*User's Guide*

# *Quick-Start Guide for TLV320AIC310xEVM Control Software*

TEXAS INSTRUMENTS


**ABSTRACT**

This quick-start guide describes the installation, operation, and use of the TLV320AIC310x evaluation module (EVM) control software. The EVM control software allows for the evaluation of the TLV320AIC310x family of audio codecs by providing a user-friendly graphical user interface (GUI) to configure the TLV320AIC310xEVM.


## Table of Contents



## List of Figures

## List of Tables



## Trademarks

LabVIEW® is a registered trademark of National Instruments.
SPI® is a registered trademark of Motorola, Inc.
All trademarks are the property of their respective owners.

# 1 TLV320AIC310xEVM Control Software Installation

Section 1 provides information on the installation process for the TLV320AIC310xEVM control software and the necessary drivers.

## 1.1 Software Installation

1. Download the latest version of the software located on the device product page. For example, if you are evaluating the TLV320AIC3106, please see the AIC3106 product page.
2. Unzip the installation file by right-clicking the zip file and selecting *Extract All*. Extract the zip file contents to a known location.
3. When the zip file is extracted, run the executable by double-clicking the .exe file. Running the executable as an administrator.
4. After running the executable, the TLV320AIC310xEVM control software setup begins. Follow the prompts in the pop-up window, accept the license agreements, and choose an installation directory. Figure 1-1 shows this process.

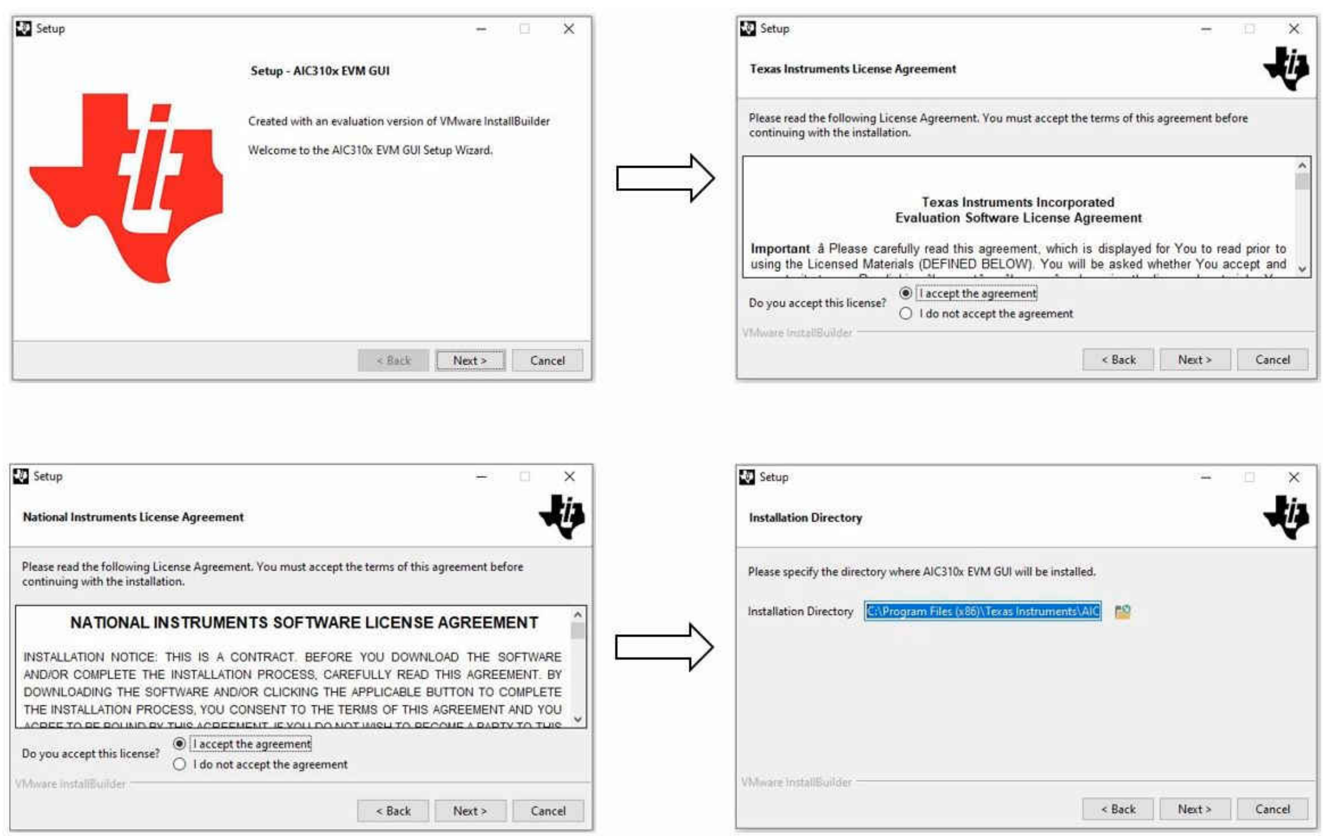



**Figure 1-1. AIC310x Control Software Setup**

LabVIEW® runtime engine, USB-MODEVM drivers and AC-MODEVM drivers will also be installed.

