.1 was the register given when assigning a Totalflow register address to the wired I/O channel in section 3.1.3, Assign a Totalflow register address to the I/O channel.

8. Click **Send**.
9. Verify that the Battery Voltage value shown in the IEC interface (Figure 38) matches the value in register 7.3.99 (Figure 36).

Figure 38: Indirect address value



3.2 Assign a specific Totalflow register address to IEC variables

The IEC variables are exposed to the user through the INI file in PCCU. Each IEC variable is automatically assigned a Totalflow register address. For the example used, the IEC variable named Battery appeared at address 91.9.1 under the Float tab in PCCU. See the figure above.

IMPORTANT NOTE: Use fixed address, instead of automatic address.

Fixed address assigns specific addresses to the IEC variables and will remain constant. It is important when external systems are programmed to access the IEC variables.

Instead of using a 255.253.x indirect address, an absolute external address could be used. For example, enter 7.3.99 in the Ch1 Physical Value to source the value directly. Use this method when it is not necessary to enter an App.Array.Index. The address property of an IEC variable can be used to control the Totalflow register address that is assigned. The format for the address is:

- The first digit of the array is the first part of the address.
- The 3 digit hex value of the index (or register number) is the second part of the address.

For example, to set the address to 91.9.2, enter 9002. The slot number is omitted, the hex value of array 9 is entered as A, and the hex value of register number 2 is entered as 002.

The table below shows the array number to use depending on the data type of the IEC variable to be addressed.

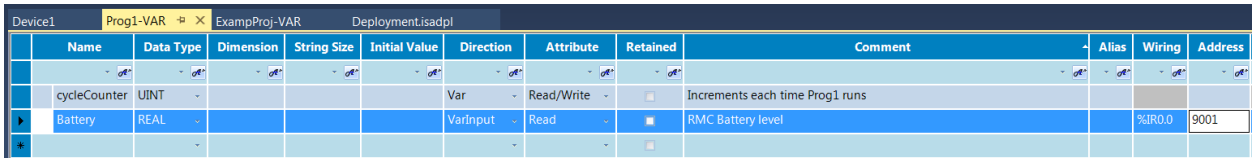
Table 4: IEC application register numbers

Type	Array	Type	Array	Type	Array
Bool	0	**Int32**	5	**Float**	9
Int8	1	**Uint32**	6	**Double**	10
Uint8	2	**Int64**	7	**String**	11
Int16	3	**Uint64**	8	**Time**	12
Uint16	4			**Date**	13
				Register	253

To set the assigned register address for the Battery IEC variable:

1. In the Prog1 Local variables window (Figure 39), type the address into the Address column of the Battery variable.

Figure 39: Assigning a register to an IEC application variable



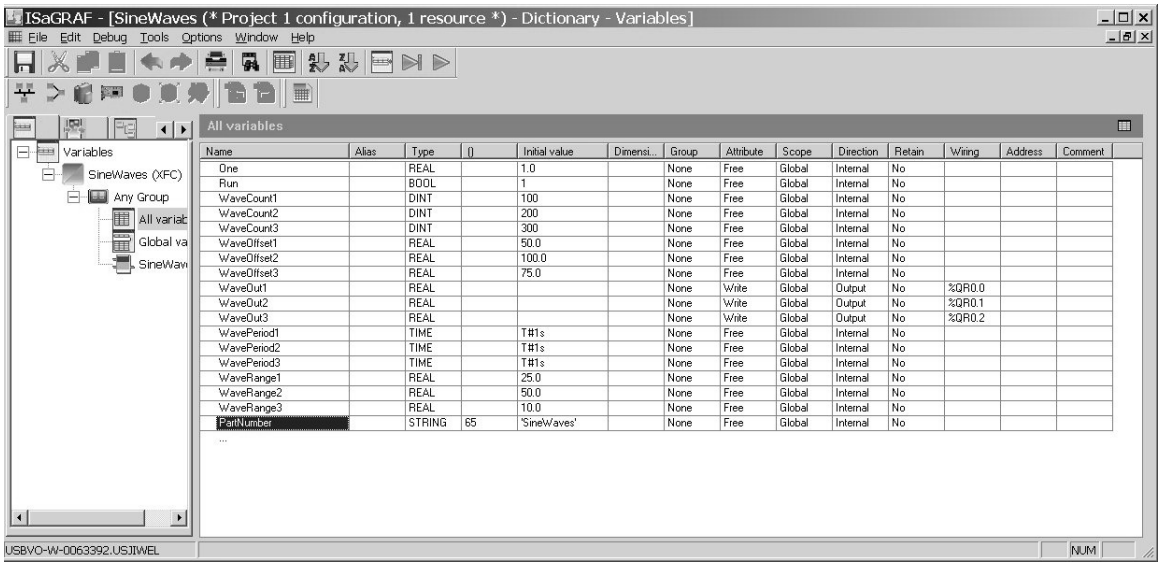
3.3 Creating the INI file for an IEC application

ABB Totalflow's PCCU32 uses INI files to define user screens within PCCU32. INI files are text files. PCCU32 parses the INI files to create the various screens within PCCU32. If a custom INI file is not created for the IEC application, a default INI file is provided.

To create a custom INI file:

1. Create the variable, PARTNUMBER, within the ISaGRAF development tool (Figure 40).

Figure 40: Create an INI file

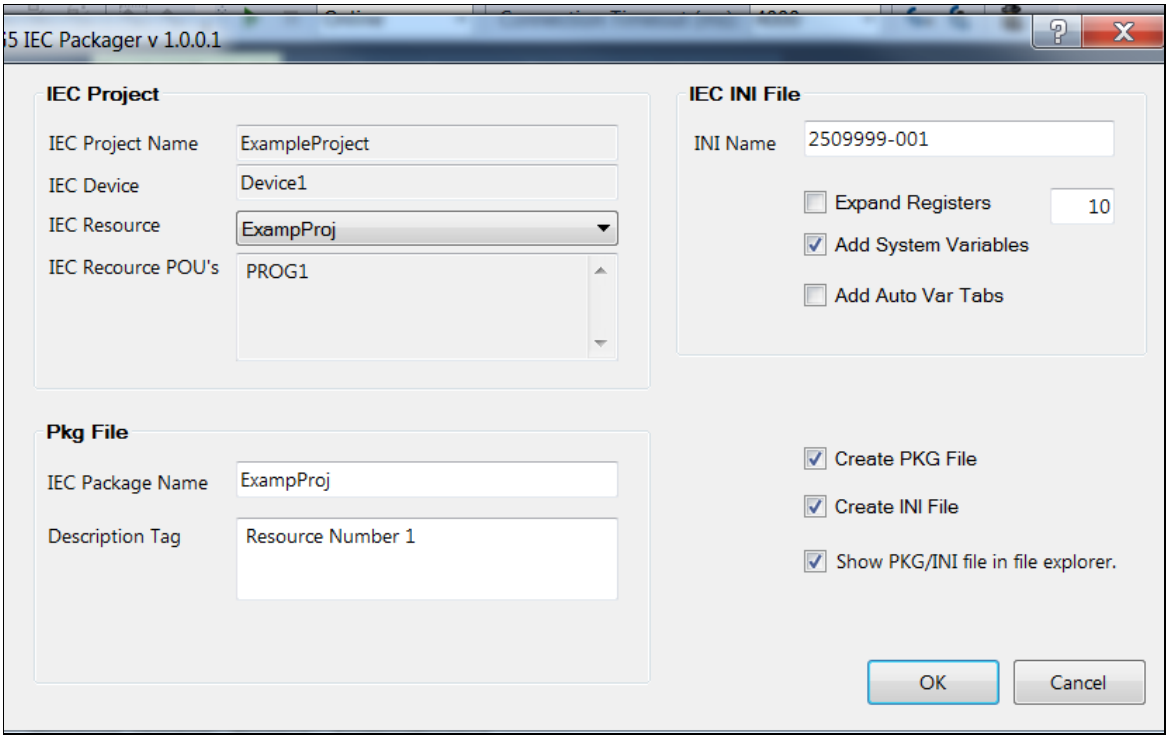


2. In the IEC Interface application in PCCU32, point to the specific, custom INI file to be used. For the purpose of this manual, the custom INI file is named Sine-001.ini (or Sine-001fm.ini Expert view) and should reside in the \PCCU\IniFiles folder.

