Device Driver User's Guide

ADIN2111

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

INTRODUCTION

This document describes the ADIN2111 software device drivers and their usage with the target evaluation board.

The ADIN2111 is a low power, store and forward 2-port Ethernet Switch with integrated 10BASE-T1L PHYs designed for line or daisy chain networks.

This revision of the user's guide documents version 1.0.0 of the software drivers.

.

DRIVER OVERVIEW

The ADIN2111 device driver software provides support for LES features such as sending and receiving frames, program destination address filter table, dynamic forwarding table, link bringup, link status query, etc.

The ADIN2111 may be used with different host processors and operating systems. To accommodate the different applications, the ADIN2111 software driver is implemented in a platform-independent fashion, where the access to the underlying hardware is done via a HAL API.

The ADN2111 driver partitioning is shown in the picture below.

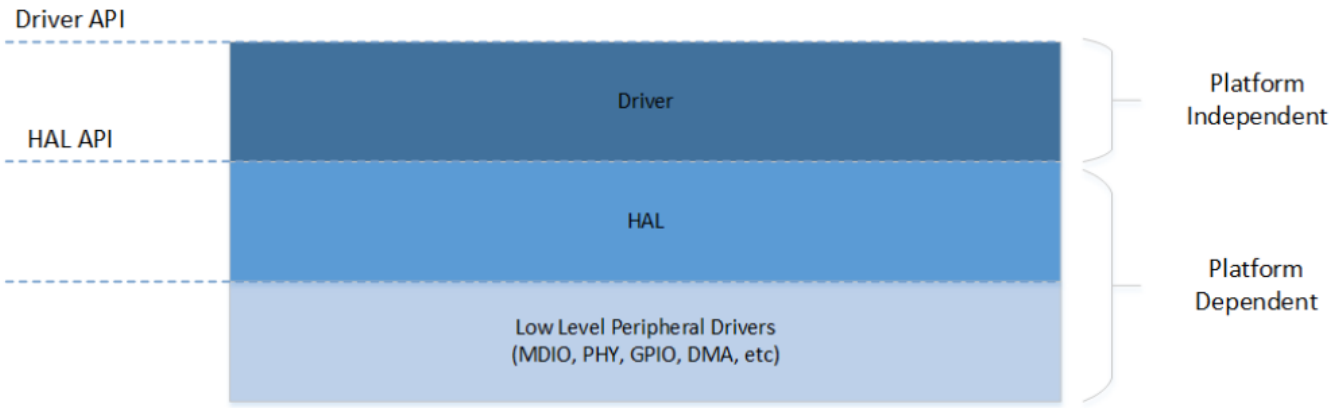

Figure 1. Driver Partitioning

TABLE OF CONTENTS

REVISION HISTORY

11/17—Revision 0.1.0: Initial Version

07/09 – Revision 0.2.0: Update for U2 support, OPEN Alliance SPI

08/18 – Revision 0.3.0: Clarify usage of SyncConfig API

12/06 – Revision 1.0.0: Remove references to S1 silicon. Add memory footprint information

GETTING STARTED

The ADIN2111 device driver is delivered as an installer, containing all the source code and documentation described in this document.

After running the installer the following structure is created in the installation directory:

- **inc**: ADIN2111 driver header files
- **src**: ADIN2111 driver sources
- **doc**: documentation, including this user's guide document
- **common**: includes the HAL API definition and ports to supported target platforms
- **examples/adin2111**: ADIN2111 examples
- **examples/bsp**: board support software