5. The setup is now ready to begin installing. A window similar to Figure 1-2 should appear. Click next to proceed.

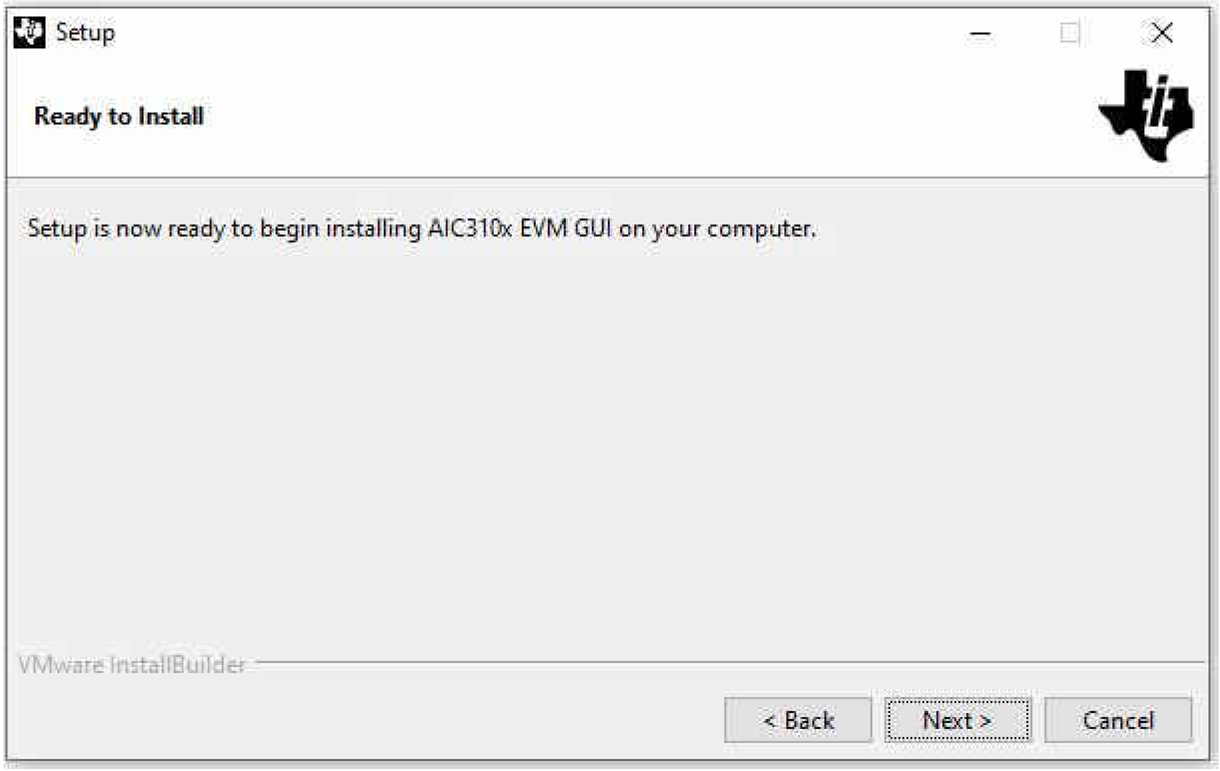



**Figure 1-2. Ready to Install**

6. If this is the first time installing NI LabView package, the GUI will install NI Package Manager and the respective run time engines or a "no operation to be performed" message if it finds the relevant NI packages. Accept the license agreement and hit next to proceed with the installation.

