Application Note

SLG46824/6 MTP Arduino Programming Example

AN-CM-255

Abstract

In this application note, we use the Arduino MTP Programmer sketch to program an SLG46824/6. Through analyzing the code, a firmware designer can create a modified version that is compatible with their unique microcontroller.  
  
This application note comes complete with design files which can be found in the References section.

<Do not remove this section break!>

Contents

[Abstract 1](#_Toc1748865)

[Contents 2](#_Toc1748866)

[Figures 2](#_Toc1748867)

[1 Terms and Definitions 3](#_Toc1748868)

[2 References 3](#_Toc1748869)

[3 Introduction 4](#_Toc1748870)

[4 Arduino-GreenPAK Connections 5](#_Toc1748871)

[5 Exporting GreenPAK NVM Data from a GreenPAK Design File 6](#_Toc1748872)

[6 Use the Arduino Sketch 10](#_Toc1748873)

[7 Programming Tips and Best Practices 10](#_Toc1748874)

[7.1 Executing Precise 16-Byte NVM Page Writes or change to Deviations from the valid command structure?: 10](#_Toc1748875)

[7.2 Transferring NVM Data into the Matrix Configuration Registers 10](#_Toc1748876)

[7.3 Resetting the I2C Address after an NVM Erase: 11](#_Toc1748877)

[8 Errata Discussion 11](#_Toc1748878)

[9 Conclusion 11](#_Toc1748879)

[Revision History 12](#_Toc1748880)

Figures

[Figure 1. Arduino Connections 5](#_Toc1749891)

[Figure 2. Simple GreenPAK Design in a SLG46826 6](#_Toc1749892)

[Figure 3. Export NVM 7](#_Toc1749893)

[Figure 4. Save as .hex File 7](#_Toc1749894)

[Figure 5. Viewing the NVM Data in Notepad++ 7](#_Toc1749895)

[Figure 6. Arduino Sketch 8](#_Toc1749896)

[Figure 7. Set EEPROM Data 8](#_Toc1749897)

[Figure 8. EEPROM Data Editor 9](#_Toc1749898)

[Figure 9. Matrix Registers, NVM, and EEPROM Protection Settings 9](#_Toc1749899)

[Figure 10. Arduino Serial Monitor 10](#_Toc1749900)

[Figure 11: ACK Behavior Modification to the Arduino Programmer 11](#_Toc1749901)

**Tables**

[Table 1: Arduino Uno / GreenPAK Connections 5](#_Toc523906999)

# Terms and Definitions

EEPROM Electrically erasable programmable read-only memory

Define acronyms and abbreviations used in the document. Use camel case for acronym definitions.

I2C Inter-integrated circuit

MTP Multiple-time programmable

NVM Non-volatile memory

OTP One-time programmable

# References

For related documents and software, please visit:

<https://www.dialog-semiconductor.com/configurable-mixed-signal>.

Download our free GreenPAK Designer software [1] to open the .gp files [2] and view the proposed circuit design. Use the GreenPAK development tools [3] to freeze the design into your own customized IC in a matter of minutes. Dialog Semiconductor provides a complete library of application notes [4] featuring design examples as well as explanations of features and blocks within the Dialog IC.

1. [GreenPAK Designer Software](https://www.dialog-semiconductor.com/greenpak-designer-software), Software Download and User Guide, Dialog Semiconductor

1. [AN-CM-255 SLG46824/6 MTP Arduino Programming Eample.gp](https://www.dialog-semiconductor.com/sites/default/files/an-cm-255_slg468246_mtp_arduino_programming_example.zip), GreenPAK Design File, Dialog Semiconductor

1. [GreenPAK Development Tools](https://www.dialog-semiconductor.com/greenpak-development-tools), GreenPAK Development Tools Webpage, Dialog Semiconductor

1. [GreenPAK Application Notes](https://www.dialog-semiconductor.com/greenpak-application-notes), GreenPAK Application Notes Webpage, Dialog Semiconductor

List the documents that are relevant for readers of the document.

