**Description**

The objective of the TIDA-00980 reference design is to develop an embedded firmware to interface and control the functions of a multi-segment RGB signal tower used in factory floor and industrial process automation of greater complexity. This design helps to give more flexible usage along with a sequence of terminal commands.

External actions or controls of hardware activate the embedded firmware functions. This embedded firmware specifically has communication links to other devices such as the RGB LED Signal Tower for Industrial Automation Reference Design (TIDA-00979) [1] and the IO-Link Sensor Transmitter BoosterPack (TIDA-00339) [2] stacked on the MSP-EXP430FR4133 LaunchPad™ Development Kit for functionality or to address a requirement to adjust the application device.

The embedded firmware enables fully flexible control of color, speed, brightness, and rows of TLC5971 LED drivers considered as segments.

**Features**

- Flexible and Easy-to-Control RGB LED Tower Light Design
- Different Switchable Modes: Demo, Temperature, Humidity, and IO-Link
- Controllable RGB Color, Brightness, Row, and Blinking Speed
- One to Five RGB LED Segments With Four Individual Channels Each
- Adjustable Maximum LED Current up to 60 mA per Channel

**Applications**

- Factory Automation and Control
- Building Automation

---

**Resources**

- TIDA-00980 Design Folder
- TIDA-00979 Design Folder
- TIDA-00339 Design Folder
- TLC5971 Product Folder
- LMZ35003 Product Folder
- MSP430F5528 Product Folder
- HDC1080 Product Folder
- TPD2E001 Product Folder
- MSP430FR4133 Product Folder
- SN65HVD102 Product Folder
- MSP-EXP430FR4133 Tool Folder

---

An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.
1 Key System Specifications

Table 1 shows the key system specifications.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>SPECIFICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power supply</td>
<td>$V_{IN_{\text{MIN}}} = 18 , \text{V}$</td>
</tr>
<tr>
<td></td>
<td>$V_{IN_{\text{TYP}}} = 24 , \text{V}$</td>
</tr>
<tr>
<td></td>
<td>$V_{IN_{\text{MAX}}} = 36 , \text{V}$</td>
</tr>
<tr>
<td>Interfaces</td>
<td>IO-Link</td>
</tr>
<tr>
<td></td>
<td>USB (CDC)</td>
</tr>
<tr>
<td>LED</td>
<td>60 RGB LEDs at 20 mA</td>
</tr>
<tr>
<td></td>
<td>Max 60 mA for each channel</td>
</tr>
<tr>
<td></td>
<td>Brightness control</td>
</tr>
<tr>
<td></td>
<td>Configured as 3×20 array</td>
</tr>
</tbody>
</table>
2 System Description

Industrial signal lights indicate the status of manufacturing equipment or the status of processes in industrial environments. These lights are often called stack lights, tower lights, or indicator lights and contain one to five different colors. The IEC 60073 standard shows how the colors correspond to different states (see Table 2).

<table>
<thead>
<tr>
<th>COLOR</th>
<th>SAFETY MEANING</th>
<th>CONDITION OF A PROCESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RED</td>
<td>Danger</td>
<td>Emergency or fault</td>
</tr>
<tr>
<td>AMBER</td>
<td>Warning</td>
<td>Abnormal</td>
</tr>
<tr>
<td>GREEN</td>
<td>Safe</td>
<td>Normal</td>
</tr>
<tr>
<td>BLUE</td>
<td>Mandatory significance</td>
<td></td>
</tr>
<tr>
<td>WHITE</td>
<td>No specific meaning assigned</td>
<td></td>
</tr>
</tbody>
</table>

This TI Design provides a flexible solution based on RGB light-emitting diodes (LEDs) and a modular approach. The solution has four printed-circuit boards (PCBs) as Figure 1 shows:

- Power and control board (TIDA-00979PW)
- LED driver with RGB LEDs (TIDA-00979LD)
- IO-Link BoosterPack™ Plug-in Module (TIDA-00339)
- MSP430FR4133 LaunchPad™ Development Kit

Five of the mentioned LED boards are daisy chained and connected to one control board. This control board runs a stack light control software and communicates through a serial peripheral interface (SPI) with the MSP430FR4133 LaunchPad.