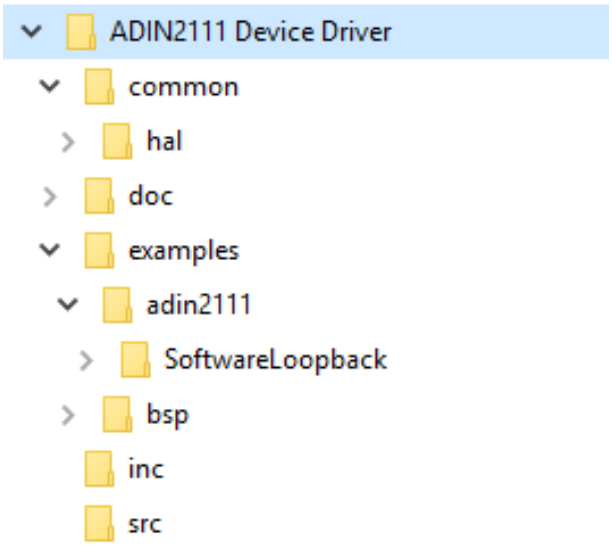

Figure 2. Directory Structure

This user's guide is found in the **doc** directory. In addition, the **doc** directory also contains the driver API reference manual in HTML format.

SOFTWARE REQUIREMENTS

The following tools and version are used for development and testing of the software driver and example code:

- IAR Embedded Workbench version 8.32.4 (www.iar.com)
- STM32CubeL4 MCU package version 1.16.0 (https://www.st.com/en/embedded-software/stm32cubel4.html)

Additional platforms-specific software requirements may be detailed in the platforms section later in this document.

DRIVER USAGE

The ADIN2111 driver implements 2 buffer queues: one for transmit and one for receive. When the application wants to transmit a frame over the 10BASE-T1L link, it submits a buffer descriptor to the Tx queue. When no previous buffers are left in the queue and there is no other SPI traffic between the host and the ADIN2111, the frame is sent to the ADIN2111 Tx FIFO and, if configured, a callback is used to notify the application that frame has been downloaded and the buffer can be reused or discarded.

On the receive side, the application should submit buffer descriptors in advance of enabling incoming traffic, for example by bringing up the link. When a new frame is available in the ADIN2111 Rx FIFO, the driver will read the frame and populate a buffer descriptor previously submitted by the application. The application is notified when a buffer descriptor from the Rx queue is available via a callback.

The following sequence of driver API calls shows a typical usage of the driver:

1. adin2111_Init(): driver initialization, including initialization of the Tx and Rx queues, MAC reset, interrupt masking, etc. When initialization is complete, the PHY is in software powerdown (no link) with auto-negotiation enabled.
2. Device configuration: this includes the configuration required before the link is brought up and communication over the 10BASE-T1L link is enabled. It is recommended to submit at least one buffer descriptor to the Rx queue (adin2111_SubmitRxBuffer()). Other examples of configuration steps that may be performed here are populating the destination address filter table or PHY settings that need to be performed in software powerdown.
3. adin2111_Enable(): brings both PHYs out of software powerdown and attempts to establish the link.
4. adin2111_GetLinkStatus(): check if the link is up
5. Normal operation: the application submits frames to the Tx queue (adin2111_SubmitTxBuffer()) and to the Rx queue (adin2111_SubmitRxBuffer()) as required to handle the communication needs
6. adin2111_Disable(): stops the operation of the driver by putting both PHYs in software powerdown, counterpart to adin2111_Enable()
7. adin2111_UnInit(): uninitializes the device, counterpart to adin2111_Init().

More details of the driver APIs used above can be found in the **ADIN2111 Device Driver API Reference Manual** HTML document, installed in the same directory as the user's guide,

The example project "SoftwareLoopback" shows an implementation of the steps described above.

BUFFER DESCRIPTORS

The driver uses buffer descriptors for both transmit (adin2111_SubmitTxBuffer) and receive (adin2111_SubmitRxBuffer) operations. The buffer descriptor type is adi_eth_BufDesc_t and defined in adi_mac.h, documented in the **ADIN2111 Device Driver API Reference Manual** HTML document.