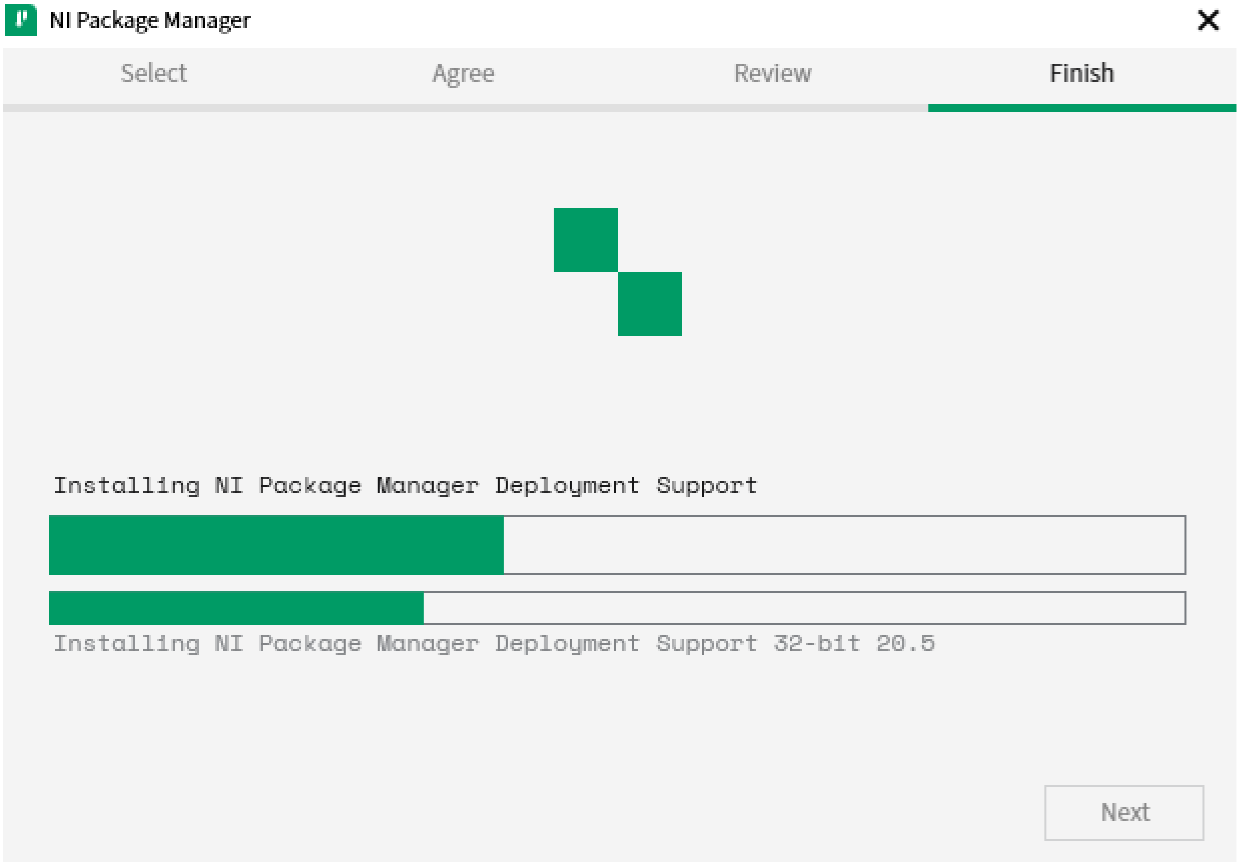



**Figure 1-3. NI Package Manager Installation**

7. Next it will install the NI LabView Run Time Engine. Accept the license agreement and hit next to proceed with the installation.