The LED driver requires an external power supply of 18 V to 36 V. The microcontroller (MCU) is externally powered through the IO-Link interface or USB. Two different interfaces can be used for controlling the function:

- IO-Link through IO-Link Booster Pack
- USB CDC Device

Different demo functions are implemented:

- Single color stack light with configurable color and brightness
- Blinking in configurable color and brightness
- Multicolor stack light
- Temperature visualization
- Humidity visualization
- Run light
- Level light
3 Block Diagram

Figure 1 shows the block diagram of this design.

Figure 1. TIDA-00980 Block Diagram
3.1 **Highlighted Products**

The TIDA-00980 reference design features the TIDA-00979, TIDA-00339, as well as the MSP430FR4133 LaunchPad.

For more information on each of these devices, see their respective product folders at [www.ti.com](http://www.ti.com).

3.1.1 **TIDA-00979**

The TIDA-00979 provides a flexible solution based on RGB light-emitting diodes (LEDs) and a modular approach. The solution has two-printed circuit boards (PCBs) as Figure 2 shows.

![TIDA-00979 Block Diagram](image.png)

**Figure 2. TIDA-00979 Block Diagram**

This design is used for driving a stack light consisting of 60 RGB LEDs organized in a 3x20 array. A software is built to communicate through SPI with TIDA-00339 with a host PC connected through USB or to display a different demo pattern.

The MCU on this design (MSP430F5528) has an integrated USB 2.0 PHY that is used to implement a USB CDC device, enabling control of the design without the requirement of an IO-Link master.

The TIDA-00339 is used through an SPI connection to communicate with an IO-Link master.

For further detail on this design, refer to the product folder.
3.1.2 TIDA-00339 and MSP430FR4133 LaunchPad™

The TIDA-00339 TI Design is a fully IO-Link compliant design enabling the user to easily evaluate the IO-Link communication. The modular approach is capable of use with different MCUs based on the LaunchPad and BoosterPack ecosystem and also allows the user to test their own sensor front end.

Figure 3 shows the block diagram of TIDA-00339 with the connection to the MSP430FR4133 LaunchPad. This setup is used for handling the IO-Link communication and has a SPI connection with the MCU on the TIDA-00979 board.
4 System Design Theory
This TI Design consists of two different designs which are used together. This section describes the connection between them and the software design.

4.1 System Overview
Figure 4 shows the connection between the two designs and provides a short board description.

Table 3 provides details on the connection wiring between the boards. An IO-Link stack runs on the MSP430FR4133, which also implements a SPI master for communicating with other devices. Therefore, a SPI slave is implemented on the MSP430F5528 on the TIDA-00979 board. The SPI connection consists of the usual signal (CS, CLK, SIMO, SOMI) as well as some additional wires for issuing a reset and an acknowledge line.

Furthermore, the power for the MSP430F5528 is delivered from the IO-Link board. Check the correct jumper setting for the power supply shown in Table 4 and Table 5 to avoid damage to the boards.

Table 3. Connection Between TIDA-00979 and TIDA-00339

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>TIDA-00979</th>
<th>TIDA-00339</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3 V</td>
<td>J3 - 1</td>
<td>J12 - 1</td>
</tr>
<tr>
<td>P1.0 ACK</td>
<td>J3 - 3</td>
<td>J12 - 3</td>
</tr>
<tr>
<td>P2.7 UCA0_CLK</td>
<td>J3 - 13</td>
<td>J12 - 13</td>
</tr>
<tr>
<td>GND</td>
<td>J4 - 2</td>
<td>J10 - 2</td>
</tr>
<tr>
<td>P1.1 /CS</td>
<td>J4 - 6</td>
<td>J10 - 6</td>
</tr>
<tr>
<td>P3.3 UCA0_SIMO</td>
<td>J4 - 12</td>
<td>J10 - 12</td>
</tr>
<tr>
<td>P3.4 UCA0_SOMI</td>
<td>J4 - 14</td>
<td>J10 - 14</td>
</tr>
<tr>
<td>P6.5 GPIO5</td>
<td>J4 - 15</td>
<td>J10 - 10</td>
</tr>
</tbody>
</table>
Table 4. Jumper Setting on TIDA-00979