Note that the size of the buffer needs to be at least 4 bytes greater than the actual payload. This is because of the Frame Check Sequence (FCS) appended to the frame:

- On transmit, the driver can be configured to calculate FCS, which is then appended to the buffer prior to the write to the ADIN2111
- On receive, the ADIN2111 will always include the FCS in the frame read by the host, hence the receive buffer needs to be sized accordingly. Note that the actual frame size (trxSize) does not include the extra 4 FCS bytes

DRIVER STRUCTURE

The file hierarchy of the ADIN2111 driver is shown in the figure below:

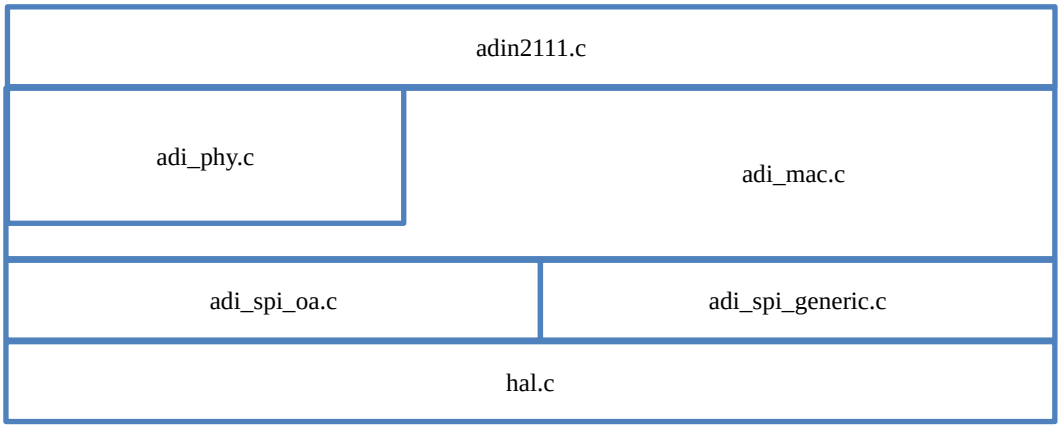

Figure 3 ADIN1110 driver hierarchy

The user API is implemented in adin2111.c, and uses functions defined at PHY level (adi_phy.c) and MAC level (adi_mac.c).

Note that only one out of "adi_spi_oa.c" and "adi_spi_generic.c" should be used for a specific project. In the example project SoftwareLoopback, separate build configurations are defined for each SPI protocol, with the relevant source code files included in each build.

DRIVER CONFIGURATIONS

Driver configurations can be configured at build time by defining the following preprocessor symbols:

- **ADIN2111**: this is required to target the ADIN2111
- **SPI_OA_EN**: select SPI protocol. The driver supports generic SPI and OPEN Alliance SPI protocols, and by default it uses generic SPI. Defining the preprocessor symbol SPI_OA_EN will select the OPEN Alliance protocol.
- **SPI_CRC_EN**: (for generic SPI only) enables CRC on the SPI transfers
- **SPI_PROT_EN**: (for OPEN Alliance SPI only) enable protection on SPI control transactions
- **STM32L4S5xx/USE_HAL_DRIVER**: required if the target host MCU is from the STM32L4S5 family
- **TX_QUEUE_NUM_ENTRIES/RX_QUEUE_NUM_ENTRIES**: define the depth of Tx and Rx queues, respectively. If they are not defined by the user, they will default to values defined in adi_mac.h (depth of 4 for each queue)
- **ADI_MAC_ENABLE_RX_QUEUE_HI_PRIO**: enables a second Rx queue, use to receive high priority frames. When the high priority Rx queue is enabled, the existing Rx queue will be used to receive normal/low priority frames. Availability of 2 separate Rx queues can be advantageous if, for example, there are different destination points in the application for high and normal/low priority frames.

CONFIGURATION SYNCHRONIZATION

OPEN Alliance SPI defines a SYNC bit that indicates if the device has been configured or if it is in an initial unconfigured state (see section 7.6 of "OPEN Alliance 10BASE-T1x MACPHY Serial Interface v1.0"). This bit is located in the register CONFIG0 and it is cleared on reset. Once the device configuration is complete, the user is required to set this bit to 1. If CONFIG0.SYNC=0, no frame transmission or reception will take place.