1. [In-System Programming Guide](https://www.dialog-semiconductor.com/sites/default/files/isp_guide_slg46824_26_rev.1.1.pdf), GreenPAK User Guides and Manuals, Dialog Semiconductor

# Introduction

In this application note, we show how to use the SLG46824/6 Arduino programming sketch to program a Dialog SLG46824/6 GreenPAK™ Multiple-Time Programmable (MTP) device.

Most GreenPAK devices are One-Time Programmable (OTP), meaning that once their Non-Volatile Memory bank (NVM) is written, it cannot be overwritten. GreenPAKs with the MTP feature, like the SLG46824 and SLG46826, have a different type of NVM memory bank that can be programmed more than once.

We’ve written an [Arduino sketch](https://www.arduino.cc/) that allows the user to program an MTP GreenPAK with a few simple serial monitor commands. In this application note we use an SLG46826 as our GreenPAK with MTP.

**We provide sample code for the Arduino Uno using an open-source platform based on C/C++. Designers should extrapolate the techniques used in the Arduino code for their specific platform.**

**For specific information regarding** I2C **signal specifications,** I2C **addressing, and memory spaces, please reference the** GreenPAK In-System Programming Guide **provided on the** [SLG46826 product page](https://www.dialog-semiconductor.com/products/slg46826)**. This application note provides a simple implementation of this programming guide.**

# Arduino-GreenPAK Connections

To program the NVM of our SLG46826 GreenPAK with our Arduino sketch, we'll first need to connect four Arduino Uno pins to our GreenPAK. You can connect these pins directly to the GreenPAK Socket Adapter or to a breakout board with the GreenPAK soldered down.

Table 1: Arduino Uno / GreenPAK Connections

|  |  |
| --- | --- |
| GreenPAK | Arduino |
| VDD (Pin 1) | Digital Pin 2 |
| GND (Pin 11) | GND |
| SCL (Pin 8) | A5 |
| SDA (Pin 9) | A4 |

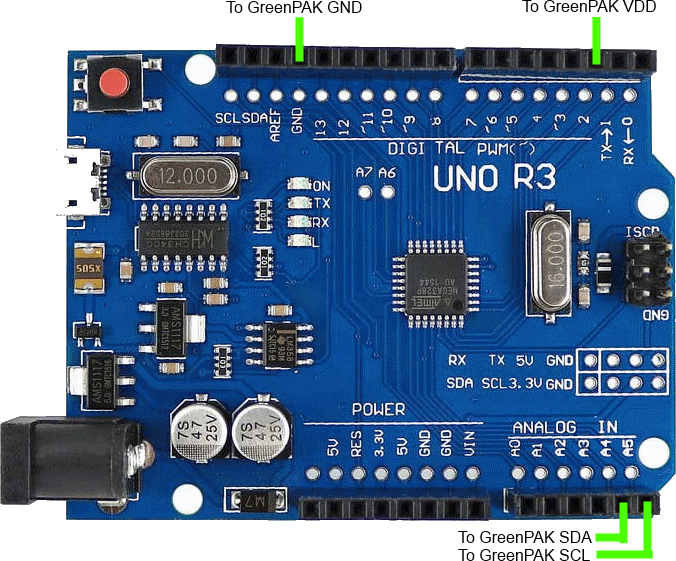


Figure 1. Arduino Connections

Please note that external I2C pull up resistors are not shown in Figure 1. Please connect a 4.7 kΩ pull up resistor from both SCL and SDA to the Arduino’s 3.3 V output.

# Exporting GreenPAK NVM Data from a GreenPAK Design File

We'll put together a very simple GreenPAK design to illustrate how to export the NVM data. The design below is a simple level shifter where the blue pins on the left are tied to VDD (3.3v), while the yellow pins on the right are tied to VDD2 (1.8v).

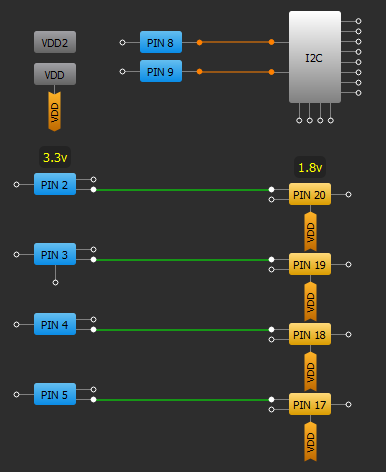


Figure 2. Simple GreenPAK Design in a SLG46826

To export the information from this design, you need to select File → Export → Export NVM, as shown in Figure 3.

