# Contents

1 Revision History ................................................................. 1
   1.1 Revision 3.0 ........................................................................ 1
   1.2 Revision 2.0 ........................................................................ 1
   1.3 Revision 1.0 ........................................................................ 1

2 PolarFire FPGA Temperature and Voltage Sensor .......................... 2
   2.1 Design Requirements ........................................................ 2
   2.2 Prerequisites ...................................................................... 3
   2.3 Demo Design ...................................................................... 3
      2.3.1 Design Implementation ............................................. 4
   2.4 Simulation Flow .................................................................. 6
      2.4.1 Simulating the Design ............................................... 6

3 Libero Design Flow ................................................................. 8
   3.1 Synthesize ......................................................................... 8
   3.2 Place and Route ............................................................... 9
      3.2.1 Resource Utilization ............................................... 9
   3.3 Verify Timing ................................................................... 9
   3.4 Generate FPGA Array Data ............................................ 9
   3.5 Generate Bitstream .......................................................... 9
   3.6 Run PROGRAM Action .................................................... 10

4 Running the Demo ................................................................. 11

5 Appendix 1: Programming the Device Using FlashPro Express .......... 12

6 Appendix 2: Running the TCL Script ........................................ 15
## Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>TVS Block Diagram</td>
<td>3</td>
</tr>
<tr>
<td>Figure 2</td>
<td>TVS Demo Design</td>
<td>4</td>
</tr>
<tr>
<td>Figure 3</td>
<td>TVS Configurator</td>
<td>5</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Simulation Options—Vsim Command</td>
<td>6</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Design Flow—Simulate</td>
<td>7</td>
</tr>
<tr>
<td>Figure 6</td>
<td>ModelSim Pro ME Wave Window</td>
<td>7</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Libero Design Flow Options</td>
<td>8</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Selecting COM Port and Connecting—Channel 0</td>
<td>11</td>
</tr>
<tr>
<td>Figure 9</td>
<td>FlashPro Express Job Project</td>
<td>12</td>
</tr>
<tr>
<td>Figure 10</td>
<td>New Job Project from FlashPro Express Job</td>
<td>13</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Programming the Device</td>
<td>13</td>
</tr>
<tr>
<td>Figure 12</td>
<td>FlashPro Express—RUN PASSED</td>
<td>14</td>
</tr>
</tbody>
</table>
## Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Design Requirements</td>
<td>2</td>
</tr>
<tr>
<td>Table 2</td>
<td>Temperature Channel Value Decoding</td>
<td>3</td>
</tr>
<tr>
<td>Table 3</td>
<td>Voltage Channel Value Decoding</td>
<td>4</td>
</tr>
<tr>
<td>Table 4</td>
<td>Resource Utilization</td>
<td>9</td>
</tr>
<tr>
<td>Table 5</td>
<td>Jumper Settings</td>
<td>10</td>
</tr>
</tbody>
</table>
1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 3.0
The following is a summary of the changes made in this revision.

• Added Appendix 2: Running the TCL Script, page 15.
• Updated Figure 2, page 4.
• Updated Figure 3, page 5.

1.2 Revision 2.0
The following is a summary of the changes made in this revision.

• Updated the document for Libero SoC v12.2.
• Removed the references to Libero version numbers.

1.3 Revision 1.0
The first publication of this document.
Each PolarFire device is equipped with a Temperature and Voltage Sensor (TVS). TVS reports die temperature and voltage of device supply rails in digital form to the FPGA fabric.

TVS is implemented using a 4-channel ADC and the channel information is given as follows:

- Channel 0 – 1 V voltage supply
- Channel 1 – 1.8 V voltage supply
- Channel 2 – 2.5 V voltage supply
- Channel 3 – Die temperature

The TVS outputs a 16-bit encoded value that represents voltage or temperature, and corresponding channel number. The temperature and voltage information is translated into standard temperature and voltage values. For more information, see UG0753: PolarFire FPGA Security User Guide.

This demo highlights the TVS feature of the PolarFire using an UART-based application (GUI). The demo design continuously pumps the data from TVS channels to UART, which is displayed on the GUI. This demo design also shows how to simulate the TVS feature of the PolarFire device.

The demo design can be programmed using any of the following options:

- Using the job file: To program the device using the job file provided along with the design files, see Appendix 1: Programming the Device Using FlashPro Express, page 12.
- Using Libero SoC: To program the device using Libero SoC, see Libero Design Flow, page 8. Use this option when the demo design is modified.