IMPORTANT NOTE: The file prefix cannot be longer than ten (10) characters. PCCU32 can automatically decrement the revision level (-001) to a lower number of the file if the current version does not exist on the current PCCU32 directory.

IMPORTANT NOTE: ABB Totalflow supplies a program, G5 IEC Packager, which will read an ISaGRAF project and generate an INI file as a starting point for a developer to customize. (see Figure below). Use Add Auto Var Tabs to make an Expert view file. Make all of the customizations and duplicate the files as an Advanced view INI file.

Figure 41: G5 IEC Packager (INI generator)



3.4 Customizing the INI file

INI files are text files that are read by PCCU32 or WinCCU32 to determine how to display data within an application. A simple example is presented in this section to illustrate the screens that are rendered based on the definitions in the INI file. Refer to the INI Programmer's Manual for details about creating the INI file and for reference information about INI commands.

3.4.1 Example of INI file text

The following example shows text used in an INI file. The description for each line is provided below. The line numbers are for reference only and are not used in an actual INI file.

Figure 42: INI file text example

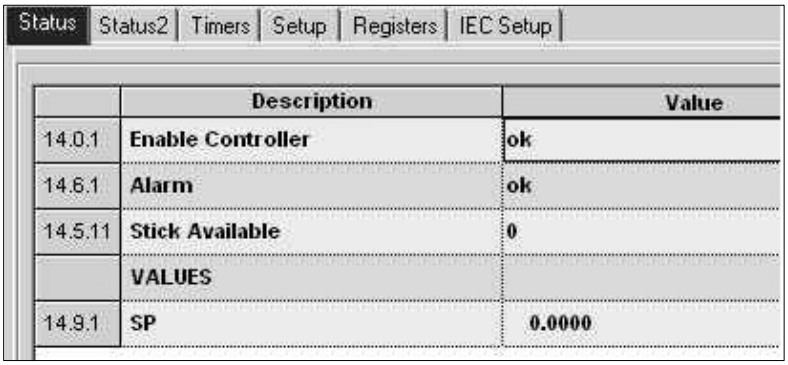
```
1 [Status]
2 dsc:Enable Controller;cmd:13.0.1;typ:e;lst:ok=0,ALARM=1;
3 dsc:Alarm;cmd:13.6.1;typ:J;lst:ok=0,ALARM=1;
4 dsc:Stick Available;cmd:13.5.11;typ:l;
5 dsc:VALUES;typ:Z;
6 dsc:SP;cmd:13.9.1;typ:f;
```

Legend: INI file text example

Line	Description	Line	Description
1	The name of the first tab	**4**	Stick Available. The register is displayed as a 4-byte integer.
2	Enable Controller. The register displayed is array 0, index 1. The 13 represents the App type, not the application number. The value is to be displayed as a drop-down box of values: OK or ALARM. The register type is byte integer as indicated by the type.	**5**	VALUES. Does not display a register value. This causes a header for the next several values to help the user distinguish sections of values.
3	Similar to line 2 but with a data type of 4-byte integer.	**6**	SP. The register is displayed as a floating point value with default width and precision.

The definitions in this example render the screen shown in the figure below. The Status tab contains the Enable Controller, Alarm, Stick Available, VALUES and SP variables. The value of the SP variable are defined in the INI file.

Figure 43: PCCU view of the example INI file



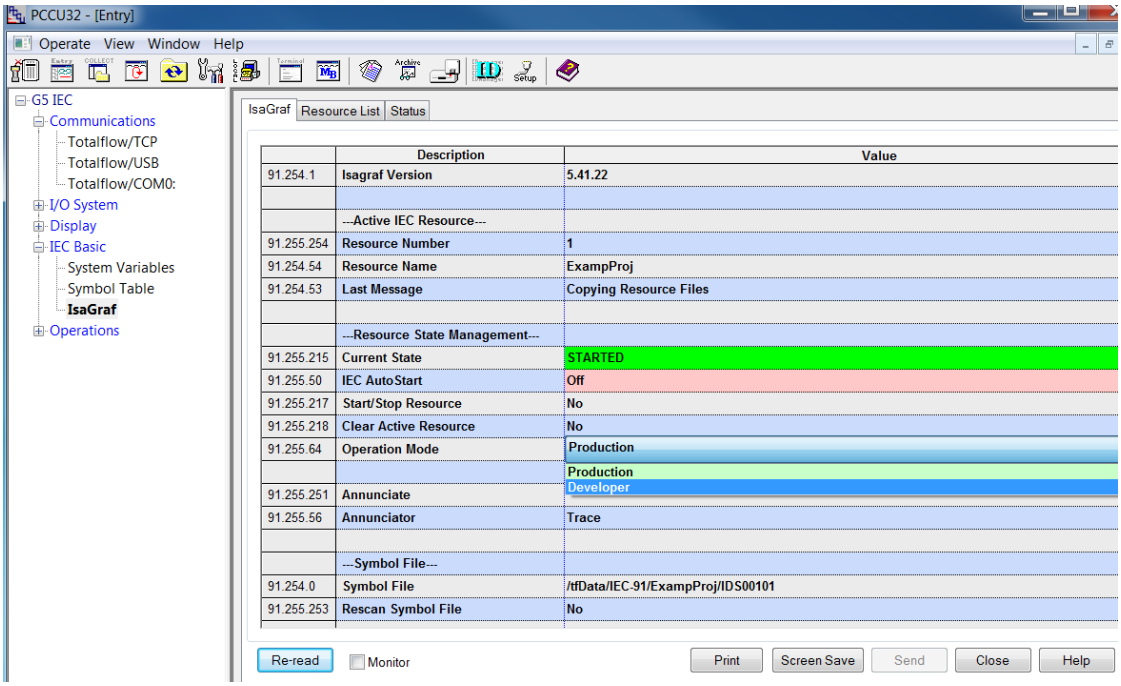
4 Direct download the resource from the ISaGRAF IDE

Before Download, change the operation mode in PCCU: on the ISaGRAF tab to Developer.

1. On the ISaGRAF tab, for the operation mode, Select **developer.**
2. Click **Send**.

Note the Resource Number value in the ISaGRAF screen. This will be used to update the correct application with the downloaded code.

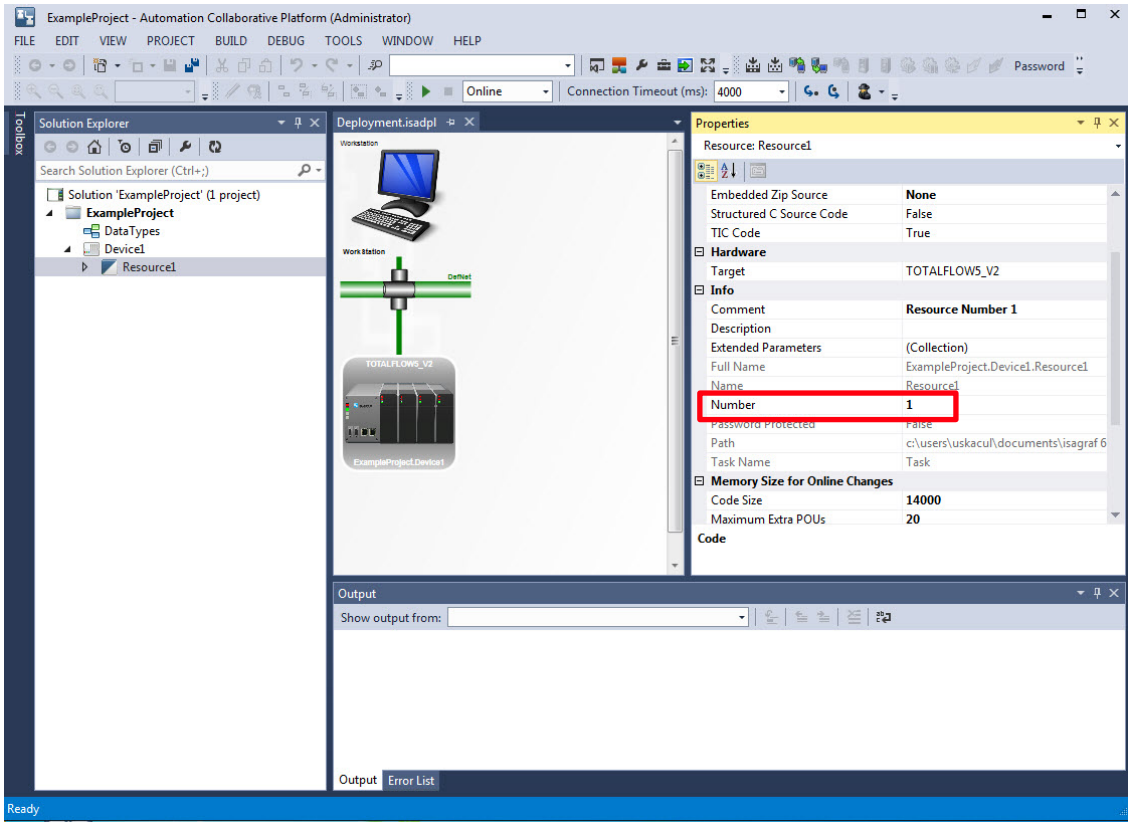
Figure 44: Developer mode



The Properties Resource Number (Figure 45) must match the Resource Number on the TOTALFLOW Controller. To see this on the TOTALFLOW Controller go to:

1. PCCU Entry Mode. Click **IEC Interface.**
2. Click **ISaGRAF.**
3. Click **ISaGRAF Tab.**
4. Click **Resource Number (APP.255.254).**

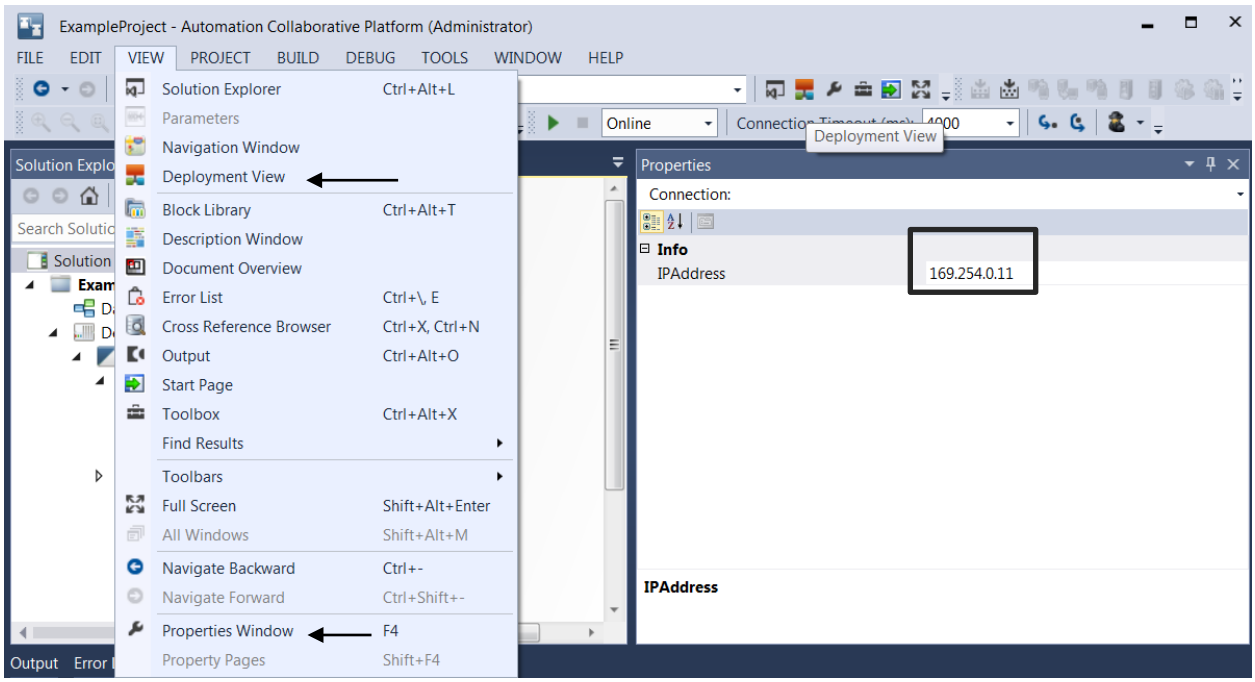
Figure 45: View the properties > resource number



Download to the target device using the ISaGRAF Workbench (IDE):

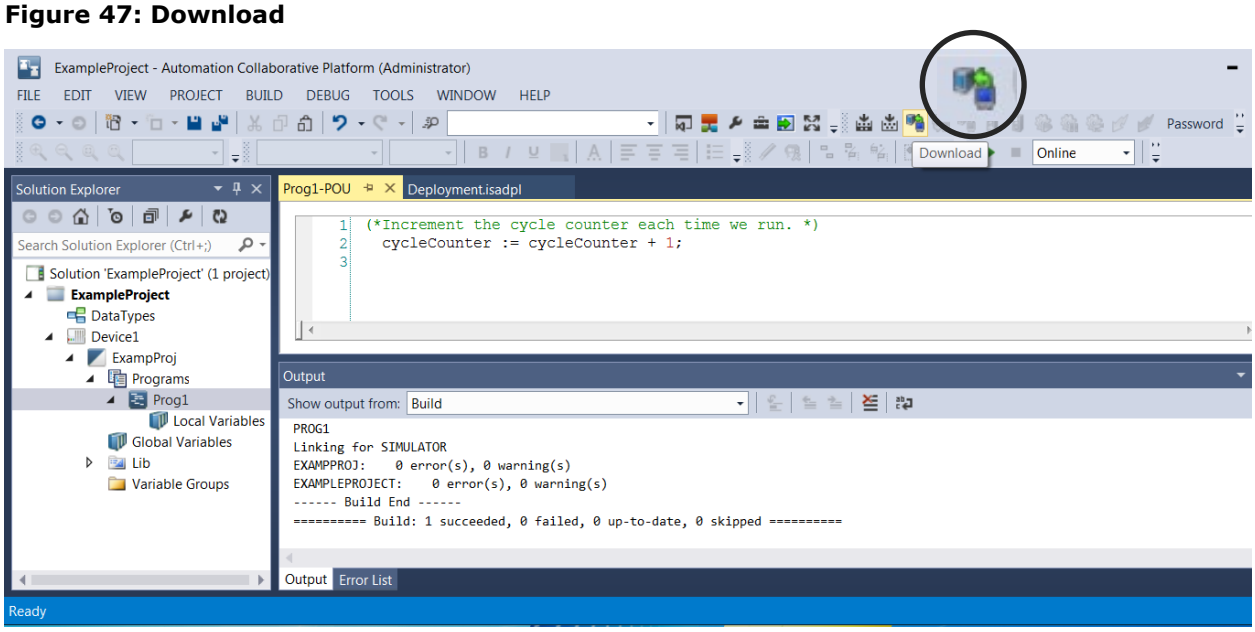
1. Select the **View** tab.
2. Select **Deployment View>Properties.**
3. Enter the IP Address.

