



Efinity[®] Debugger Tutorial

UG-EFN-TUTDBG-v2.0
November 2025
www.efinixinc.com



Contents

Introduction.....	3
Development Board Support.....	3
Automated Debugging Flow.....	3
Prepare the Tutorial Files.....	4
Create a Debug Profile.....	4
Program the FPGA.....	5
Run the Debugger.....	5
Manual Debugging Flow.....	6
Prepare the Tutorial Files.....	6
Create a Debug Profile.....	7
Add Debug Code to Your Project.....	8
Program the FPGA.....	9
Run the Debugger.....	9
Combined Manual and Automated Debugging Flow.....	11
Prepare the Tutorial Files.....	11
Create First Profile and Add Debug Code.....	11
Create Second Debug Profile, Compile, and Program.....	12
Run the Debugger.....	12
Multiple Manual Debugging Flow.....	13
Prepare the Tutorial Files.....	13
Compile and Program the FPGA.....	13
Run the Debugger.....	14
Use Another Board.....	14
Where to Learn More.....	15
Revision History.....	15

Introduction

The Efinity[®] software includes a hardware Debugger to probe signals in your FPGA design via the JTAG interface. The Debugger includes two debug cores:

- You use a manual flow and the Profile Editor to configure Virtual I/O (vio) cores.
- You can use a manual flow or the Debug Wizard's automated flow to configure Logic Analyzer (la) cores.

The following sections walk you through the Debugger's automated and manual flows.



Note: These tutorials assume that you have already installed the Efinity[®] software and USB driver for the board.

Development Board Support

The Efinity software v2025.2 and higher includes example design files that target multiple Efinix development boards. The Efinity project files target the Titanium Ti60 F225 Development Board by default, but you can easily use another board by:

- Changing the target family and FPGA in the Project Editor.
- Importing the interface design for another board with the provided Interface Settings File (.isf).

The .isf files are named according to which board they support. For example, **Ti60F225_kit.isf** is for the Ti60 F225 Development Board. Most boards are supported.

Automated Debugging Flow

This tutorial walks you through the Debugger automated flow using an example **helloworld** design. You add a Logic Analyzer debug core to your design and configure it using the Debug Wizard, and use the GTKWave waveform viewer to see the debug results. The instructions assume that you have working knowledge of the Efinity[®] software.

Tip: The following steps use the Ti60 F225 Development Board; however, the procedure for other Efinix boards is similar. Refer to [Use Another Board](#) on page 14 for instructions.

Prepare the Tutorial Files

In this step you set up your environment and copy the Debugger tutorial design to your working directory.

1. Run the Efinity setup script if you have not already done so:
 - Linux: `source <Efinity path>/bin/setup.sh`
 - Windows: `<Efinity path>\bin\setup.bat`
2. Copy the folder `<Efinity path>/examples/helloworld-dbg` to your working directory.
3. Connect the Ti60 F225 Development Board (or another board) to your computer using a USB cable.

Create a Debug Profile



Debug Wizard

In this task you add the Logic Analyzer debug core and configure it.

1. Open the **helloworld** project.
2. Synthesize the design. You do not need to do a full compile; the Debug Wizard only uses the post-map netlist.
3. Click the Debug Wizard icon in the main icon bar to launch it.
4. In the **Signals from** list, choose **Elaborated Netlist** to browse for signals in the pre-map netlist, or **Post-Map** to use signals from the post-map netlist.
5. Select the `led` and `counter` buses from the list on the left and use the `>>` button to move them to the right. Leave the **Probe Type** at the default, which is **DATA AND TRIGGER**.
6. Click **Next**. The wizard generates a debug profile.
7. Leave **Enable "Auto Instantiation"** turned on. This option enables the debug profile in your project. Click **Finish**.
8. The software prompts you to recompile. Click **OK**.
9. Perform a full compile (Run the complete flow from synthesis to bitstream).




Program the FPGA

-  Open Debugger
-  Select Image File
-  Refresh USB Target
-  Start Programming FPGA

You program the FPGA on the development board using these steps:

1. Click **Open Debugger** to launch the Debugger. The programming controls are in the Program box.
2. The Ti60 F225 Development Board (or other board) displays as the **USB Target**. If it does not, make sure that the board is connected to your computer and click **Refresh USB Targets**.
3. Click the **Select Image File** button.
4. Browse to the **outflow** directory and choose *<helloworld>.bit*.
5. Click **Start Programming**. The console displays programming messages.