### 2.1 Design Requirements

The following table lists the hardware and software requirements for this demo design.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating system</td>
<td>64-bit Windows 7, 8, or 10</td>
</tr>
<tr>
<td>Hardware PolarFire Evaluation Kit (MPF300-EVAL-KIT)</td>
<td>Rev D or later</td>
</tr>
<tr>
<td>Software Libero SoC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ModelSim</td>
</tr>
<tr>
<td></td>
<td>FlashPro Express</td>
</tr>
</tbody>
</table>

**Note:** Libero SmartDesign and configuration screen shots shown in this guide are for illustration purpose only. Open the Libero design to see the latest updates.
2.2 Prerequisites

Before you begin:

1. For demo design files download link:
   http://soc.microsemi.com/download/rsc/?f=mpf_dg0852_df
2. Download and install Libero SoC (as indicated in the website for this design) on the host PC from the following location:
   https://www.microsemi.com/product-directory/design-resources/1750-libero-soc
   The latest versions of ModelSim, Synplify Pro, and FTDI drivers are included in the Libero SoC installation package.

2.3 Demo Design

The top-level block diagram of the TVS design is shown in the following figure. All four channels of TVS are enabled in the design to monitor the die temperature and voltage rails. The Fabric logic captures the TVS channels output and sends to UART IF through CoreUART IP.

![TVS Block Diagram](image)

The GUI receives channel wise TVS values and decodes as described to display them:

**Die Temperature:**

The temperature channels 16-bit output value is represented in Kelvin and can be decoded as listed in the following table. For example, the temperature channel's output value of 0x133B implies 307.56 Kelvin.

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Reserved</td>
</tr>
<tr>
<td>[14:4]</td>
<td>Integer value of temperature</td>
</tr>
<tr>
<td>[3:0]</td>
<td>Fractional value of temperature</td>
</tr>
</tbody>
</table>

![Temperature Channel Value Decoding](image)
Voltage:
The data present on the VALUE and CHANNEL outputs is valid only when the VALID output is asserted. When a channel is disabled by deasserting the corresponding channel enable input, then the channel data present on the outputs is not valid even if the VALID output is asserted. The voltage channels 16-bit output value is represented in millivolts (mV) and can be decoded as listed in the following table. For example, the voltage channels output value of 0x385E implies 1803.75 mV.

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Signed bit</td>
</tr>
<tr>
<td>[14:3]</td>
<td>Integer value of voltage</td>
</tr>
<tr>
<td>[2:0]</td>
<td>Fractional value of voltage</td>
</tr>
</tbody>
</table>

2.3.1 Design Implementation
The following figure shows the Libero SoC software design implementation of the TVS demo design.

Figure 2 • TVS Demo Design

The top-level design includes the following components:
- TVS_IP_0 Macro
- Core_UART_0
- TVS_to_UART_0 logic
- clock_gen_0
- INIT_MONITOR_0 and PF_RESET_0
2.3.1.1 TVS_IP_0 Macro

The following figure shows the TVS interface configurator.

Figure 3 • TVS Configurator

The GUI displays the die temperature in degree Celsius by converting Kelvin values.

Celsius value = Kelvin value - 273.15

2.3.1.2 TVS_to_UART_0

The TVS to UART logic captures the Temperature and Voltage values from the TVS macro and sends the data to Core_UART_0.

2.3.1.3 clock_gen_0

CCC is configured to generate the 100 MHz clock.
2.4 Simulation Flow

The TVS simulation model updates the outputs of the TVS macro based on reading instructions given in the .mem file or .txt file. The file name must be passed to the simulation model for the TVS outputs to toggle. The parameter used to store the .mem file name is called "TVS_MEMFILE". Add the following vsim command to pass the file name.

```
-gTVS_MEMFILE="PATH_TO_FILE_RELATIVE_TO_SIMULATION_FOLDER"
```

.MEM File Format

The following format of the file is in hex:

- \( \text{<simulation time (τ) in hexadecimal>} \)
- \( \text{<value of channel 0 at time τ>} \)
- \( \text{<value of channel 1 at time τ>} \)
- \( \text{<value of channel 2 at time τ>} \)
- \( \text{<value of channel 3 at time τ>} \)

The .mem file contains the simulation time followed by the digital values (16-bit) of the four ADC channels at that time instance. A value is required for the channel even if it is not used. The value can be 0. The simulation starts with all channel outputs being 0. The pattern can be repeated several times in the .mem file to reflect several values of the channel outputs. The content of the mem file is limited to 256 lines.

2.4.1 Simulating the Design

The Libero project includes a testbench to simulate the TVS block. The testbench captures all the four TVS channel values using CoreUART IP. The digital values for the four channels is passed through the .mem file.

2.4.1.1 Simulation Settings

To pass the mem file for simulation, perform the following steps:

1. Open the Libero SoC project settings (Project > Project Settings).
2. Select Vsim commands under the Simulation options. Enter `-gTVS_MEMFILE="tvs_values.mem"` in the Additional options field and then click Save.