Figure 46: Enter IP address



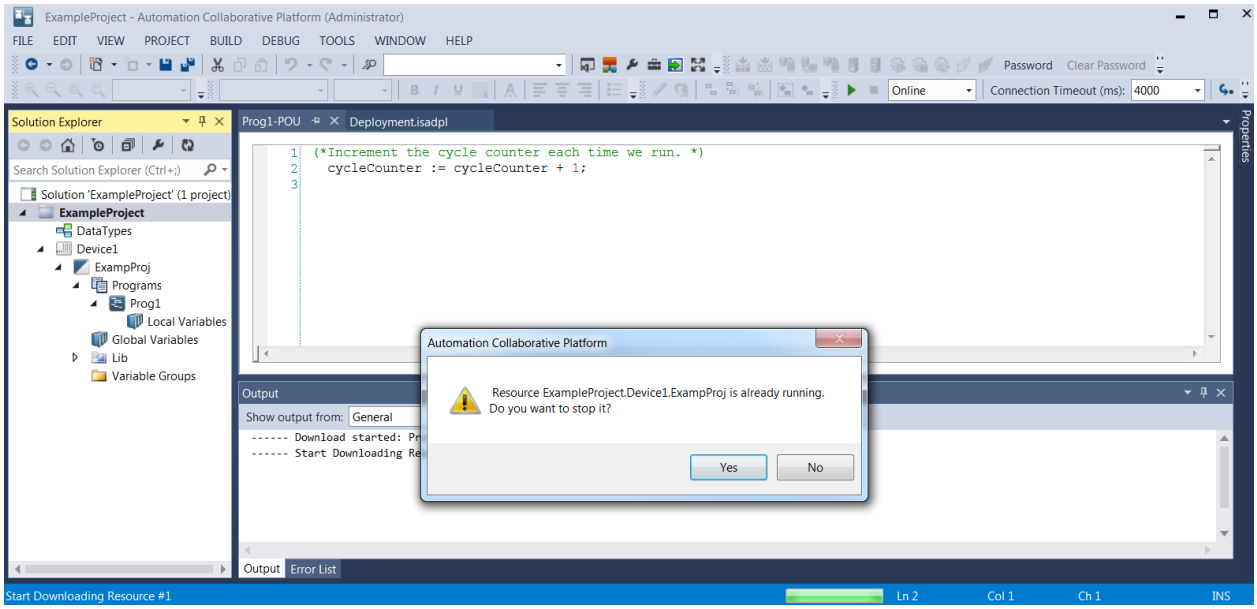
4. Select the **Download** icon to download the resource.

Figure 47: Download



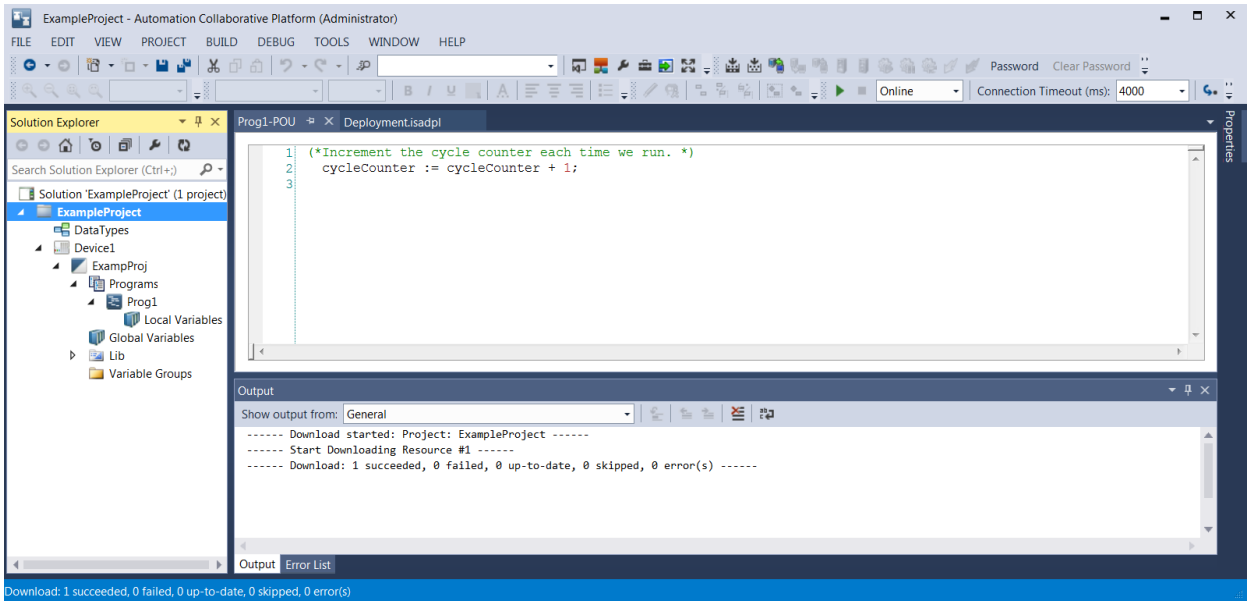
The resource is running, the following alert will open.

Figure 48: Download Resource alert



5. Click **Yes**.

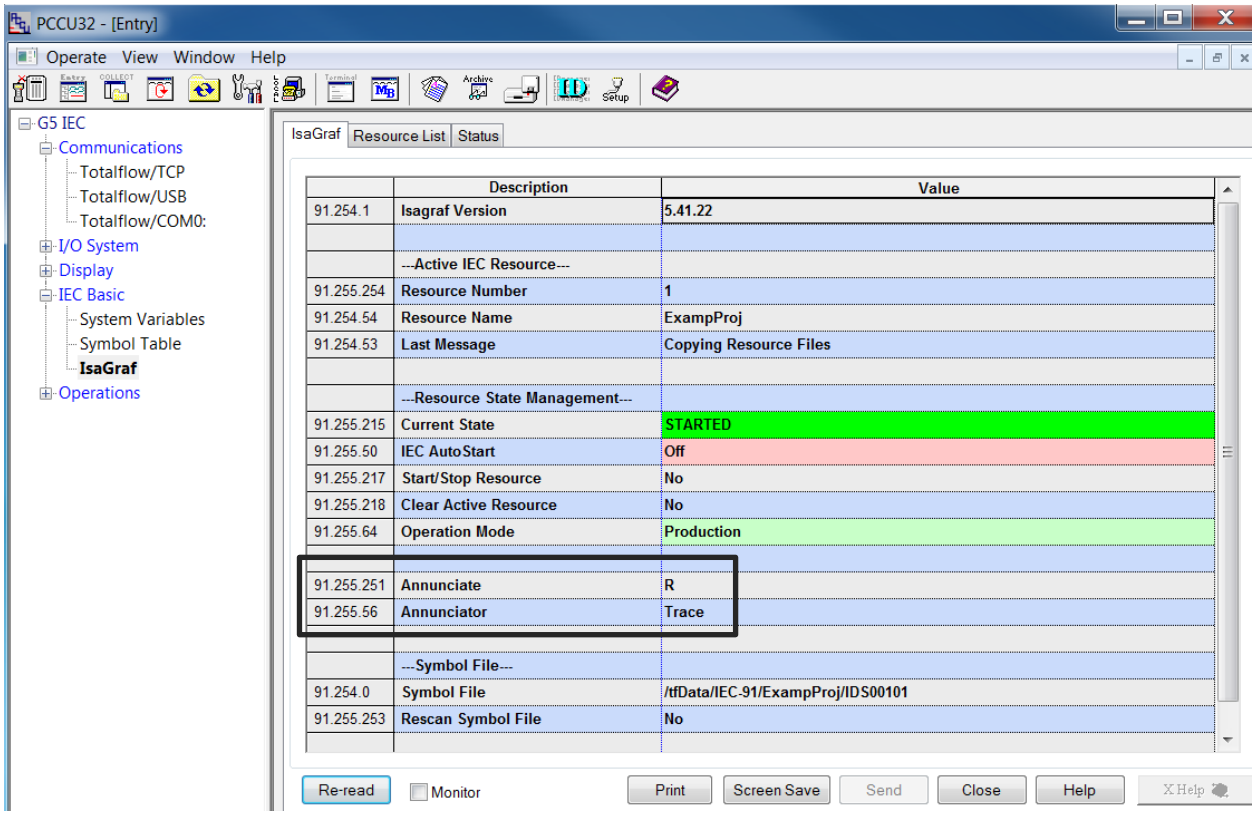
Figure 49: Result of a successful resource download



5 Annunciators

To trace the progress of the running resource, view the **ISaGRAF tab**>**Annunciate** feature (255.25.1).

Figure 50: Annunciate feature



5.1 Annunciation to view the state on RMC display

The Annunciator option has several modes that display application statues to the RMC board's display.

Manual: Displays the character selected in the Annunciate field

Trace: Displays pre-defined characters as shown in Table 5. These characters are associated with different status in the application.

Auto: Building on Trace, after the application is running, a "breathing" asterisk is shown on the RMC display (highest CPU usage.)