<table>
<thead>
<tr>
<th>JUMPER</th>
<th>SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>J8</td>
<td>Open</td>
</tr>
<tr>
<td>J9</td>
<td>Short 1 - 2</td>
</tr>
<tr>
<td>J11</td>
<td>Short 1 - 2</td>
</tr>
</tbody>
</table>

Table 5. Jumper Settings of TIDA-00339

<table>
<thead>
<tr>
<th>JUMPER</th>
<th>SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>Short 1 - 2</td>
</tr>
<tr>
<td>J2</td>
<td>Short 1 - 2</td>
</tr>
<tr>
<td>J4</td>
<td>Short 2 - 3</td>
</tr>
<tr>
<td>J6</td>
<td>Short 1 - 2</td>
</tr>
<tr>
<td>J7</td>
<td>Short 1 - 2</td>
</tr>
<tr>
<td>Other</td>
<td>Open</td>
</tr>
</tbody>
</table>

4.2 Software

The software running on the MSP430FR4133 is supplied by TMG (Technologie Management Gruppe), whereas the software running on the MSP430F5528 has been adapted to implement different stack light features such as:

- Single-color stack light
- Blinking stack light
- Multicolor stack light
- Temperature visualization
- Humidity visualization
- Run light
- Level light

Two different interfaces are implemented for controlling the function: IO-Link (through the TIDA-00339) and USB (as a CDC device, so it is enumerated as a serial port). Different modes are additionally implemented, which can be switched by pressing S2:

- Demo mode: Switching between different patterns
- Temperature visualization mode: Displaying the current temperature
- Humidity mode: Displaying the measured humidity
- IO-Link mode: The desired pattern and configuration is set through IO-Link or USB

The following Section 4.2.1 explains the implementation of the different features in detail.

4.2.1 Demo Mode

The application starts up in a default demo mode that cycles between different patterns and is implemented in the file Mode.c in the MODE_DEMO() function. The main loop has to run without blocking statements, such as delays, otherwise the USB stack may cause problems.

Six patterns are implemented in total:

- All on: In this mode all LEDs are set to the same color and brightness. The function pattern1() anticipates the number of LEDs, the RGB color as eight-bit values, as well as the brightness up to a value of 128. This function is non-blocking and can be called once to set a new color, brightness, or number of LEDs that are turned ON.
- Blink: The blink mode is implemented in function pattern2(). The number of LEDs, the blink interval (the time depends on the timer ISR configuration), as well as the color and brightness must be passed. This function is non-blocking and must be called every iteration of the main loop. Controlling the state of the LEDs is done in the ISR; this function just configures this state and turns the LEDs
corresponding with the state that is set in the ISR ON or OFF.

- Stack light: This mode is implemented in function pattern3(). The segment, as well as the brightness, must be given as arguments for the function call. This function is non-blocking and must be called whenever something changes.
- Temperature visualization: This mode is implemented in pattern4(). The number of LEDs as well as the brightness must be given. This function is non-blocking and must be called whenever something changes.
- Run light: A run light pattern is implemented in function pattern5(). The parameters are the range as well as the RGB color and the brightness. This function is non-blocking and has an internal state, so it must be called within the main loop to work properly.
- Level light: A level is displayed using the LEDs in this mode. Function pattern6() handles this action and the parameters are the same as for the temperature visualization.

Table 6 provides an overview of the implemented pattern and the parameters. R (red), G (green), and B (blue) represents an eight-bit value for the corresponding color. BC is a global brightness value that corresponds with the current of the LEDs. A seven-bit value can be passed here.