Run the Debugger

-  Connect Debugger
-  Disconnect Debugger
-  Add Trigger Condition

After you program the FPGA with the design containing the debug core, you can run the Debugger to observe the values on the probed signals. In the Debugger:

1. Click **Connect Debugger**.
2. In the **Trigger Setup** tab, click Add Trigger Condition.
3. Select **led[5:0]** and click **OK**.
4. Specify a **Value** of 110111, which triggers when the LED output is 110111.
5. Click **Run**. The Debugger waits for the trigger and then captures data.
6. When the Debugger finishes, it automatically opens the waveform in GTKWave
7. Click Disconnect Debugger to stop the Debugger.

Manual Debugging Flow

This tutorial walks you through the Debugger manual flow using an example **helloworld** design. You add Logic Analyzer and Virtual I/O cores manually in the Debugger's Profile Editor perspective, and use the GTKWave waveform viewer to see the debug results. The instructions assume that you have working knowledge of the Efinity® software.

Tip: The following steps use the Ti60 F225 Development Board; however, the procedure for other Efinix boards is similar. Refer to [Use Another Board](#) on page 14 for instructions.



Note: If you are switching from automated debugging flow to manual debugging flow, you need to go to **File > Edit Project > Debugger tab** and turn off the **Debugger Auto Instantiation** option.

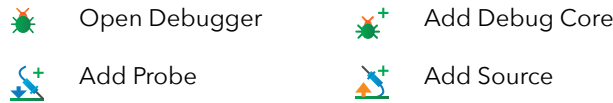
Prepare the Tutorial Files

In this step you set up your environment and copy the Debugger tutorial design to your working directory.

1. Run the Efinity setup script if you have not already done so:
 - Linux: `source <Efinity path>/bin/setup.sh`
 - Windows: `<Efinity path>\bin\setup.bat`
2. If you have not already done so, copy the folder `<Efinity path>/examples/helloworld-dbg` to your working directory.
3. Connect the Ti60 F225 Development Board (or another board) to your computer using a USB cable.

Tip: If you would like to use a project that is already set up manually (instead of doing all the steps yourself), copy the project in `<Efinity path>/examples/helloworld-dbg_GOLDEN`. Compile the project, and then skip to [Program the FPGA](#) on page 9.

Create a Debug Profile



In this task you add Virtual I/O and Logic Analyzer debug cores to a profile and configure them:

- You add the input signal(s) to control and the output signal(s) to observe to the Virtual I/O core.
- You add the wire, register, or signal to observe to Logic Analyzer core.

Remember to map the clock source after you instantiate the debug core.

1. Open the **helloworld** project in the **helloworld-dbg** directory.
2. Click **Open Debugger** to launch the Debugger. Because your project does not have a debug profile, the Debugger opens to the Profile Editor perspective.
3. Input a profile name into the **Profile name** field.
4. Click **Add Debug Core > Virtual I/O** to add a new core with the default **Core name (vio0)**.
5. Add three probes (in) and three sources (out) with the following name and width settings (leave the other settings at the defaults):

Name	Type	Width	Description
counter	Probe	29	Shows the values for counter[28:0].
raddr	Probe	4	Shows the values for raddr[3:0].
led	Probe	6	Shows the values for the pattern on led[5:0].
vio_reverse	Source	1	Control the reverse button.
vio_mux_sel	Source	1	Control the multiplexer select.
vio_maddr	Source	4	Control the memory address maddr[3:0].

6. Click **Add Debug Core > Logic Analyzer** to add a second core with the default **Core name (la0)**.
7. Add three probes (in) with the same name and width settings as the probes in the VIO core (leave the other settings at the defaults):

Name	Type	Width	Description
counter	Probe	29	Captures values in counter[28:0]
raddr	Probe	4	Captures the values in maddr[3:0]
led	Probe	6	Captures the values in led[5:0]

8. Click **Generate** (Generate Debug RTL). The Debugger creates the file `<profile name>_top.v` and template files (`<profile name>_TEMPLATE.v` and `<profile name>_TEMPLATE.vhd`) in your project directory.
9. Close the Debugger.