Table 5: Pre-defined characters:

Indicator	Description
K	Killed
C	Cleared Active Resource
H	Halting Signal on Annunciator
S	Starting Resource
0-6	Progress of Resource Connect/Disconnect
A	Active Resource
F	Failed to Connect Resource
X	Resource Disconnected

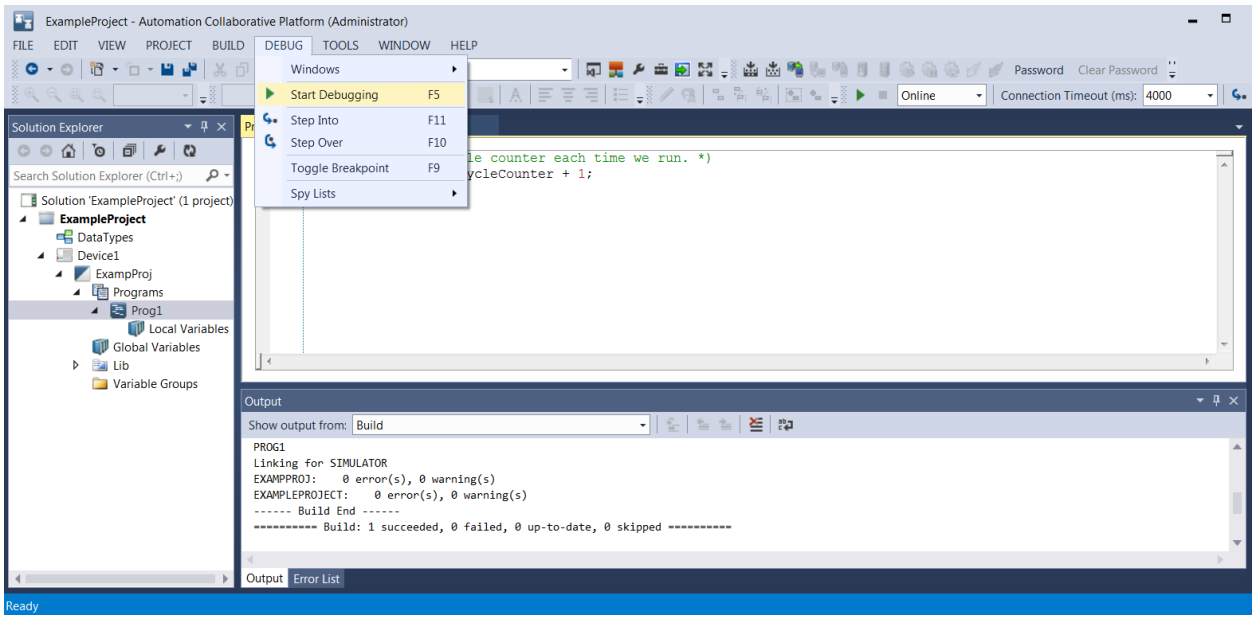
Indicator	Description
R	Running IEC Application
F+ (numeric value)	(Numeric Value)’ = Failed Connection Working toward Recovery
N	Not Running
I	Idle
r	Requesting PARTNUMBER from IEC application at startup.
M	Resource files are missing and cannot be load
T	The resource process has terminated unexpectedly or is not reachable
P	State change request is pending
U	App has undecided license allocation and is not running.
D	App is disabled due to license allocations and is not running.
?	Unknown Kernel State

IMPORTANT NOTE: Other characters which are not listed in this document may appear due to internal state management.

6 Debug mode

1. Select the **Debug** tab from the main toolbar.
2. Select **Start Debugging.**

Figure 51: Debug tab



After debug is completed, the system is ready for use.

7 Program example

An example of an ISaGRAF program is included in this section for reference. The program uses ST (structured text) language and implements a hysteresis on 10 inputs. An input value larger than the HiHiLimit will set the output to a 1 and will reset the output when the input is below the HiLimit. The example includes an event log function that will record the Cold start, Warm Start, and change in the output value.

7.1 Main program

```
(* setup initial CFG values - first time. They will be read back from config file after a first time.*)
IF (Cold = FALSE) THEN
        Enable    := FALSE;
        HiHiLimit  := 100.0;
        HiLimit     :=  90.0;
        (* clear event log on first run *)
        FOR pos := 1 TO ANY_TO_DINT(Event_Log_Size) DO
                Event_Code[pos]  := State_Unused;
                Event_Input[pos] := 0;
                Event_Value[pos] := 0.0;
                Event_Stamp[pos] := 0;
        END_FOR;
        ReturnStatus := Event_Log(State_Cold, 0, 0.0);
        Cold := TRUE;
END_IF;
(* setup variables after a warm start *)
IF (Warm = False) THEN
        FOR pos := 1 TO ANY_TO_DINT(Input_Size) DO
                State[pos]  := State_OK;
                Output[pos] := 1;
        END_FOR;
        ReturnStatus := Event_Log(State_Warm, 0, 0.0);
        Warm := TRUE;
END_IF;
(* if not enabled – don't execute main body *)
IF (Enable) THEN
        FOR pos := 1 TO ANY_TO_DINT(Input_Size) DO
                IF (State[pos] = State_OK and Value[pos] > HiHiLimit[pos]) THEN
                        State[pos] := State_HiHi;
                        ReturnStatus := Event_Log(State, pos, Value);
                        Output[pos] := 0;
                ELSIF (State[pos] = State_HiHi and Value[pos] < HiLimit[pos]) THEN
```

```
                    State[pos] := State_OK;
                    ReturnStatus := Event_Log(State, pos, Value);
                    Output[pos] := 1;
                END_IF;
            END_FOR;
END_IF;
```

7.2 EVENT_LOG function (Code, Input, Value)

```
(* Move all data down by one row – start at bottom, filling from above *)
FOR pos := ANY_TO_DINT(Event_Log_Size) TO 2 BY -1 DO
        Event_Code[pos]  := Event_Code[pos - 1];
        Event_Input[pos] := Event_Input[pos - 1];
        Event_Value[pos] := Event_Value[pos - 1];
        Event_Stamp[pos] := Event_Stamp[pos - 1];
END_FOR;
(* place new log in first row *)
Event_Code[1]  := Code;
Event_Input[1] := Input;
Event_Value[1] := Value;
Event_Stamp[1] := DateTime;
Event_Log := TRUE;
```

7.3 Variables

The tables below describe the defined variables and words used by the example program shown above.

Table 6: Defined variables

Name	Type	Retain	Scope	Dim	I/O	Init
HiHiLimit	Real	Yes	Global	[1..10]	rw	
HiLimit	Real	Yes	Global	[1..10]	rw	
Value	Real	Yes	Global	[1..10]	rw	
Cold	bool	Yes	Global		rw	
Warm	bool	No	Local		rw	
DateTime	uint32	No	Global		R	
Enable	uint32	Yes	Global		rw	
Event _Log_Size	uint32	No	Global		rw	30
Event_Input	uint32	Yes	Global	[1..30]	rw	
Event_Code	uint32	Yes	Global	[1..30]	rw	

Event_Stamp	uint32	Yes	Global	[1..30]	rw	
Input_Size	uint32	No	Global		rw	10
Pos	uint32	No	Main		rw	
Pos	uint32	No	Event_Log		rw	
ReturnStatus	uint32	No	Main		rw	
Output	uint32	No	Global	[1..10]	W	
State	uint32	Yes	Global	[1..10]	rw	
DateTime	uint32	No	Global		R	
PartNumber	String	No	Global		rw	2509999
NumberRegisters	String	No	Global		rw	1

Table 7: Defined Words

Define words	Value
State_OK	0
State_HiHi	1
State_Cold	2
State_Warm	3
State_Unused	4