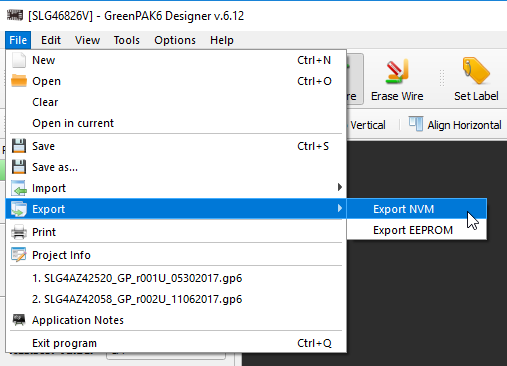


Figure 3. Export NVM

You will then need to select Intel HEX Files (\*.hex) as the file type and save the file.

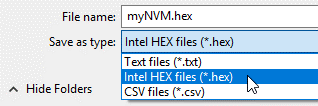


Figure 4. Save as .hex File

Now, you'll need to open the .hex file with a text editor (like Notepad++). To learn more about the Intel’s HEX file format and syntax, check out its [Wikipedia page](https://en.wikipedia.org/wiki/Intel_HEX). For this application we’re only interested in the data portion of the file as shown in Figure 5.

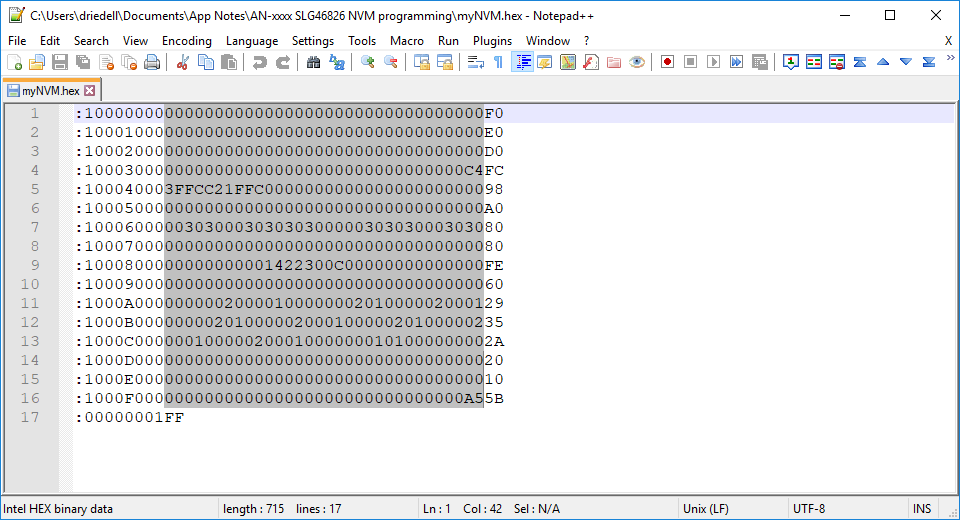


Figure 5. Viewing the NVM Data in Notepad++

Highlight and copy the 256 bytes of NVM configuration data located within the HEX file. Each line that we are copying is 32 characters long, which corresponds to 16 bytes.

Paste the information into the highlighted nvmString[] section of the Arduino sketch as shown in Figure 6. If you’re using a non-Arduino Microcontroller, you could write a function to parse the nvmData saved in the GreenPAK .GP6 file. (If you open a GreenPAK file with a text editor, you’ll see that we store project information in an easily-accessible XML format.)