Add Debug Code to Your Project



Interface Designer



Create Block



Generate Interface Output Files

When you generate the debug code, the software copies the `<profile name>_top.v` file to your project directory. You need to add the file to your project, instantiate the RTL, and compile.

1. In the Efinity® main window, click the Project tab under the dashboard.
2. Right-click **Design** and choose **Add**.
3. Browse to your project directory.
4. Select the `<profile name>_top.v` and click **Open**.
5. In the Block Editor panel, add the JTAG User Tap block to the interface design.
 - a) Open the Interface Designer.
 - b) Select **JTAG User Tap**.
 - c) Click Create Block.
 - d) Choose **JTAG_USER1** as the **JTAG Resource**.
 - e) Click Generate Interface Output Files to generate SDC constraints.
 - f) Close the Interface Designer.
6. Double click the **helloworld.v** design to open the Code Editor. Edit the design to enable the debug code:
 - a) Add all of the JTAG input and output pins to the project top module (**helloworld**). The **helloworld_template.v** in the **outflow** directory has a template that you can copy and paste. Alternatively, you can copy and paste the following into the **helloworld** IO port module:

```
input jtag_inst1_CAPTURE,
input jtag_inst1_DRCK,
input jtag_inst1_RESET,
input jtag_inst1_RUNTEST,
input jtag_inst1_SEL,
input jtag_inst1_SHIFT,
input jtag_inst1_TCK,
input jtag_inst1_TDI,
input jtag_inst1_TMS,
input jtag_inst1_UPDATE,
output jtag_inst1_TDO,
```

- b) Declare the following probe signal and width in the **helloworld** module:

```
wire vio_reverse;
wire vio_mux_sel;
wire [AWIDTH-1:0] vio_maddr;
```

- c) Change hardware pushbutton to the `vio0` source

Old	New
<pre>assign raddr = counter[COUNTER_SIZE-1:DELAY_SIZE](vio_mux_sel)? vio_maddr :</pre>	<pre>assign raddr = counter[COUNTER_SIZE-1:DELAY_SIZE];</pre>

- d) Change hardware pushbutton to the `vio0` source

Old	New
else if(~reverse)	else if(~reverse vio_reverse)

- e) Instantiate the debug core in the project's top module. You can copy the example code from the generated `<debug profile name>_TEMPLATE.v` file; remember to replace `$INSERT_YOUR_CLOCK_NAME` with `clk`. You can also copy the following code into the project's top module (this code assumes that the debug profile is named `dbg_manual`):

```
dbg_manual_edb_top_dbg_manual_edb_top_inst (
    .bscan_CAPTURE ( jtag_inst1_CAPTURE ),
    .bscan_DRCK ( jtag_inst1_DRCK ),
    .bscan_RESET ( jtag_inst1_RESET ),
    .bscan_RUNTEST ( jtag_inst1_RUNTEST ),
    .bscan_SEL ( jtag_inst1_SEL ),
    .bscan_SHIFT ( jtag_inst1_SHIFT ),
    .bscan_TCK ( jtag_inst1_TCK ),
    .bscan_TDI ( jtag_inst1_TDI ),
    .bscan_TMS ( jtag_inst1_TMS ),
    .bscan_UPDATE ( jtag_inst1_UPDATE ),
    .bscan_TDO ( jtag_inst1_TDO ),
    .vio0_clk ( clk ),
    .vio0_counter ( counter ),
    .vio0_raddr ( raddr ),
    .vio0_led ( led ),
    .vio0_vio_reverse ( vio_reverse ),
    .vio0_vio_mux_sel ( vio_mux_sel ),
    .vio0_vio_maddr ( vio_maddr ),
    .la0_clk ( clk ),
    .la0_counter ( counter ),
    .la0_raddr ( raddr ),
    .la0_led ( led )
);
```

- f) Save the **helloworld.v** design.
7. Compile the design (run the complete flow from synthesis to bitstream).




Program the FPGA

-  Open Debugger
-  Select Image File
-  Refresh USB Target
-  Start Programming FPGA

You program the FPGA on the development board using these steps:

1. Click **Open Debugger** to launch the Debugger. The programming controls are in the Program box.
2. The Ti60 F225 Development Board (or other board) displays as the **USB Target**. If it does not, make sure that the board is connected to your computer and click **Refresh USB Targets**.
3. Click the **Select Image File** button.
4. Browse to the **outflow** directory and choose `<helloworld>.bit`.
5. Click **Start Programming**. The console displays programming messages.