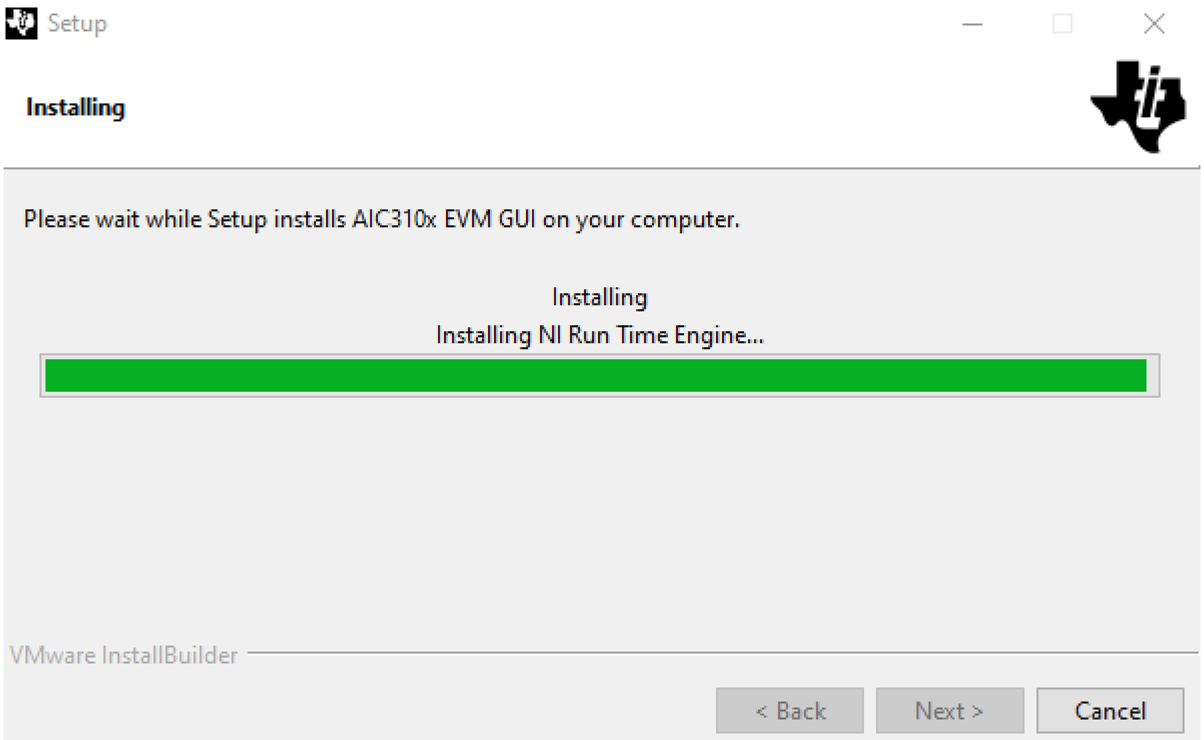



**Figure 1-4. NI LabView Run Time Engine Installation**

8. When installation is complete close the screen by clicking the 'X' and **do not** reboot the system untill all driver installations are complete.