The driver provides an API for the purpose of setting the SYNC bit: adin2111_SyncConfig(). This API will also enable the host IRQ corresponding to INT_N and set the internal driver variable configSync. As such, this API needs to be called when using OPEN Alliance SPI as well as generic SPI protocol, even though the latter does not mandate the usage of CONFIG0.SYNC bit.

MEMORY FOOTPRINT

The following table shows the memory footprint of the device driver (in bytes) when configured with either of the SPI protocols. The figures were obtained by compiling the driver with medium optimization setting in IAR Embedded Workbench. Note the SPI read/write buffers make up most of the "rw data" figures, sufficiently large to contain a full Ethernet frame with some margin. When configured with OpenAlliance SPI protocol, these number can be reduced by limiting the number of chunks allowed to be transferred in a single SPI transaction.

	OpenAlliance SPI			**Generic SPI**		
	ro code	**ro data**	**rw data**	**ro code**	**ro data**	**rw data**
adi_phy.o	3520	3	144	3520	3	144
adi_mac.o	4072	3	136	4500	2	128
adi_spi_oa.o	4000	0	4004	-	-	-
adi_spi_generic.o	-	-	-	1964	0	4008
fcs.o	56	1024	0	56	1024	0
adin2111.o	1190	0	4	1174	0	4
Total (driver)	12838	1030	4288	11214	1029	4284
Driver struct	0	0	1164	0	0	500
Total	**12838**	**1030**	**5452**	**11214**	**1029**	**4784**

CALLBACKS

The driver includes an API (adin2111_RegisterCallback) that allows registering callback functions for various events. Currently the following events are supported (defined in adi_mac_InterruptEvt_e):

- TX_RDY asserted: triggered when STATUS1.TX_RDY interrupt source is asserted
- RX_RDY asserted: triggered when STATUS1.P1_RX_RDY interrupt source is asserted
- Receive frame ready: triggered when a full frame was read from the ADIN2111 but before it is copied from the SPI buffer to the buffer submitted by the application

DYNAMIC FORWARDING TABLE

When a frame is submitted for transmit, the caller can specify a single port to use for transmit (ADIN2111_PORT_1 or ADIN2111_PORT_2) or to transmit it on both ports (ADIN2111_PORT_FLOOD). In normal operation transmitting a frame through a single port is more efficient but it requires the application to maintain a map of the network.

To help determine what port to use for transmit, the driver implements a dynamic forwarding table, built from received frames: the source address is extracted from each received frame and inserted into the table if it doesn't already exist, together with the port on which the frame was received. When submitting a buffer for transmit with the port set to ADIN2111_PORT_AUTO, the address is looked up in the dynamic forwarding table:

- if it exists, the frame will be sent to the port retrieved from the table
- if it does not exist, it is sent to both port (equivalent of setting port to ADIN2111_PORT_FLOOD)

The table supports an aging mechanism, whereas each new entry in the table starts with a maximum age value of ADIN2111_DYNAMIC_TABLE_MAX_AGE. On each aging tick (supplied by the application by calling the adin2111_AgeTick() function), every entry whose address has not been encountered since the previous age tick will have its age decremented. An entry with age 0 is considered invalid and can be replaced by a new entry if needed.

This way old entries (for example MAX addresses corresponding to hosts that were meantime removed from the network) are removed from the forwarding table without explicit user intervention.

HARDWARE PLATFORMS

The ADIN2111 device driver is designed to facilitate porting to different hardware platforms. It also includes out of the box support for specific hardware platforms.

SUPPORTED PLATFORMS

The following describes the hardware platforms supported out of the box, and additional requirements specific to each platform.

Eval-ADIN2111

Eval-ADIN2111 is based on the STM32L4S5 (ARM Cortex-M4F MCU), please consult the evaluation board user's guide for evaluation board details.