Run the Debugger

-  Connect Debugger
-  Disconnect Debugger
-  Add Trigger Condition

After you program the FPGA with the design containing the debug core, you can run the Debugger to observe the values on the probed signals. In the Debugger:

1. Click Connect Debugger. The view opens to the **la0** tab.
2. In the **Trigger Setup** tab, click Add Trigger Condition.
3. Choose **led[5:0]** and click **OK**.
4. Specify a value of **110111**, which triggers when the LED output is 110111.
5. Click the **vio0** tab. The **Value** fields show the data captured on the probes. To reverse the LED blinking direction, change the **Value** for **vio_reverse** to **1** and press Enter. To stop the LEDs blinking, change the **Value** for **vio_mux_sel** to **1** and press Enter.
6. Click Disconnect Debugger to stop capturing data.

When the probe value is equal to the compared value, the cell background turns green to indicate a match. If the values do not match, the cell background turns red. By default, the compared values are an empty string, with the cell background color not set.

Figure 1: Example of Debugger with Comparator

The screenshot shows the Efinity Debugger interface. The main window is titled 'vio0' and contains a table of signals. The table has columns for Name, Type, Width, Radix, Value, Control, and Comparator. The signals listed are counter, raddr, led, vio_reverse, vio_mux_sel, and vio_maddr. The 'led' signal has a value of '03' and a comparator value of 'f0', which is highlighted in red. The 'vio_mux_sel' signal has a value of '0' and a comparator value of '0', which is highlighted in green. The 'vio_reverse' signal has a value of '0' and a comparator value of '0', which is highlighted in red. The 'vio_maddr' signal has a value of '0' and a comparator value of '0', which is highlighted in red. The 'counter' signal has a value of '293d8b9' and a comparator value of '293d8b9', which is highlighted in green. The 'raddr' signal has a value of 'a' and a comparator value of 'a', which is highlighted in green.

On the right side of the interface, there is a 'Configuration' panel with the following settings:

- USB Target: Quad RS232-HS
- USB Info: ID: 0403:6011
- Bitstream: buntu/home/phyung/efinity/helloworld.bit
- Device Status: Last Updated: Mon Sep 30 24 10:47:28
- Debugging: 0x10660a79

At the bottom of the interface, there is a 'Console' panel showing the following log output:

```

Mon September 30 24 10:47:17 - Board Profile: Generic Board
Profile Using FT4232
Mon September 30 24 10:47:18 - Valid device ID found:
0x10660A79
Mon September 30 24 10:47:18 - Board Profile: Generic Board
Profile Using FT4232
Mon September 30 24 10:47:18 - Using FTDI URL (SPI = ftdi://
0x0403:0x6011:0:1/1, JTAG = ftdi://0x0403:0x6011:0:2/1)
Mon September 30 24 10:47:19 - jtag programming started!
Mon September 30 24 10:47:19 - JTAG Programming on ftdi://
0x0403:0x6011:0:2/1
Mon September 30 24 10:47:19 - Programming '//wsL.localhost/
Ubuntu/home/phyung/efinity/helloworld.bit' via JTAG at freq
6.0 MHz
Mon September 30 24 10:47:19 - Device ID read from JTAG:
0x10660A79
Mon September 30 24 10:47:24 - ... finished with JTAG
programming
Mon September 30 24 10:47:25 - Detecting device status...
Mon September 30 24 10:47:28 - Device is in user mode!
Mon September 30 24 10:47:32 - Connecting to URL: ftdi://
0x0403:0x6011:0:2/1
Mon September 30 24 10:47:32 - Connecting to JTAG_TAP: efx_ti
  
```



Note: Debug profiles are paired to their respective generated debug core via a UUID key. Ensure that you select the correct debug profile when connecting to the debugger.

Combined Manual and Automated Debugging Flow

This tutorial describes how to use the manual and automated flows together, for example, if you want to use a simple probe and auto-generate the rest. The following sections describe how to set it up yourself. A completed example project, `helloworld-dbg_AUTO_AND_MANUAL`, is located in the `<Efinity path>/examples` directory.

Tip: The following steps use the Ti60 F225 Development Board; however, the procedure for other Efinix boards is similar. Refer to [Use Another Board](#) on page 14 for instructions.