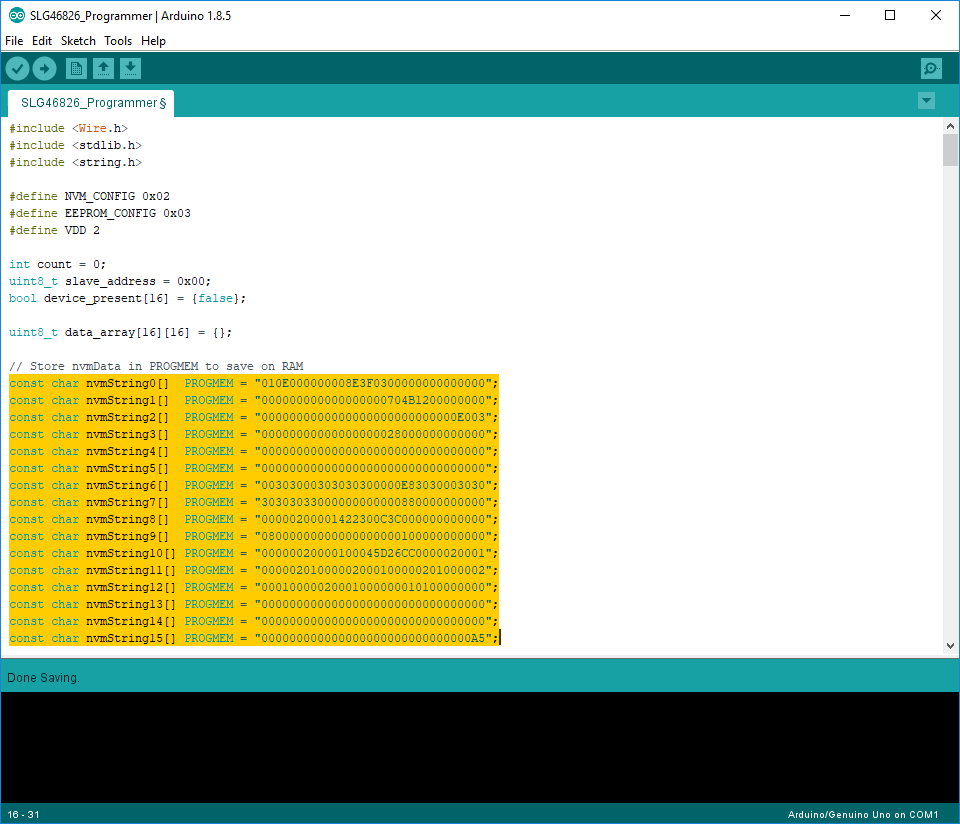


Figure 6. Arduino Sketch

To set the EEPROM data for your GreenPAK design, select the EEPROM block from the components panel, open its properties panel, and click "Set Data."