In order to run the examples projects, the STM32CubeL4 package needs to be downloaded and installed. The example projects use an environment variable **STM32CUBE_FW_L4_PATH** to point to the location of the STM32CubeL4 on the local hard drive.

The following steps detail how to setup the environment variable (Windows 10):

1. Right-click on **Start Menu** and choose **Settings**, this will open the **Windows Settings** window
2. In the search box type "environment" and choose **Edit environment variables for your account**
3. Click **New…** and enter:
    a. **Variable Name:** STM32CUBE_FW_L4_PATH
    b. **Variable Value:** browse to the installation directory for the STM32CubeL4 package
4. Click **OK**. Note that IAR EWARM needs to be restarted for the new environment variable to take effect.

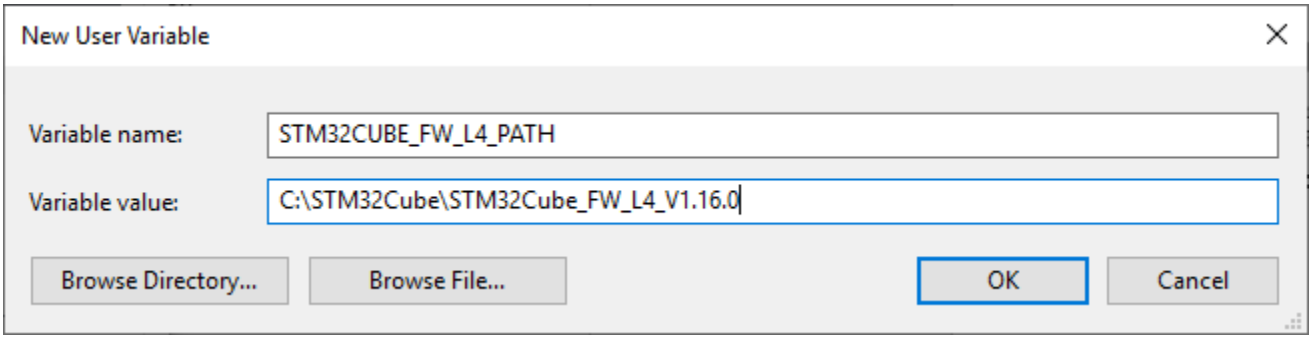

Figure 4 Environment variable setup

PORTING TO A NEW PLATFORM

To port the driver to another platform, the HAL API needs to be implemented, providing access to the SPI host interface and interrupt handling.

The HAL API is documented in the "ADIN2111 Device Driver API Reference Manual" HTML document.

RUNNING THE EXAMPLES

All example projects are located under “examples/adin2111” in the device driver installation directory.

They have been developed and tested using IAR Embedded Workbench 8.32.4.

Each example has a “readme.txt” file in the project directory which provides project specific information, such as expected LED behavior, UART output, configuration options, etc.

The following sections present additional details related to examples setup.

SYMBOLS DEFINITION

It is sometimes necessary to manipulate preprocessor symbols to control features of the driver/example code. One such instance is related to the SPI CRC feature: the SDO/CRC_ENB pin strap option of the ADIN2111 configures the SPI slave hardware to use or not use CRC, and the software needs to match the hardware configuration to function correctly. The preprocessor symbol SPI_CRC_EN is used to configure the software: if CRC is used by the hardware, SPI_CRC_EN needs to be defined, and if CRC is not used by the hardware, SPI_CRC_EN should not be defined.