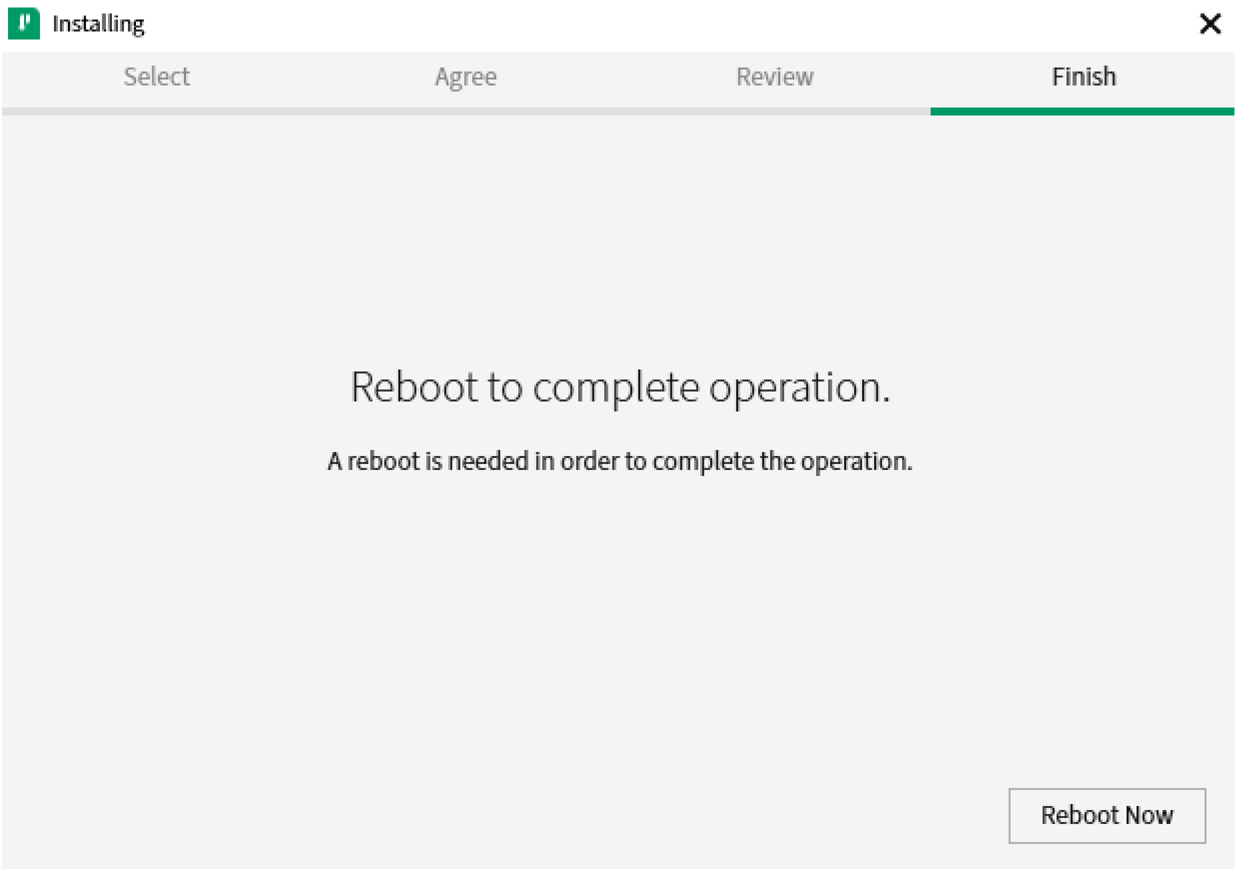



**Figure 1-5. Reboot Screen, hit 'X' to close - do not reboot now**

9. Next it will install the NI VISA Run Time Engine. Click next to proceed, follow the installation prompts and keep the default feature and setting.

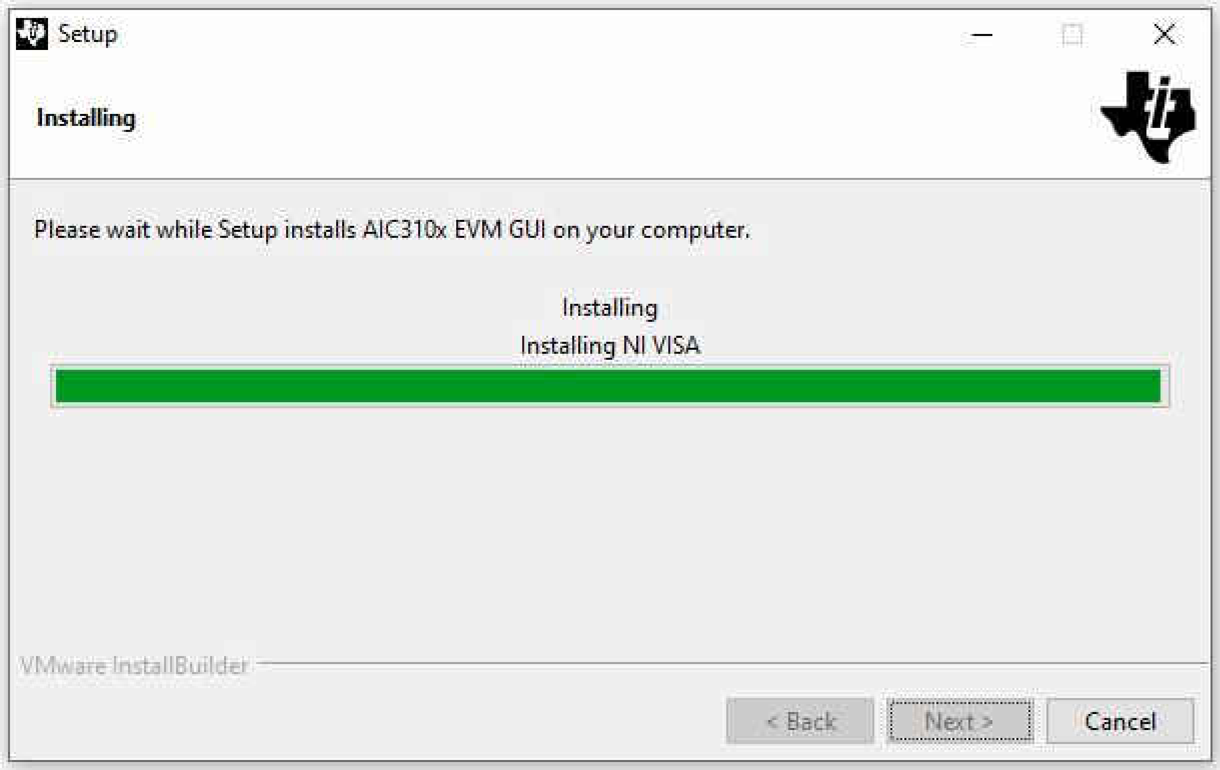



**Figure 1-6. NI Visa Run Time Engine Installation**


**Figure 1-7. NI VISA Run Time Engine**

10. Upon completion of the installation. Click "Restart Later" to close the screen, do not restart the system now and let the GUI installation to proceed.