A sample tvs_values.mem is provided in the simulation folder. The .mem file must be available in simulation folder of the Libero project. The tvs_values.mem file captures the 16-bit digital output of the TVS block at different time instances.

Figure 4 • Simulation Options—Vsim Command

To simulate the design, perform the following steps:
1. In the **Design Flow** tab, right-click **Simulate** under **Verify Pre-Synthesis Design** and then select **Open Interactively**.

   **Figure 5 • Design Flow—Simulate**

   ![Diagram of Design Flow—Simulate](image)

   When simulation is completed, the Wave window appears as shown in the following figure. Since all the four channels are enabled, TVS circuit outputs value of the four channels at a given point in time on the VALUE output along with channel number on the CHANNEL output. The data present on the VALUE and CHANNEL outputs are valid only when the VALID output is asserted. Observe the following from the simulation results:

   - After the channel is enabled for conversion, the TVS block takes 390 microseconds to complete the conversion.
   - Each channel has a conversion delay of 410 microseconds.
   - The conversion rate is equal to 1920 microseconds, which is same as the conversion rate set in the TVS configurator.
   - TVS block generate the output values based on the values given in the tvs_values.mem file.

   **Figure 6 • ModelSim Pro ME Wave Window**

   ![ModelSim Pro ME Wave Window](image)

2. Close ModelSim Pro ME and the Libero project.
This chapter describes the Libero design flow of the demo design. The Libero design flow involves the following steps:

- Synthesize
- Place and route
- Verify Timing
- Generate Bitstream
- Run PROGRAM Action

The following figure shows these options in the Design Flow tab.

**Figure 7 • Libero Design Flow Options**

### 3.1 Synthesize

To synthesize the design, perform the following steps:

1. From the Design Flow window, double-click Synthesize.

   When the synthesis is successful, a green tick mark appears as shown in Figure 7, page 8.

2. Right-click Synthesize and select View Report to view the synthesis report and log files in the Reports tab.
3.2 Place and Route

1. From the Design Flow window, double-click Place and Route. When place and route is successful, a green tick mark appears as shown in Figure 7, page 8.
2. Right-click Place and Route and select View Report to view the place and route report and log files in the Reports tab.

3.2.1 Resource Utilization

The following table lists the resource utilization of the design after place and route. These values may vary slightly for different Libero runs, settings, and seed values.

<table>
<thead>
<tr>
<th>Type</th>
<th>Used</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4LUT</td>
<td>269</td>
<td>299544</td>
<td>0.09</td>
</tr>
<tr>
<td>DFF</td>
<td>232</td>
<td>299544</td>
<td>0.08</td>
</tr>
<tr>
<td>I/O Register</td>
<td>0</td>
<td>510</td>
<td>0.00</td>
</tr>
<tr>
<td>Logic Elements</td>
<td>303</td>
<td>299544</td>
<td>0.10</td>
</tr>
</tbody>
</table>

3.3 Verify Timing

To verify timing, perform the following steps:

1. From the Design Flow window, double-click Verify Timing.
2. When the design successfully meets the timing requirements, a green tick mark appears as shown in Figure 7, page 8.
3. Right-click Verify Timing and select View Report to view the verify timing report and log files in the Reports tab.

3.4 Generate FPGA Array Data

To generate FPGA array data, double-click Generate FPGA Array Data from the Design Flow window. A green tick mark is displayed after the successful generation of the FPGA array data as shown in Figure 7, page 8.

3.5 Generate Bitstream

To generate the bitstream, perform the following steps:

1. Double-click Generate Bitstream from the Design Flow tab. When the bitstream is successfully generated, a green tick mark appears as shown in Figure 7, page 8.
2. Right-click Generate Bitstream and select View Report to view the corresponding log file in the Reports tab.
3.6 Run PROGRAM Action

After generating the bitstream, the PolarFire device must be programmed. To program the PolarFire device, perform the following steps:

1. Ensure that the following Jumper Settings are set on the board.

**Table 5 • Jumper Settings**

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>J18, J19, J20, J21, and J22</td>
<td>Short pin 2 and 3 for programming the PolarFire FPGA through FTDI</td>
<td>Closed</td>
</tr>
<tr>
<td>J28</td>
<td>Short pin 1 and 2 for programming through the on-board FlashPro5</td>
<td>Open</td>
</tr>
<tr>
<td>J26</td>
<td>Short pin 1 and 2 for programming through the FTDI SPI</td>
<td>Closed</td>
</tr>
<tr>
<td>J4</td>
<td>Short pin 1 and 2 for manual power switching using SW3</td>
<td>Closed</td>
</tr>
<tr>
<td>J12</td>
<td>Short pin 3 and 4 for 2.5 V</td>
<td>Closed</td>
</tr>
</tbody>
</table>

2. Connect the power supply cable to the J9 connector on the board.
3. Connect the USB cable from the Host PC to J5 (FTDI port) on the board.
4. Power on the board using the SW3 slide switch.