Symbols are defined in Project Options->C/C++ Compiler/Preprocessor, and the figure below shows the symbol being defined for “SoftwareLoopback”:

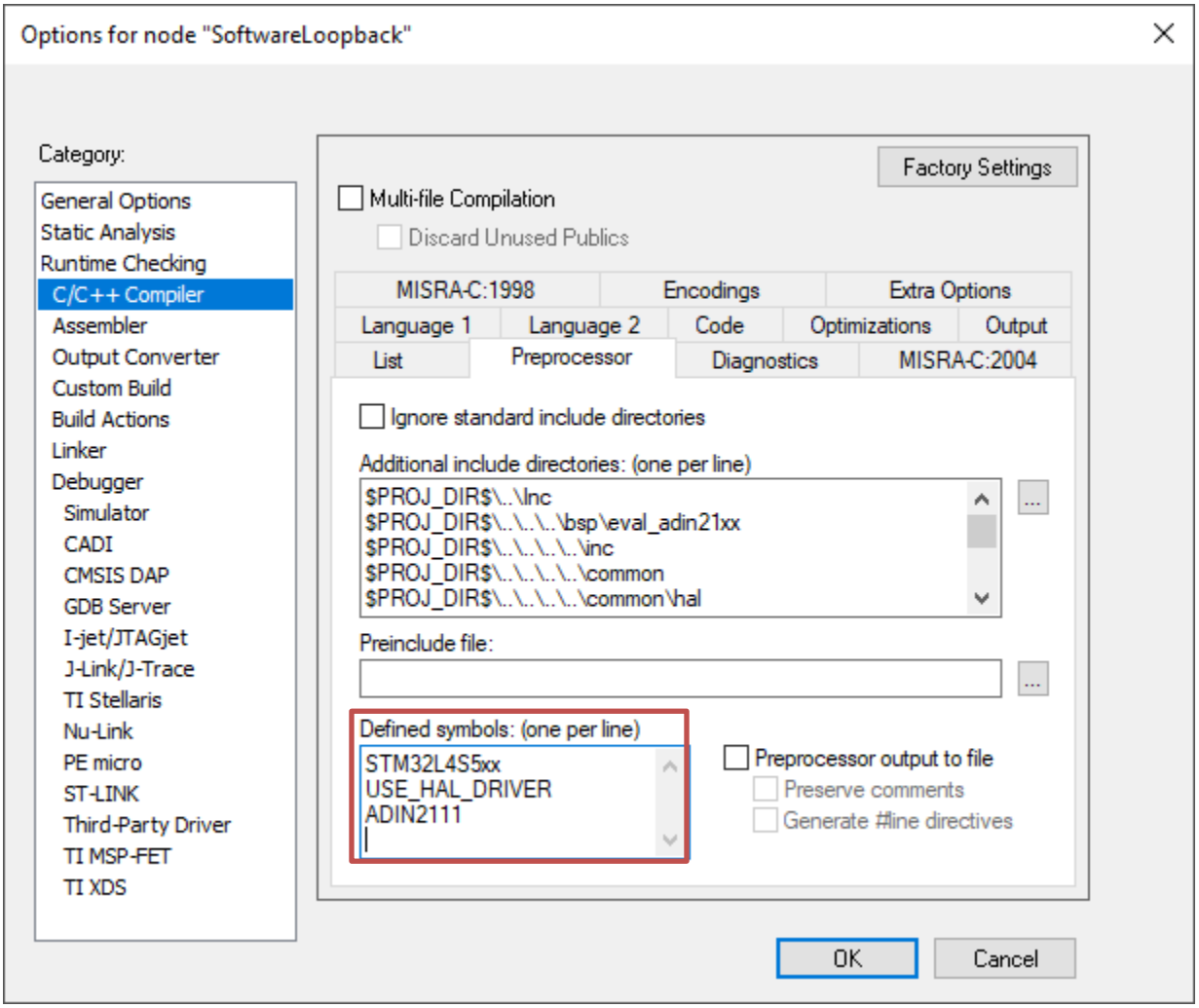

Figure 5 Symbols definition

ESD Caution

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

Legal Terms and Conditions

By using the evaluation board discussed herein (together with any tools, components documentation or support materials, the "Evaluation Board"), you are agreeing to be bound by the terms and conditions set forth below ("Agreement") unless you have purchased the Evaluation Board, in which case the Analog Devices Standard Terms and Conditions of Sale shall govern. Do not use the Evaluation Board until you have read and agreed to the Agreement. Your use of the Evaluation Board shall signify your acceptance of the Agreement. This Agreement is made by and between you ("Customer") and Analog Devices, Inc. ("ADI"), with its principal place of business at One Technology Way, Norwood, MA 02062, USA. Subject to the terms and conditions of the Agreement, ADI hereby grants to Customer a free, limited, personal, temporary, non-exclusive, non-sublicensable, non-transferable license to use the Evaluation Board FOR EVALUATION PURPOSES ONLY. Customer understands and agrees that the Evaluation Board is provided for the sole and exclusive purpose referenced above, and agrees not to use the Evaluation Board for any other purpose. Furthermore, the license granted is expressly made subject to the following additional limitations: Customer shall not (i) rent, lease, display, sell, transfer, assign, sublicense, or distribute the Evaluation Board; and (ii) permit any Third Party to access the Evaluation Board. As used herein, the term "Third Party" includes any entity other than ADI, Customer, their employees, affiliates and in-house consultants. The Evaluation Board is NOT sold to Customer; all rights not expressly granted herein, including ownership of the Evaluation Board, are reserved by ADI. CONFIDENTIALITY. This Agreement and the Evaluation Board shall all be considered the confidential and proprietary information of ADI. Customer may not disclose or transfer any portion of the Evaluation Board to any other party for any reason. Upon discontinuation of use of the Evaluation Board or termination of this Agreement, Customer agrees to promptly return the