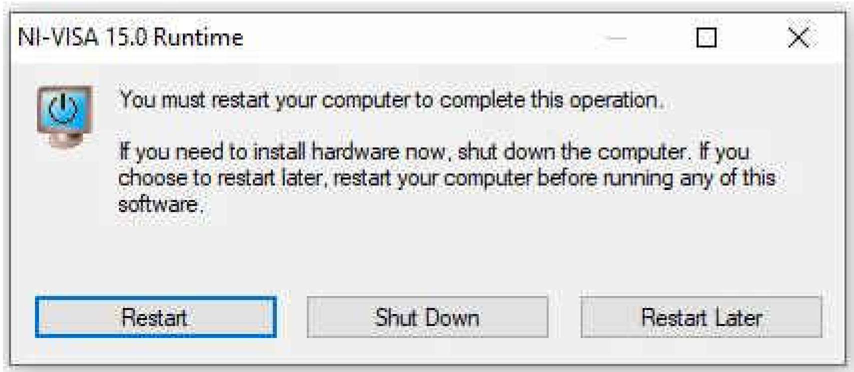



**Figure 1-8. NI VISA Run Time Completion - Restart Later**

11. If NI package was installed previously a "No operation to be performed." screen will appear. Close the screen by clicking the 'X' and proceed with the GUI installation.

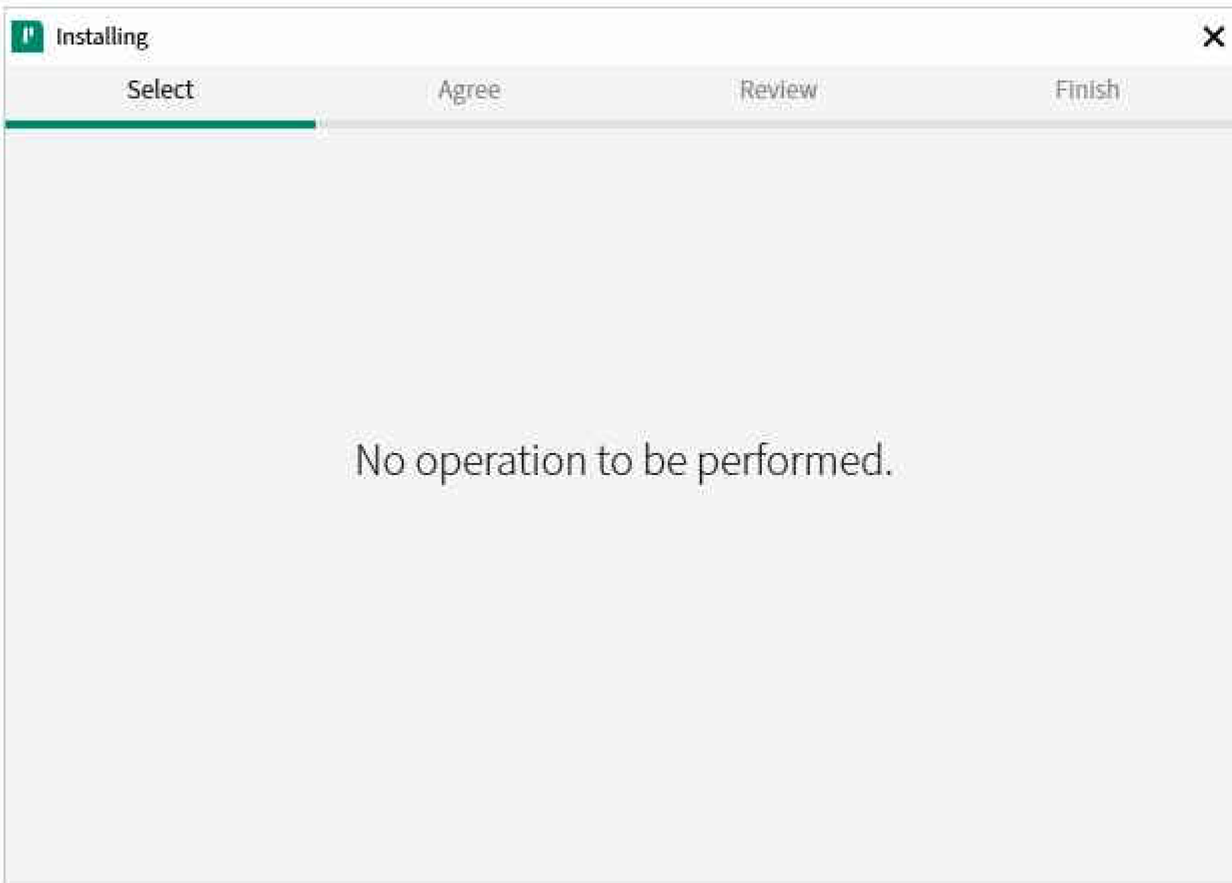



**Figure 1-9. No Operation Screen**

12. When the NI run time is done installing, the GUI will install the USB-MODEVM automatically.

13. The AC MODEVM USB audio device driver setup now begins. Follow the prompts by confirming the installation and choosing the installation location. Click *Install* when ready. When the installation is complete, click *Next*. Figure 1-10 shows this process.