8 Read and write efficiency

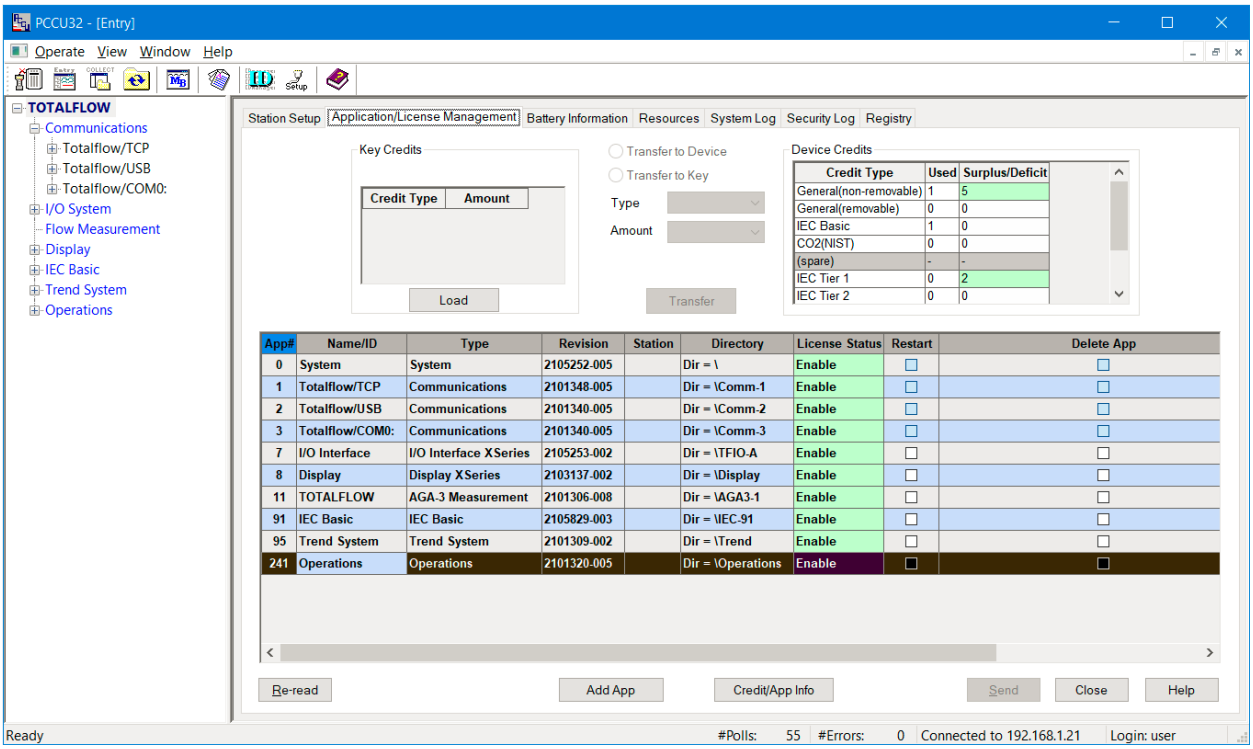
When developing your own custom Totalflow application with ISaGRAF Workbench, there are several methods of passing data between your ISaGRAF application and the other Totalflow applications running on the same controller. Each of these communication methods has different advantages. These methods vary widely in how much CPU load is required to process the data request. The following is a comparison of these communication methods.

8.1 Background

Your custom ISaGRAF application runs as a separate process along-side the built-in Totalflow applications. The Totalflow.exe process contains all the built-in Totalflow applications (AGA3, I/O Interface, Trend, Operations, etc.).

All the applications listed on the PCCU Application/License Management tab are contained inside the Totalflow.exe process.

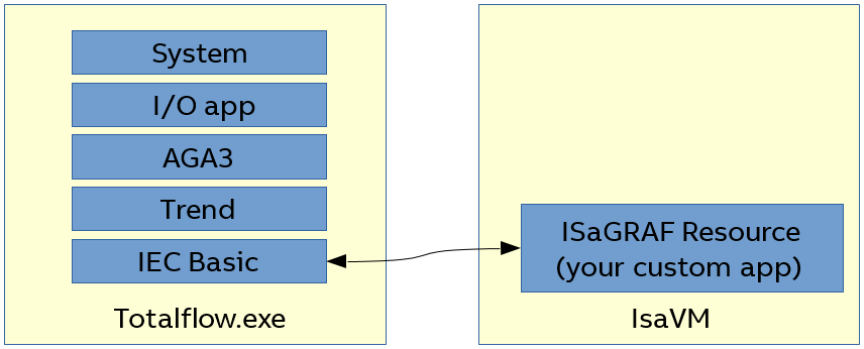
Figure 52: Application/License Management tab



The IsaVM process runs along-side Totalflow.exe on the same controller. IsaVM is the interpreter that runs your custom ISaGRAF application. The ISaGRAF application must send messages to the Totalflow process to request register values.

The IEC Basic app acts as a communication interface to pass register values between your custom ISaGRAF application and the Totalflow applications.

Figure 53: Communication between ISaGRAF and Totalflow apps



There are several communication methods available to pass data between your custom ISaGRAF application and the Totalflow.exe applications:

- I/O Device Inputs and Outputs
- Get*() and Set*() function blocks
- Read and write ISaGRAF registers from Totalflow.exe applications

For instructions about how to configure each of these read/write methods, see section 3, Interface with the IEC application.

Following is a brief description of each of these methods.

8.2 ISaGRAF I/O Device

The ISaGRAF I/O Device is the most efficient method of reading register values from the Totalflow application into your custom ISaGRAF application.

The ISaGRAF I/O Device can read 64 Totalflow registers in a single message request.

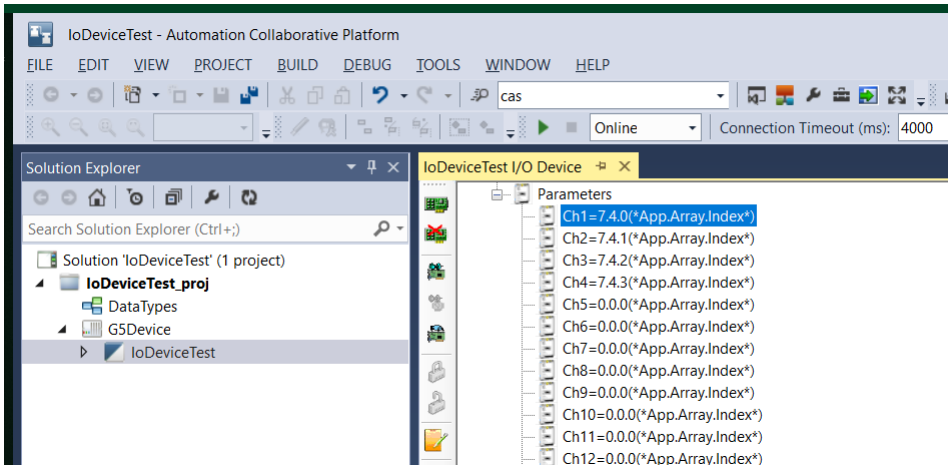
Advantages:

- The most efficient way to read and write registers from Totalflow applications.

Disadvantages:

- No programmatic control. Read and write is automatic and cannot be disabled.

Figure 54: I/O Device



8.3 Get*() and Set*() Function Blocks

Get*() and Set*() function blocks are called from within your ISaGRAF code. These function blocks provide some capability not available with I/O Device, but require much more CPU overhead.

Advantages:

- The App/Array/Index inputs can be variables.
- The function can be called conditionally only when required.

Disadvantages:

- High CPU cost

Figure 55: GetRealTest

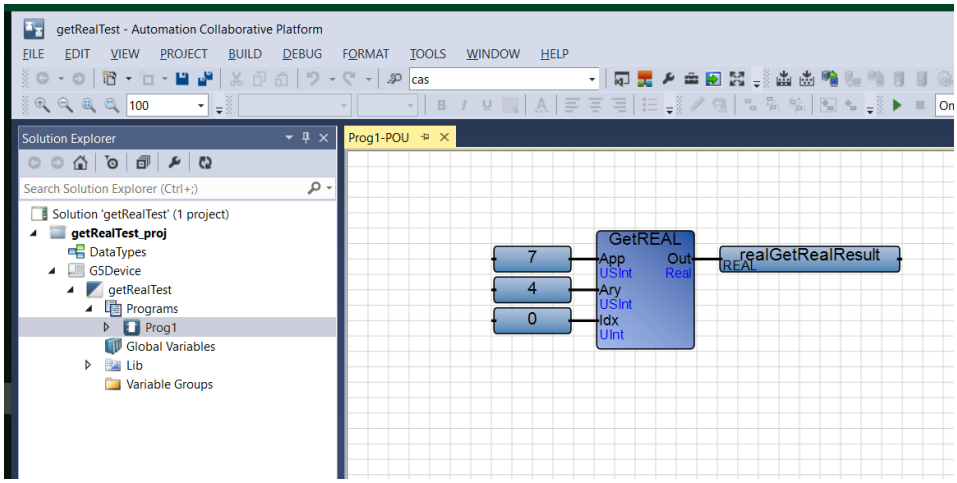
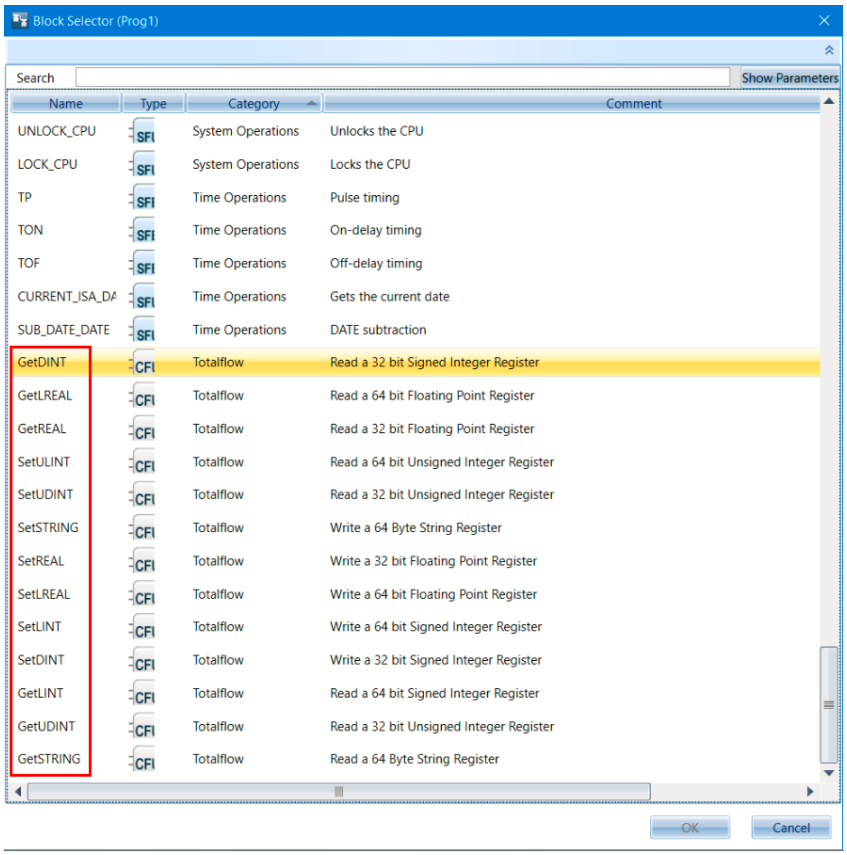


Figure 56: Block selector



8.4 Totalflow apps read IsaVM registers

The following example shows an Operations app reading registers from the ISaGRAF application (app slot 91).

This method has a very high CPU cost. Its use should be minimized if possible. When data must be read from the ISaGRAF application (for example, status to be displayed on SCADA or HMI), it is recommended to use an intermediate Holding Registers application. Configure the HMI to read the status from a Holding Register, and use ISaGRAF I/O Device to write the status to the Holding Register.

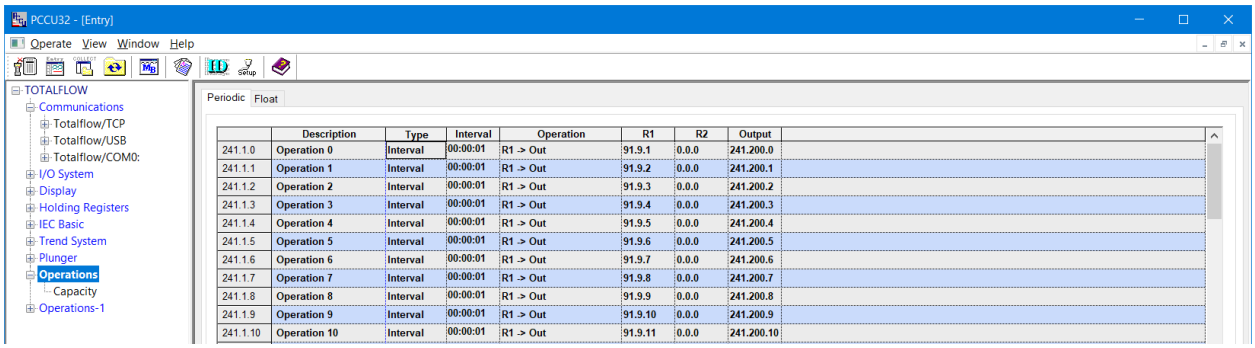
Advantages:

- Configuration changes can be made easily without re-compiling the ISaGRAF application.

Disadvantages:

- Very high CPU cost

Figure 57: Reading ISaGRAF app from Totalflow operations app



	Description	Type	Interval	Operation	R1	R2	Output
241.1.0	Operation 0	Interval	00:00:01	R1 -> Out	91.9.1	0.0.0	241.200.0
241.1.1	Operation 1	Interval	00:00:01	R1 -> Out	91.9.2	0.0.0	241.200.1
241.1.2	Operation 2	Interval	00:00:01	R1 -> Out	91.9.3	0.0.0	241.200.2
241.1.3	Operation 3	Interval	00:00:01	R1 -> Out	91.9.4	0.0.0	241.200.3
241.1.4	Operation 4	Interval	00:00:01	R1 -> Out	91.9.5	0.0.0	241.200.4
241.1.5	Operation 5	Interval	00:00:01	R1 -> Out	91.9.6	0.0.0	241.200.5
241.1.6	Operation 6	Interval	00:00:01	R1 -> Out	91.9.7	0.0.0	241.200.6
241.1.7	Operation 7	Interval	00:00:01	R1 -> Out	91.9.8	0.0.0	241.200.7
241.1.8	Operation 8	Interval	00:00:01	R1 -> Out	91.9.9	0.0.0	241.200.8
241.1.9	Operation 9	Interval	00:00:01	R1 -> Out	91.9.10	0.0.0	241.200.9
241.1.10	Operation 10	Interval	00:00:01	R1 -> Out	91.9.11	0.0.0	241.200.10

8.5 Totalflow apps write IsaVM registers

The following example shows an Operations app writing registers to the ISaGRAF application (app slot 91). Software architecture limits these Totalflow-to-ISaGRAF writes to 5 registers-per-second.

This method should be avoided if possible. If data must be written to the ISaGRAF application (for example, commands from SCADA or HMI), it is recommended to use an intermediate Holding Registers application. Configure the HMI to write its command to a Holding Register, then use I/O Device to read the Holding Register into the ISaGRAF application.

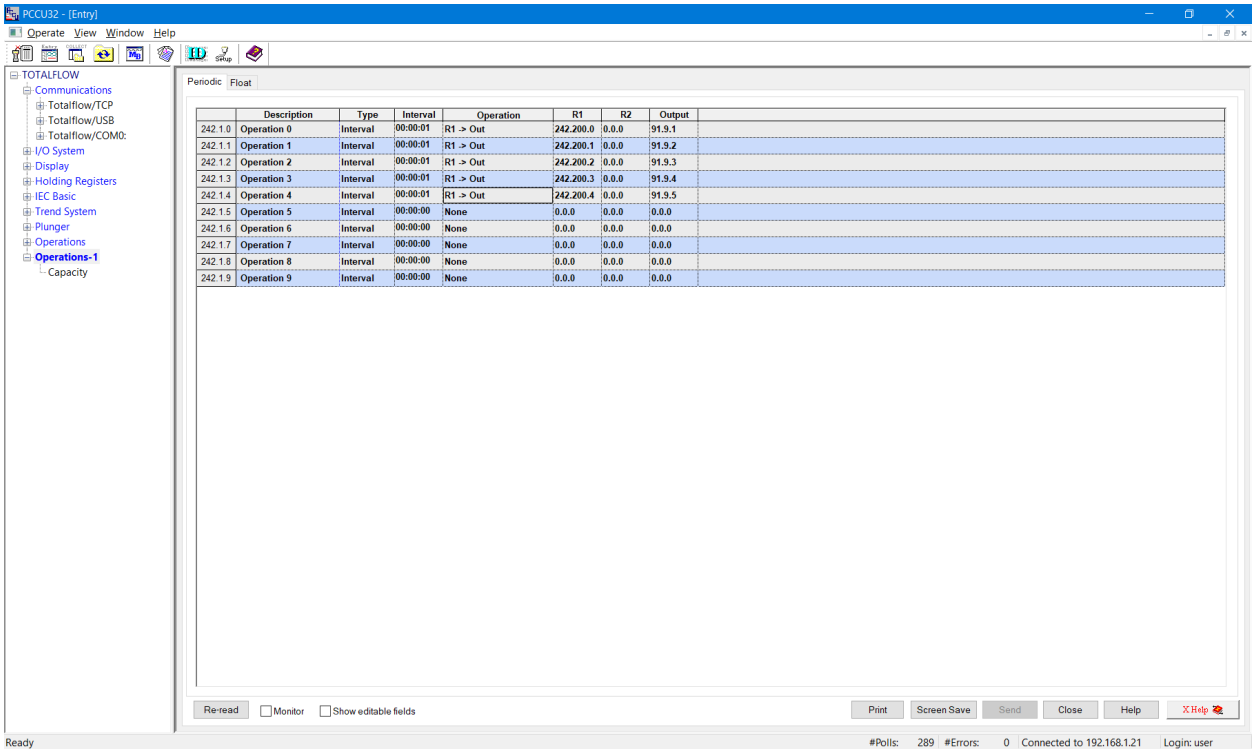
Advantages:

- Configuration changes can be made easily without re-compiling the ISaGRAF application.