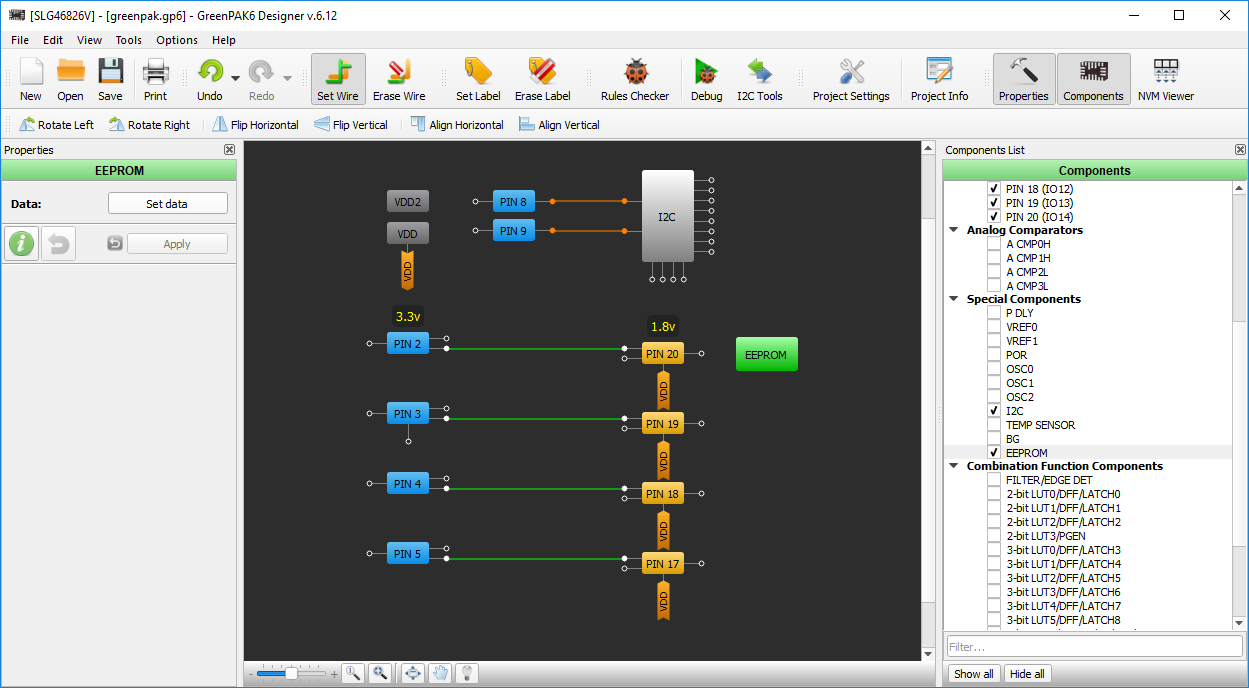


Figure 7. Set EEPROM Data

Now you can edit each byte in the EEPROM individually with our GUI interface.

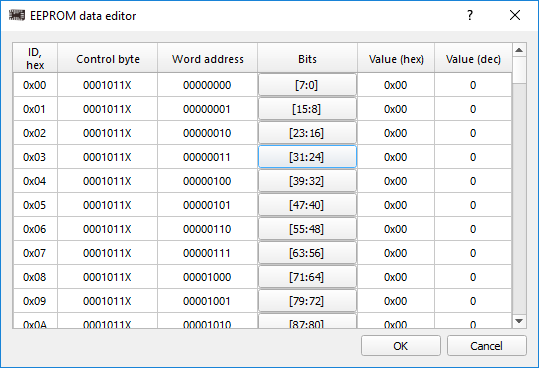


Figure 8. EEPROM Data Editor

Once your EEPROM data is set, you can export it to a HEX file using the same method described previously for exporting the NVM data. Insert these 256 bytes of EEPROM data into the eepromString[] section of the Arduino sketch.

For each custom design, it is important to check the protection settings within the “Security” tab of the project settings. This tab configures the protection bits for the matrix configuration registers, the NVM, and the EEPROM. Under certain configurations, uploading the NVM sequence can lock the SLG46824/6 to its current configuration and remove the MTP functionality of the chip.

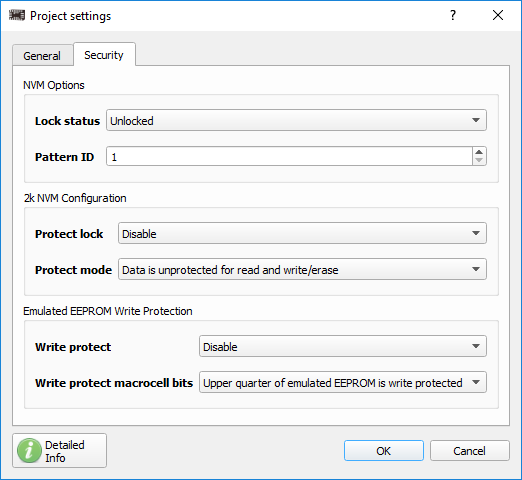


Figure 9. Matrix Registers, NVM, and EEPROM Protection Settings

# Use the Arduino Sketch

Upload the sketch to your Arduino and open the serial monitor with a 115200 baud rate. Now you can use the sketch's MENU prompts to perform several commands:

* Read - reads either the device’s NVM data or EEPROM data using the specified slave address
* Erase - erases either the device’s NVM data or EEPROM data using the specified slave address
* Write - Erases and then writes either the device’s NVM data or EEPROM data using the specified slave address. This command writes the data that is saved in the nvmString[] or eepromString[] arrays.
* Ping - returns a list of device slave addresses that are connected to the I2C bus

The results of these commands will be printed to the serial monitor console.