<table>
<thead>
<tr>
<th>FUNCTION NAME</th>
<th>DISPLAY</th>
<th>ARGUMENTS</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern1</td>
<td>All LEDs ON</td>
<td>Number of LEDs, R, G, B, BC</td>
<td>Call once</td>
</tr>
<tr>
<td>Pattern2</td>
<td>Blinking LEDs</td>
<td>Number of LEDs, Interval, R, G, B, BC</td>
<td>Call regularly</td>
</tr>
<tr>
<td>Pattern3</td>
<td>Stack light (red, amber, green, blue, white)</td>
<td>Segment, BC</td>
<td>Call once</td>
</tr>
<tr>
<td>Pattern4</td>
<td>Temperature visualization</td>
<td>Value, BC</td>
<td>Call once</td>
</tr>
<tr>
<td>Pattern5</td>
<td>Run light</td>
<td>Number of LEDs, R, G, B, BC</td>
<td>Call regularly</td>
</tr>
<tr>
<td>Pattern6</td>
<td>Level light</td>
<td>Value, BC</td>
<td>Call once</td>
</tr>
</tbody>
</table>

### 4.2.2 Temperature and Humidity Mode

By pressing S2, the application switches in temperature mode and the D19 LED lights up. In this mode, the current ambient temperature is measured by an HDC1080 device on the PCB of TIDA-00979 and is visualized using the temperature pattern. The default range is configured to display a range from 15°C to 30°C. This range is configured with temp_min and temp_max in the source code and can either be changed in the source code or through IO-Link or USB.

By pressing S2 a second time, D18 lights up and humidity is visualized the same way using the level pattern. The default range here is from 0% to 100% and can be configured by setting level_min and level_max either in the source code or through IO-Link or USB.

The function handling both modes is implemented in MODE_I2C() and is non-blocking.

#### 4.2.2.1 HDC1080 Sensor Data

The HDC1080 is a digital humidity sensor with an integrated temperature sensor that provides excellent measurement accuracy at low power. The HDC1080 operates over a wide supply range and is a low-cost, low-power alternative to competitive solutions in a wide range of common applications. The humidity and temperature sensors are factory calibrated.

The HDC1080 sensor has an I^2C interface with a seven-bit address of 0x40. The device is a low-speed, multi-slave serial bus with a typical frequency of 100 kHz to 400 kHz. Figure 5 shows a typical bus connection with one master and two slaves. Pullup resistors are necessary due to the open drain design.
The scope shot in Figure 6 shows a typical transmission for the HDC1080 sensor. This example shows the transmission for configuring the sensor and starting a measurement. A start is sent to address 0x40 with the write bit set, followed by the register (0x02) that should be written and the 16-bit data (0x1000).

The next transmission starts a conversation by issuing a write to register 0x0 without data. After a certain time for the conversation, the results can be read from register 0x0.
The temperature value is a 16-bit value with the least significant bit (LSB) set to 0 because the resolution is 14 bit. The formula from the datasheet is simplified to avoid floating point operations.

The following code shows how this simplification is done: First copy the relevant data to a temporary 32-bit variable. Then the value is multiplied and the division is replaced by a bit-shift operation. The last line calls the function displaying the pattern with the scaled measurement. The variable temp contains the temperature in °C.

```c
long temp;
temp = (reg_data[0]<<<8)|reg_data[1] & 0xffff;
temp *= 165;
temp >>= 16;
temp -= 40;
pattern4(20*(temp-temp_min)/(temp_max - temp_min), DEFAULT_BC);
```

The same procedure can be done for the humidity value:

```c
temp = ((reg_data[2]<<<8)|reg_data[3]) & 0xffff;
temp *= 100;
temp >>= 16;
pattern6(20*(temp-level_min)/(level_max - level_min), DEFAULT_BC);
```

### 4.2.3 IO-Link and USB Mode

After pressing S2 a third time, the IO-Link or USB mode is entered and the D18 and D19 LEDs light up. In this mode, commands can be sent to control the different modes either through IO-Link or through USB.

The function MODE_SPI() handles the SPI communication and sets the pattern depending on the received process data.

#### 4.2.3.1 IO-Link

After entering SPI mode, the IO-Link stack is initialized by calling IOLSPISAPI_Init() and must be polled regularly by calling IOLSPISAPI_Task(). For details about the SPI communication, refer to the TIDA-00339 product folder for a referenced example.

The stack then calls the IOLSPISAPI_CbfNewOutputs() function and the new 7-byte process data is copied to the pdOut variable whenever it changes.