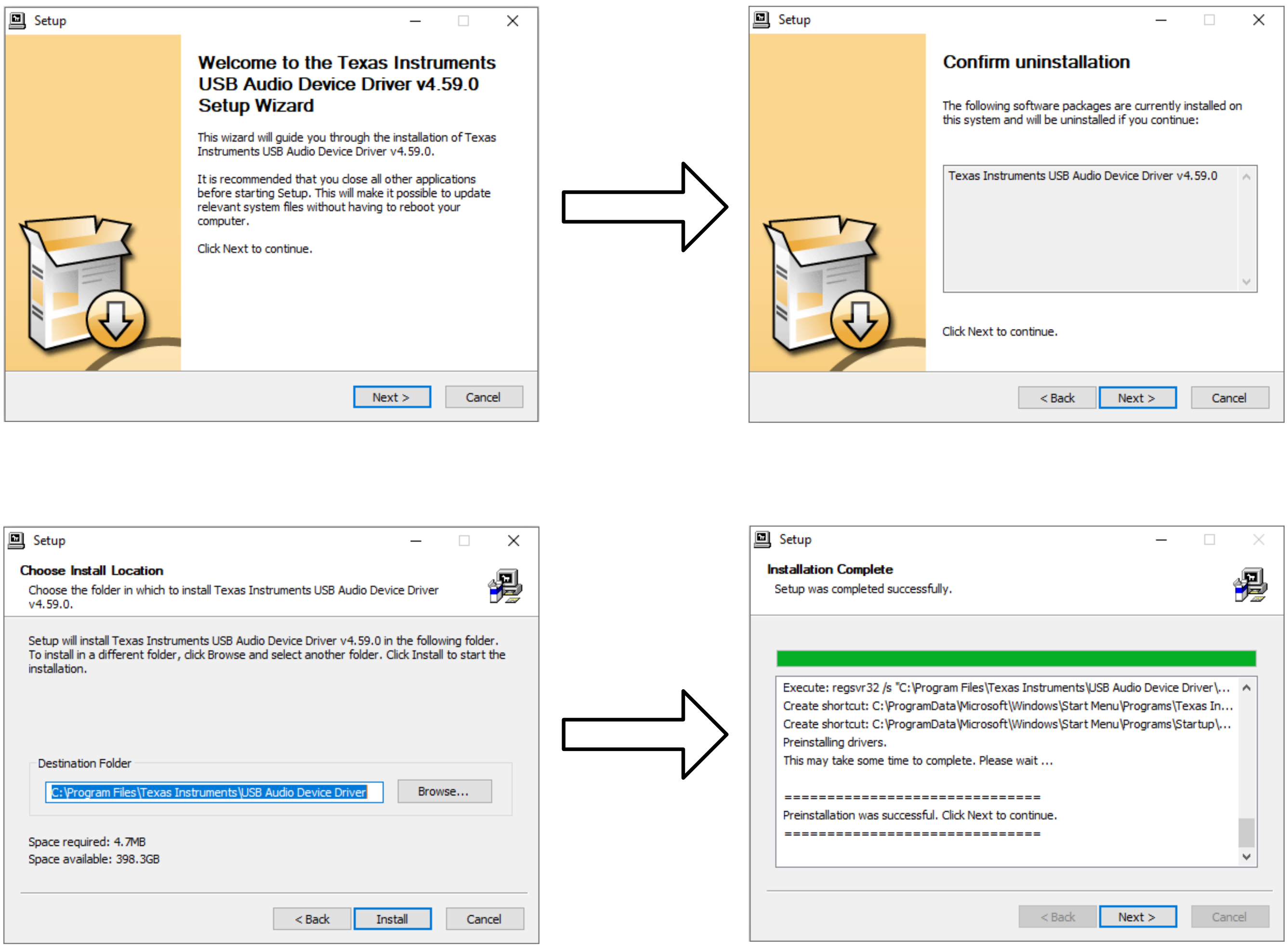



**Figure 1-10. USB AUDIO Device Driver Setup**

14. The USB audio device driver is now installed. Click *Finish* to close the setup wizard.
15. A pop-up window will appear asking the user to disconnect and reconnect the device to complete the driver installation. Click *Yes* to continue.
16. The setup has now finished (Figure 1-11). Check whether you wish to have a shortcut created and run the AIC310x EVM GUI or click *Finish* to end installation.