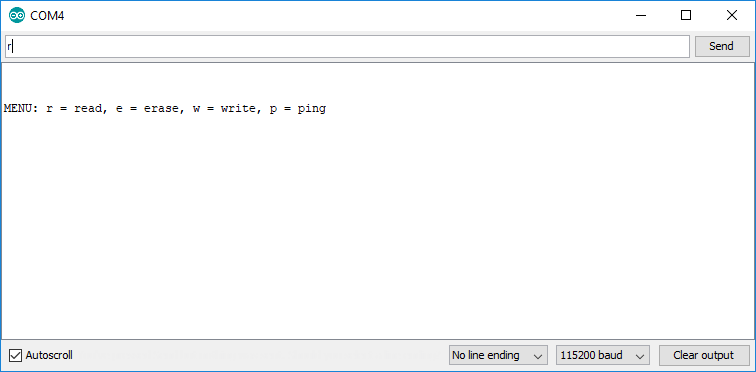


Figure 10. Arduino Serial Monitor

# Programming Tips and Best Practices

Over the course of supporting the SLG46824/6, we’ve documented a few programming tips to help avoid common pitfalls associated with erasing and writing to the NVM address space. The following subsections outline this topic in more detail.

## Executing Precise 16-Byte NVM Page Writes:

When writing data to the SLG46824/6’s NVM, there are three techniques to avoid:

* Page writes with less than 16 bytes
* Page writes with more than 16 bytes
* Page writes that don’t begin at the first register within a page (IE: 0x10, 0x20, etc.)

If any of the above techniques are used, the MTP interface will disregard the I2C write to avoid loading the NVM with incorrect information. We recommend performing an I2C read of the NVM address space after writing to verify correct data transfer.

## Transferring NVM Data into the Matrix Configuration Registers

When the NVM is written, the matrix configuration registers are not automatically reloaded with the newly written NVM data. The transfer must be initiated manually by cycling the PAK VDD or by generating a soft reset using I2C. By setting register <1601> in address 0xC8, the device re-enables the Power-On Reset (POR) sequence and reloads the register data from the NVM into the registers.

## Resetting the I2C Address after an NVM Erase:

When the NVM is erased, the NVM address containing the I2C slave address will be set to 0000. After the erase, the chip will maintain its current slave address within the configuration registers until the device is reset as described above. Once the chip has been reset, the I2C slave address must be set in address 0xCA within the configuration registers each time the GreenPAK is power-cycled or reset. This must be done until the new I2C slave address page has been written in the NVM.

# Errata Discussion

When writing to the “Page Erase Byte” (Address: 0xE3), the SLG46824/6 produces a non-I2C compliant ACK after the “Data” portion of the I2C command. This behavior might be interpreted as a NACK depending on the implementation of the I2C master.

To accommodate for this behavior, we modified the Arduino programmer by commenting out the code shown in Figure 11. This section of code checks for an I2C ACK at the end of every I2C command in the eraseChip() function. This function is used to erase the NVM and EEPROM pages. Since this section of code is located in a For loop, the “return -1;” line causes the MCU to pre-maturely exit the function.

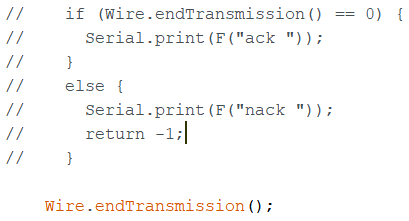


Figure 11: ACK Behavior Modification to the Arduino Programmer

Despite the presence of a NACK, the NVM and EEPROM erase functions will execute properly. For a detailed explanation of this behavior, please reference “Issue 2: Non-I2C Compliant ACK Behavior for the NVM and EEPROM Page Erase Byte” in the SLG46824/6 errata document (Revision XC) on [Dialog’s website](https://www.dialog-semiconductor.com/products/slg46826).

# Conclusion

In this application note we describe the process of using the provided Arduino programmer to upload custom NVM and EEPROM strings to a GreenPAK IC. The code in the Arduino Sketch is thoroughly commented, but if you have any questions regarding the sketch, please contact one of our Field Application Engineers or post your question on our [forum](https://support.dialog-semiconductor.com/forums/greenpak-and-greenfet-questions). For more in-depth information regarding MTP programming registers and procedures, please reference Dialog’s [In-System Programming Guide](https://www.dialog-semiconductor.com/products/slg46826) (provided in the Resources section at the link).