These seven bytes are presented as three variables in software: an 8-bit wide pdOut.PDout8 that defines the operation mode as shown in Table 7, a 16-bit wide variable pdOut.PDOut16 that contains process data, and a 32-bit wide variable that contains configuration data. Whenever a pattern is displayed, the necessary parameters are set through pdOut16 as Table 8 shows. These bits are only relevant when PDOut8[5:4] is set to 0. The meaning of the 32 bits depends on the mode that is selected, see Table 9 for details.

<table>
<thead>
<tr>
<th>BITS</th>
<th>FUNCTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0 (On/Off)</td>
<td>0: Off</td>
</tr>
<tr>
<td></td>
<td>1: Pattern On</td>
</tr>
<tr>
<td>Bit 3 - 1 (Pattern)</td>
<td>000: Pattern1 - All LEDs on</td>
</tr>
<tr>
<td></td>
<td>001: Pattern2 - Blinking</td>
</tr>
<tr>
<td></td>
<td>010: Pattern3 - Stack Light (fixed color)</td>
</tr>
<tr>
<td></td>
<td>011: Pattern4 - Temperature (fixed color)</td>
</tr>
<tr>
<td></td>
<td>100: Pattern5 - Run Light</td>
</tr>
<tr>
<td></td>
<td>101: Pattern6 - Level Light (fixed color)</td>
</tr>
<tr>
<td>Bit 5 - 4 (Mode)</td>
<td>00: Normal Mode</td>
</tr>
<tr>
<td></td>
<td>01: Configure Temperature Range</td>
</tr>
<tr>
<td></td>
<td>10: Configure Level Range</td>
</tr>
</tbody>
</table>
### Table 8. pdOut.PDOut16

<table>
<thead>
<tr>
<th>PATTERN</th>
<th>BITS 15 TO 8</th>
<th>BITS 7 TO 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern1</td>
<td>—</td>
<td>Number of LEDs</td>
</tr>
<tr>
<td>Pattern2</td>
<td>Interval</td>
<td>Number of LEDs</td>
</tr>
<tr>
<td>Pattern3</td>
<td>—</td>
<td>Segment number</td>
</tr>
<tr>
<td>Pattern4</td>
<td>Temperature MSB</td>
<td>Temperature LSB</td>
</tr>
<tr>
<td>Pattern5</td>
<td>—</td>
<td>Number of LEDs</td>
</tr>
<tr>
<td>Pattern6</td>
<td>Level MSB</td>
<td>Level LSB</td>
</tr>
</tbody>
</table>

### Table 9. pdOut.PDOut32

<table>
<thead>
<tr>
<th>MODE</th>
<th>BITS 31 TO 24</th>
<th>BITS 23 TO 16</th>
<th>BITS 15 TO 8</th>
<th>BITS 7 TO 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Red value</td>
<td>Green value</td>
<td>Blue value</td>
<td>Brightness</td>
</tr>
<tr>
<td>Configure level</td>
<td>level_max[15:8]</td>
<td>level_max[7:0]</td>
<td>level_min[7:0]</td>
<td>level_min[7:0]</td>
</tr>
</tbody>
</table>

A range of 15°C to 30°C is predefined for the temperature. In level mode, the default range is from 0 to 100.

When pattern3 (temperature) or pattern6 (level) has been selected, the range of the input value can be set in the corresponding configuration mode and is scaled for display.

Table 10 shows some examples.

### Table 10. Example Commands

<table>
<thead>
<tr>
<th>PROCESS DATA</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 00 14 FF FF FF 01</td>
<td>Pattern1</td>
</tr>
<tr>
<td>07 00 14 00 00 00 01</td>
<td>Pattern4</td>
</tr>
<tr>
<td>20 00 00 01 00 00 00</td>
<td>Set Level Range</td>
</tr>
</tbody>
</table>

#### 4.2.3.2 USB

USB is also an option in addition to sending data through IO-Link. This feature enables the possibility to test the application without requiring an IO-Link master. A USB CDC stack is implemented, so the device enumerates as a serial port. A connection can be established with any serial terminal program at any baudrate.

The commands that can be sent are similar to the IO-Link commands. Before sending a block of data, a ‘d’ must be sent, followed by the process data in 8-bit blocks with a space in between and a line break (‘\r’) at the end. One example of a command is:

```
d 01 00 14 ff ff ff 01
```

This command turns ON all LEDs in white with minimal brightness.