Evaluation Board to ADI. ADDITIONAL RESTRICTIONS. Customer may not disassemble, decompile or reverse engineer chips on the Evaluation Board. Customer shall inform ADI of any occurred damages or any modifications or alterations it makes to the Evaluation Board, including but not limited to soldering or any other activity that affects the material content of the Evaluation Board. Modifications to the Evaluation Board must comply with applicable law, including but not limited to the RoHS Directive. TERMINATION. ADI may terminate this Agreement at any time upon giving written notice to Customer. Customer agrees to return to ADI the Evaluation Board at that time. LIMITATION OF LIABILITY. THE EVALUATION BOARD PROVIDED HEREUNDER IS PROVIDED "AS IS" AND ADI MAKES NO WARRANTIES OR REPRESENTATIONS OF ANY KIND WITH RESPECT TO IT. ADI SPECIFICALLY DISCLAIMS ANY REPRESENTATIONS, ENDORSEMENTS, GUARANTEES, OR WARRANTIES, EXPRESS OR IMPLIED, RELATED TO THE EVALUATION BOARD INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, TITLE, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS. IN NO EVENT WILL ADI AND ITS LICENSORS BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES RESULTING FROM CUSTOMER'S POSSESSION OR USE OF THE EVALUATION BOARD, INCLUDING BUT NOT LIMITED TO LOST PROFITS, DELAY COSTS, LABOR COSTS OR LOSS OF GOODWILL. ADI'S TOTAL LIABILITY FROM ANY AND ALL CAUSES SHALL BE LIMITED TO THE AMOUNT OF ONE HUNDRED US DOLLARS ($100.00). EXPORT. Customer agrees that it will not directly or indirectly export the Evaluation Board to another country, and that it will comply with all applicable United States federal laws and regulations relating to exports. GOVERNING LAW. This Agreement shall be governed by and construed in accordance with the substantive laws of the Commonwealth of Massachusetts (excluding conflict of law rules). Any legal action regarding this Agreement will be heard in the state or federal courts having jurisdiction in Suffolk County, Massachusetts, and Customer hereby submits to the personal jurisdiction and venue of such courts. The United Nations Convention on Contracts for the International Sale of Goods shall not apply to this Agreement and is expressly disclaimed.

©2020 Analog Devices, Inc. All rights reserved. Trademarks and registered trademarks are the property of their respective owners.

ANALOG DEVICES

www.analog.com