Prepare the Tutorial Files

In this step you set up your environment and copy the Debugger tutorial design to your working directory. You use the same files as in the manual debugging flow.

1. Run the Efinity setup script if you have not already done so:
 - Linux: `source <Efinity path>/bin/setup.sh`
 - Windows: `<Efinity path>\bin\setup.bat`
2. Create a new working directory for the combined flow project.
3. Copy the folder `<Efinity path>/examples/helloworld-dbg` to your new working directory.
4. Connect the Ti60 F225 Development Board (or another board) to your computer using a USB cable.

Create First Profile and Add Debug Code

These procedures are the same as the manual flow.

1. [Create a Debug Profile](#) on page 7.
2. [Add Debug Code to Your Project](#) on page 8. However, **do not compile** yet.

Create Second Debug Profile, Compile, and Program



Debug Wizard

In this task you add the Logic Analyzer debug core and configure it. These steps are the same as the automated flow.

1. Synthesize the design. You do not need to do a full compile; the Debug Wizard only uses the post-map netlist.
2. Click the Debug Wizard icon in the main icon bar to launch it.
3. In the **Configuration Settings** > **JTAG USER TAP** drop-down list box, choose **USER2**. The manual profile you created earlier uses the **USER1** JTAG TAP.
4. In the **Signals from** list, choose **Elaborated Netlist** to browse for signals in the pre-map netlist, or **Post-Map** to use signals from the post-map netlist.
5. Select the `led` and `counter` buses from the list on the left and use the **>>** button to move them to the right. Leave the **Probe Type** at the default, which is **DATA AND TRIGGER**.
6. Click **Next**. The wizard generates a debug profile.
7. Leave **Enable "Auto Instantiation"** turned on. This option enables the debug profile in your project. Click **Finish**.
8. The software prompts you to recompile. Click **OK**.
9. Perform a full compile (Run the complete flow from synthesis to bitstream).
10. Follow the instructions in [Program the FPGA](#) on page 9.

Run the Debugger

USER1 - User Tap



Import Profile

With the two debug profiles configured and the FPGA programmed, you are ready to debug.

1. Debug with the manual debugger profile:
 - a) Change to **Perspectives** > **Profile Editor**.
 - b) Click **Import Profile**.
 - c) Choose the file **dbg_manual.json** and click **OK**.
 - d) Change to the Debugger using **Perspectives** > **Debug**.
 - e) Change the User TAP to **USER1**.
 - f) Connect the debugger.
 - g) Disconnect the debugger when finished.
2. Debug with the automated debugger profile:
 - a) Change to **Perspectives** > **Profile Editor**.
 - b) Click **Import Profile**.
 - c) Choose the file **debug_profile.wizard.json** and click **OK**.
 - d) Change to the Debugger using **Perspectives** > **Debug**.
 - e) Change the User TAP to **USER2**.
 - f) Connect the debugger.
 - g) Disconnect the debugger when finished.

Multiple Manual Debugging Flow

The Debugger supports flows with multiple manual debug profiles. For example, in a team environment, each user may need their own debug configuration. This tutorial explains how to use multiple manual debug profiles.

Tip: The following steps use the Ti60 F225 Development Board; however, the procedure for other Efinix boards is similar. Refer to [Use Another Board](#) on page 14 for instructions.

Prepare the Tutorial Files

In this step you set up your environment and copy the Debugger tutorial design to your working directory. You use the same files as in the manual debugging flow.

1. Run the Efinity setup script if you have not already done so:
 - Linux: `source <Efinity path>/bin/setup.sh`
 - Windows: `<Efinity path>\bin\setup.bat`
2. Create a new working directory for the combined flow project.
3. Copy the folder `<Efinity path>/examples/helloworld-dbg_MULTI_USERS` to your new working directory.
4. Connect the Ti60 F225 Development Board (or another board) to your computer using a USB cable.

This example has two debug profiles: `vio` and `la`.

Compile and Program the FPGA



Open Debugger



Select Image File



Refresh USB Target



Start Programming FPGA

You compile and then program the FPGA on the development board using these steps:

1. Open the helloworld project.
2. Perform a full compile.
3. Click **Open Debugger** to launch the Debugger. The programming controls are in the Program box.
4. The Ti60 F225 Development Board (or other board) displays as the **USB Target**. If it does not, make sure that the board is connected to your computer and click **Refresh USB Targets**.
5. Click the **Select Image File** button.
6. Browse to the **outflow** directory and choose `<helloworld>.bit`.
7. Click **Start Programming**. The console displays programming messages.