When the device is programmed successfully, a green tick mark appears as shown Figure 7, page 8.
4 Running the Demo

This chapter describes how to install and use the Graphic User Interface (GUI) to run the TVS demo. The PolarFire TVS demo application is a simple GUI that runs on the host PC to communicate with the PolarFire Device.

To install the GUI, perform the following steps:

1. Extract the contents of the `mpf_dg0852_df.rar` file. From the `mpf_dg0852_df\GUI\TVS_Monitor_GUI_Installer` folder, double-click the `setup.exe` file.
2. Follow the instructions displayed on the installation wizard.

After successful installation, `TVS_Monitor_GUI` appears on the Start menu of the host PC desktop.

To run the TVS demo, perform the following steps:

1. From the Start menu, click `TVS_Monitor_GUI` to launch the application. Ensure that the board is connected and appropriate Log Folder is selected.
2. Click Connect. On successful connection, the GUI shows the temperature and voltage values. Log file is created with time stamp in the file name at the Log Folder location. By default, Log Folder points to the 'SupportFiles' folder in the installation directory. User can modify the Log Folder location before connecting to the board.

   **Note:** Ensure that the Log Folder is not a system restricted location. In this case, user is required to launch the GUI with admin privileges (right-click and run as admin).

3. Upper Limit, Lower Limit, and the minimum variation to log for each of the channels are configurable in the setup.ini file. Channel values are logged in the log file if there is a variation exceeding the specified 'min var' values in the setup.ini file.

The following figure shows the standard temperature and voltage values of channel 0 (1.05 V). The plot corresponds to the values of Channel 0. Similarly, select the other channels and view their corresponding values and plots.

**Figure 8 • Selecting COM Port and Connecting—Channel 0**

![TVS Monitor GUI](image)

**Note:** The GUI updates the TVS channel values with the delay entered in the Delay (ms) field.
Appendix 1: Programming the Device Using FlashPro Express

This section describes how to program the PolarFire device with the .job programming file using FlashPro Express. The .job file is available at the following design files folder location:

`mpf_dg0852_df\Programming_Job`

To program the device, perform the following steps:

1. Ensure that the jumper settings on the board are the same as listed in Table 5, page 10.
   
   **Note:** The power supply switch must be switched off while making the jumper connections.

2. Connect the power supply cable to the J9 connector on the board.
3. Connect the USB cable from the Host PC to the J5 (FTDI port) on the board.
4. Power on the board using the SW3 slide switch.
5. On the host PC, launch the FlashPro Express software.
6. Click **New** or select **New Job Project from FlashPro Express Job** from **Project** menu to create a new job project, as shown in the following figure.

*Figure 9 • FlashPro Express Job Project*
7. Enter the following in the New Job Project from FlashPro Express Job dialog box:
   - **Programming job file:** Click Browse, navigate to the location where the .job file is located, and select the file. The default location is: `<download_folder>\mpf_dg0852_df\Programming_Job`.
   - **FlashPro Express job project location:** Click Browse and navigate to the location where you want to save the project.

*Figure 10 • New Job Project from FlashPro Express Job*

8. Click OK. The required programming file is selected and ready to be programmed in the device.

9. The FlashPro Express window appears as shown in the following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click Refresh/Rescan Programmers.

*Figure 11 • Programming the Device*
10. Click **RUN** to program the device. When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in the following figure. See **Running the Demo**, page 11 to run the TVS demo.

**Figure 12 • FlashPro Express—RUN PASSED**

11. Close **FlashPro Express** or in the **Project** tab, click **Exit**.
Appendix 2: Running the TCL Script

TCL scripts are provided in the design files folder under directory TCL_Scripts. If required, the design flow can be reproduced from Design Implementation till generation of job file.

To run the TCL, follow the steps below:

1. Launch the Libero software
2. Select Project > Execute Script....
3. Click Browse and select script.tcl from the downloaded TCL_Scripts directory.
4. Click Run.

After successful execution of TCL script, Libero project is created within TCL_Scripts directory.

For more information about TCL scripts, refer to mpf_dg0852_df/TCL_Scripts/readme.txt.

Refer to Libero® SoC TCL Command Reference Guide for more details on TCL commands. Contact Technical Support for any queries encountered when running the TCL script.