# Revision History

Summarize the changes made to the document. The latest revision is shown at the top of the Revision History table. Optionally show change details for the latest revision in a separate table row, with hyperlinks to the related sections.

|  |  |  |
| --- | --- | --- |
| Revision | Date | Description |
| 1.1 | 25-Feb-2019 | Modified Arduino script to accommodate for SLG46824/6 (XC Revision) errata. Discussion added in Section 8 of this AN. |
| 1.0 | 05-Sep-2018 | Initial Version |

**Approvals**

This section should be completed by the author. As a style of hidden text, it does not appear in the final document. Please refer to section 6.1 of [CQD1008](http://dms/sites/Forms/QualityTemplates/CQD1008.docx) for further guidance.

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Section(s) Reviewed** | **Name** | **Date of Approval** |
| ASSP Apps or Customer Support | <Section number(s)> | <Name> | <DD-Mmm-YYYY> |
| Marketing or Systems Engineer | <Section number(s)> | <Name> | <DD-Mmm-YYYY> |
| Technical Writer | <Section number(s)> | <Name> | <DD-Mmm-YYYY> |
| TPL (optional) | <Section number(s)> | <Name> | <DD-Mmm-YYYY> |
| System Architect (optional) | <Section number(s)> | <Name> | <DD-Mmm-YYYY> |
| Software Engineer (optional) | <Section number(s)> | <Name> | <DD-Mmm-YYYY> |

Status Definitions

|  |  |
| --- | --- |
| Status | Definition |
| DRAFT | The content of this document is under review and subject to formal approval, which may result in modifications or additions. |
| APPROVED or unmarked | The content of this document has been approved for publication. |

Disclaimer

Information in this document is believed to be accurate and reliable. However, Dialog Semiconductor does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information. Dialog Semiconductor furthermore takes no responsibility whatsoever for the content in this document if provided by any information source outside of Dialog Semiconductor.

Dialog Semiconductor reserves the right to change without notice the information published in this document, including without limitation the specification and the design of the related semiconductor products, software and applications.

Applications, software, and semiconductor products described in this document are for illustrative purposes only. Dialog Semiconductor makes no representation or warranty that such applications, software and semiconductor products will be suitable for the specified use without further testing or modification. Unless otherwise agreed in writing, such testing or modification is the sole responsibility of the customer and Dialog Semiconductor excludes all liability in this respect.

Customer notes that nothing in this document may be construed as a license for customer to use the Dialog Semiconductor products, software and applications referred to in this document. Such license must be separately sought by customer with Dialog Semiconductor.

All use of Dialog Semiconductor products, software and applications referred to in this document are subject to Dialog Semiconductor’s [Standard Terms and Conditions of Sale](http://www.dialog-semiconductor.com/standard-terms-and-conditions), available on the company website ([www.dialog-semiconductor.com](http://www.dialog-semiconductor.com)) unless otherwise stated.

Dialog and the Dialog logo are trademarks of Dialog Semiconductor plc or its subsidiaries. All other product or service names are the property of their respective owners.

|  |  |  |  |
| --- | --- | --- | --- |
| Contacting Dialog Semiconductor | | | |
| United Kingdom (Headquarters)  Dialog Semiconductor (UK) LTD  Phone: +44 1793 757700  Germany  Dialog Semiconductor GmbH  Phone: +49 7021 805-0  The Netherlands  Dialog Semiconductor B.V.  Phone: +31 73 640 8822 | North America  Dialog Semiconductor Inc.  Phone: +1 408 845 8500  Japan  Dialog Semiconductor K. K.  Phone: +81 3 5769 5100  Taiwan  Dialog Semiconductor Taiwan  Phone: +886 281 786 222 | Hong Kong  Dialog Semiconductor Hong Kong  Phone: +852 2607 4271  Korea  Dialog Semiconductor Korea  Phone: +82 2 3469 8200 | China (Shenzhen)  Dialog Semiconductor China  Phone: +86 755 2981 3669  China (Shanghai)  Dialog Semiconductor China  Phone: +86 21 5424 9058 |
| Email:  enquiry@diasemi.com | Web site:  www.dialog-semiconductor.com |  |  |

© 2019 Dialog Semiconductor. All rights reserved.