Independent from where the process data pdOut has been sent, the command is parsed in Pattern_call() and the configured pattern is displayed or a configuration value is changed.
5 Getting Started

To begin, use the following instructions.
1. Verify the jumper settings (see Table 4 and Table 5).
2. Connect five TIDA-00979LD boards together (J2 to J11).
3. Connect J2 of the TIDA-00979LD board to J7 of the TIDA-00979PW board.
4. Connect TIDA-00979PW to TIDA-00339 as described in Table 3.
5. Connect J3 of TIDA-00339 to an IO-Link master.
6. Provide 24 V to J3 of TIDA-00979PW.

The TIDA-00979 runs in demo mode after the previous five steps have been complete. Connect the programming tool to J10 of TIDA-00979PW to change the firmware.

5.1 Necessary Equipment

The following list shows the instruments and equipment required for a basic test setup.
- 1x TIDA-00979PW
- 5x TIDA-00979LD
- 1x TIDA-00339
- 1x MSP430FR4133 LaunchPad
- IO-Link Master (this design uses the TMG – USB IO-Link Master V2 SE)
- A PC with:
  - A USB interface
  - An MSP430™ MCU programmer
  - Code Composer Studio™ software
  - Serial terminal software, such as PuTTY or TeraTerm
- Power supply of 18 V to 36 V

5.2 Software Installation

Refer to the user manual of the USB IO-Link master in use for further details on its software installation.
The IO-Link stack for TIDA-00339 can be found in the corresponding product folder.
The source files for the TIDA-00979 can be found in the product folder of this design. Import the files into Code Composer Studio, compile them, and then flash them to the MSP430F5528 device. After programming, the demo mode starts automatically.

5.3 Software Usage

The application starts in demo mode, switching between the different patterns described in Table 6.
By pressing S2, the demo mode can be switched to temperature mode, after which the D19 LED lights up.
The ambient temperature displays in a range of 15°C to 30°C.
Pressing S2 a second time switches to humidity mode and lights up D18.
Pressing S2 a third time causes both LEDs to light up and the design waits for data through IO-Link or USB.

5.3.1 IO-Link Data

In this example, the TMG IO-Link Master V2 - SE tool is used to send process data to the design. Figure 7 shows this example. The used sequence (01 00 14 ff ff ff 01) turns ON all LEDs in color and with minimal brightness as Figure 8 shows.
Figure 7. TMG IO-Link Tool Example Sequence
Figure 8. Showing Pattern 1 With All LEDs ON
5.3.2 USB Data

A USB can be used in the same way as an IO-Link. The device enumerates as a serial port; in this case, it is COM12. Connect to the device (using the PuTTY software, for example) at any baudrate. When connected, type a set of process data with a 'd' placed in front and hit the ENTER key. Figure 9 shows an example input for temperature display with a value of 20°C and minimal brightness.

![Figure 9. USB Connection to TIDA-00980](image)

After sending this set of data, six rows of LEDs light up, as Figure 10 shows.

![Figure 10. Temperature Displaying 20°C](image)
6 Design Files

6.1 Schematics
To download the schematics, see the design files at TIDA-00980.

6.2 Bill of Materials
To download the bill of materials (BOM), see the design files at TIDA-00980.

6.3 PCB Layout Recommendations

6.3.1 Layout Prints
To download the layout prints, see the design files at TIDA-00980.

6.4 Altium Project
To download the Altium project files, see the design files at TIDA-00980.

6.5 Gerber Files
To download the Gerber files, see the design files at TIDA-00980.

6.6 Assembly Drawings
To download the assembly drawings, see the design files at TIDA-00980.

7 Software Files
To download the software files, see the design files at TIDA-00980.

8 Related Documentation


8.1 Trademarks
LaunchPad, BoosterPack, MSP430, Code Composer Studio are trademarks of Texas Instruments.
IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ("TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT. AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include: without limitation, TI's standard terms for semiconductor products [http://www.ti.com/sc/docs/stdterms.htm], evaluation modules, and samples [http://www.ti.com/sc/docs/sampterms.htm].

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2017, Texas Instruments Incorporated