Run the Debugger

USER1 - User Tap



Import Profile

In this step you import the debug profiles and debug with them.

1. Debug with the `vio` manual debugger profile:
 - a) Change to **Perspectives > Profile Editor**.
 - b) Click **Import Profile**.
 - c) Choose the file **vio.json** and click **OK**. This profile uses JTAG User **USER1**.
 - d) Change to the Debugger using **Perspectives > Debug**.
 - e) Connect the debugger.
 - f) Disconnect the debugger when finished.
2. Debug with the `1a` debugger profile:
 - a) Change to **Perspectives > Profile Editor**.
 - b) Click **Import Profile**.
 - c) Choose the file **1a.json** and click **OK**. This profile uses JTAG User **USER2**.
 - d) Change to the Debugger using **Perspectives > Debug**.
 - e) Connect the debugger.
 - f) Disconnect the debugger when finished.

Use Another Board

Using another board is easy. First, change the device and family:

1. Choose **File > Edit Project**. The Project Editor dialog box opens.
2. Choose the device family in the **Family** drop-down list box.
3. Click **Select** next to **Device**. The **Device Selector** dialog box opens.
4. Choose the device on your board and click **OK**.

After you change the device, the software prompts you that the design has changed and you need to re-run the tool flow. Click **OK**. Then it displays the **Migrate Design** dialog box. You do not want to save the old interface:

1. Choose **Create New Design**.
2. Turn off the **Enable Design Backup** option.
3. Click **Next**.
4. The software gives a message that migration completed. Turn on the **Open Interface Design** option.
5. Click **Finish**. The Interface Designer opens.

Now you can import the correct interface design for your board:

1. Choose **File > Import Design**. The Import Design dialog box opens.
2. Choose **Interface Scripting File (.isf)** and click **Next**.
3. Click the file button to browse for the **.isf**.
4. Choose the **.isf** for your board.
5. Click **Finish**. The Interface Design loads the settings.
6. Save.

Where to Learn More

The Efinity® software includes documentation as PDF user guides and on-line HTML help. This documentation is provided with the software. You can also access the latest versions of PDF documentation in the [Support Center](#):

- [Efinity Software User Guide](#)
- [Efinity Software Installation User Guide](#)
- [Efinity Synthesis User Guide](#)
- [Efinity Timing Closure User Guide](#)
- [Efinity Trion Tutorial](#)
- [Efinity Debugger Tutorial](#)
- [Efinity Programmer User Guide](#)
- [Efinity IP Packager User Guide](#)
- [Trion Interfaces User Guide](#)
- [Titanium Interfaces User Guide](#)
- [Topaz Interfaces User Guide](#)
- [Efinity Interface Designer Python API](#)
- [Efinity Command-Line Interface User Guide](#)
- [Quantum® Trion Primitives User Guide](#)
- [Quantum® Titanium Primitives User Guide](#)
- [Quantum® Topaz Primitives User Guide](#)

In addition to documentation, Efinix field application engineers have created a series of videos to help you learn about aspects of the software. You can view these videos in the [Support Center](#).

Revision History

Table 1: Revision History

Date	Version	Description
November 2025	2.0	Tutorial now supports multiple Efinix development boards. (DOC-2715) Added steps for debugging with combined manual and automated profiles, as well as multiple manual profiles. (DOC-2715) Updated debug profile filenames and steps for 2025.2 release. (DOC-2615)
July 2025	1.5	Added icons and updated wording. Moved Run the Debugger on page 14 into an appendix.
March 2025	1.4	Notes added to Manual Debugging Flow on page 6 and Run the Debugger on page 9. (DOC-2232)
November 2024	1.3	Added paragraph explaining the functionality of the Comparator, as well as an example illustration. (DOC-2153)

Date	Version	Description
September 2024	1.2	Major revision of Add Debug Code to Your Project on page 8. (DOC-1848) Added new section: Run the Debugger on page 14. (DOC-1848) Rearranged the tables in topic Create a Debug Profile on page 7.
December 2022	1.1	Corrected the source and probe names for the manual flow's Run the Debugger topic.
August 2022	1.0	Initial release. The content in this tutorial originally appeared in the Efinity Trion Tutorial.