Disadvantages:

- Limited to 5 registers-per-second

Figure 58: Writing ISaGRAF from Totalflow operations app



	Description	Type	Interval	Operation	R1	R2	Output
242.1.0	Operation 0	Interval	00:00:01	R1 -> Out	242.200.0	0.0.0	91.9.1
242.1.1	Operation 1	Interval	00:00:01	R1 -> Out	242.200.1	0.0.0	91.9.2
242.1.2	Operation 2	Interval	00:00:01	R1 -> Out	242.200.2	0.0.0	91.9.3
242.1.3	Operation 3	Interval	00:00:01	R1 -> Out	242.200.3	0.0.0	91.9.4
242.1.4	Operation 4	Interval	00:00:01	R1 -> Out	242.200.4	0.0.0	91.9.5
242.1.5	Operation 5	Interval	00:00:00	None	0.0.0	0.0.0	0.0.0
242.1.6	Operation 6	Interval	00:00:00	None	0.0.0	0.0.0	0.0.0
242.1.7	Operation 7	Interval	00:00:00	None	0.0.0	0.0.0	0.0.0
242.1.8	Operation 8	Interval	00:00:00	None	0.0.0	0.0.0	0.0.0
242.1.9	Operation 9	Interval	00:00:00	None	0.0.0	0.0.0	0.0.0

8.6 Comparison

These communication methods vary widely in how much CPU load is required to process the data request. The following table shows the CPU cost-per-register for each method.

Table 8: CPU cost for each method

Method	CPU % / register	Notes
I/O device	0.004 %	64 registers per I/O device
Get*() and Set*() function blocks	0.1 %	
Totalflow app read from IsaVM	0.2 %	
Totalflow app write to IsaVM	N/A	5 register limit

To find the CPU load caused by reading registers into your ISaGRAF application:

CPU load = (# of registers to be read) x (CPU % / register)

Example 1. If you read 128 registers using GetReal() function blocks:

- CPU load = 128 x 0.1 %
- CPU load = 12.8 %

Example 2. If you read 128 registers using the I/O Device:

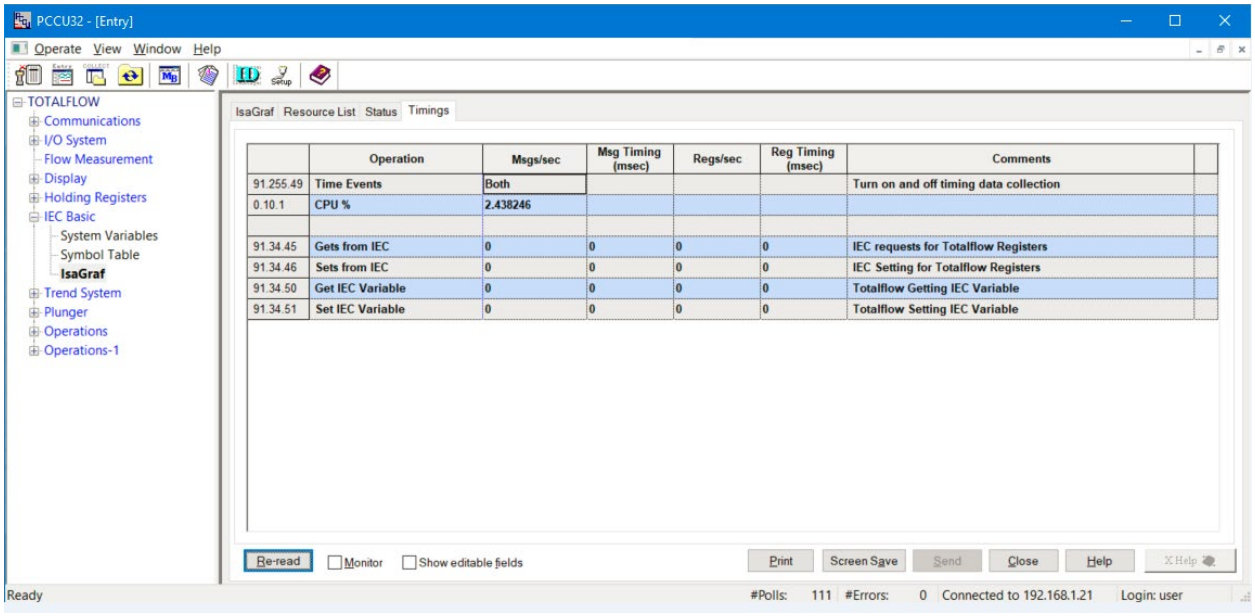
- CPU load = 128 x 0.004 %
- CPU load = 0.51 %

8.7 Run-time diagnostics

RMC flash 2105457-031 and newer includes message and register counters. To view the counters, launch PCCU and go to **Entry mode** > **IEC Basic** > **ISaGRAF** > **Timings** tab.

The Msgs/sec column shows the number of messages transferred between the IsaVM (ISaGRAF interpreter) and Totalflow.exe processes. More messages means a higher CPU load. To reduce CPU load, use the ISaGRAF I/O Device to send 64 registers per message.

Figure 59: Message timing



Typographical conventions

Element	Convention	Example
Cross-reference to a figure or table in the document	Hyperlink to the figure or table in blue, with underline.	See Figure 2.
Cross-reference to a specific section in the document	Hyperlinks to sections referenced throughout the document appear in blue, with underline.	See Section 2.
Cross-reference to another document or website	Hyperlink to the website appears in blue, with underline	See the RMC user manual at abb.com.
Greater-than character (>)	Indicates that the following item is an additional menu selection	From the menu, locate and select **Calibrate** > **Diff. Press. Sensor** > **Calibration Units** > **Edit**.
Name of selection buttons, menus, or navigation tree items in instructions that the user will click	Bold text, and the capitalization agrees with the name as displayed on the user interface	Click the **Monitor** tab and select the **Add Advanced Setup** tab.
Programs, including utility and accessory programs	Title capitalization	Microsoft Word
URL	All lowercase for a fully specified URL	www.abb.com/totalflow
User input	Bold and lowercase, unless case sensitive. If the user-input string contains placeholder text, the text is in small caps.	Type **config**. SMALL CAPS FOR PLACE HOLDERS

Contact us

ABB Inc.

Measurement & Analytics

Quotes: totalflow.inquiry@us.abb.com

Orders: totalflow.order@us.abb.com

Training: totalflow.training@us.abb.com

Support: upstream.support@us.abb.com

+1 800 442 3097 (opt. 2)

www.abb.com/upstream

Main Office - Bartlesville
7051 Industrial Blvd
Bartlesville, OK 74006
Ph: +1 918 338 4888

Kansas Office - Liberal
2705 Centennial Blvd
Liberal, KS 67901
Ph: +1 620 626 4350

Texas Office - Houston
3700 W. Sam Houston
Parkway S., Suite 600
Houston, TX 77042
Ph: +1 713 587 8000

Texas Office – Odessa
8007 East Business 20
Odessa, TX 79765
Ph: +1 432 272 1173

Texas Office – Pleasanton
150 Eagle Ford Road
Pleasanton, TX 78064
Ph: +1 830 569 8062