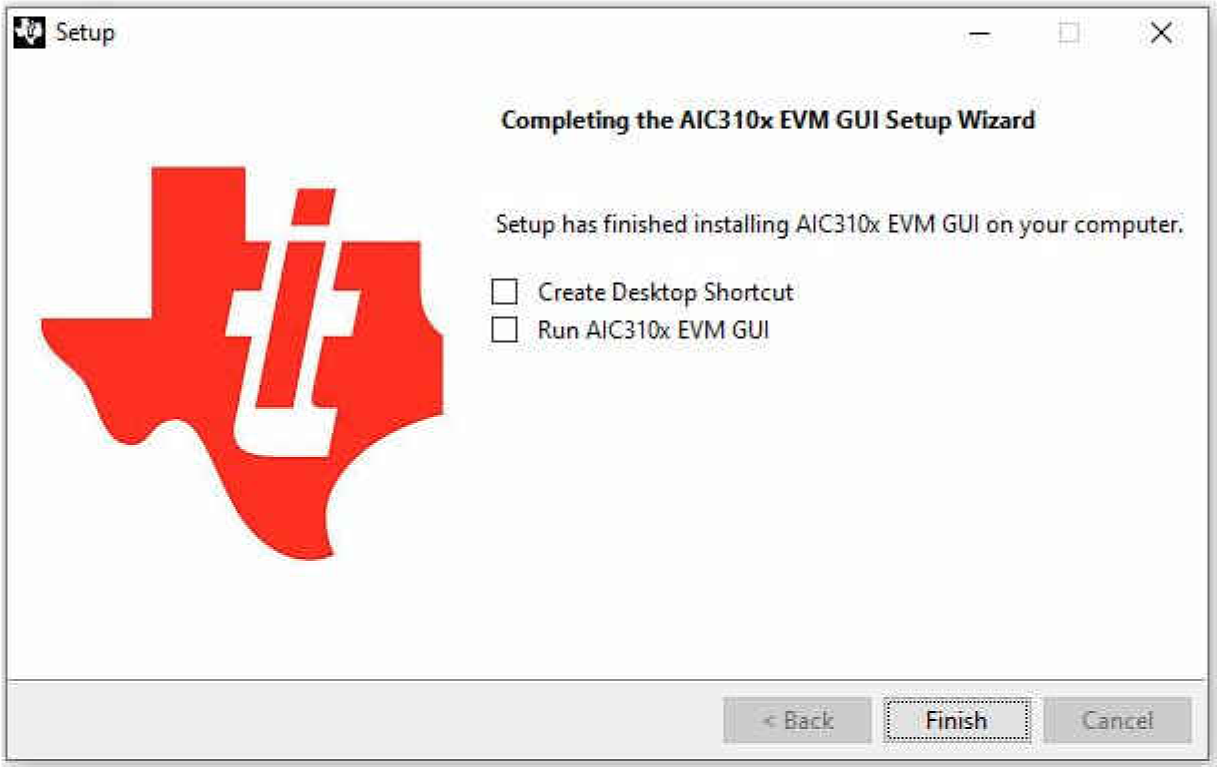



**Figure 1-11. Setup Finished**

The installation process is now complete for the TLV320AIC310x control software and the required drivers. Prior to connecting the EVM, reboot the PC.

## 1.2 EVM Connections

1. Ensure that the TLV320AIC310xEVM is installed on the USB-MODEVM or AC-MODEVM interface board, aligning J11A/J7, J12A/J11, J16A/J8, J17A/J12, and J18A/J9 with the corresponding connectors on the TLV320AIC310xEVM.
2. Verify that the jumpers and switches are in their default positions. The default positions of the jumper and switches are provided in the corresponding TLV320AIC310x user's guide.
3. Attach a USB cable from the PC to the USB-MODEVM or AC-MODEVM interface board. The default configuration provides power, control signals, and streaming audio via the USB interface from the PC. On the USB-MODEVM, LED D2 or D3 on AC-MODEVM lights up to indicate that the USB interface is active.
4. For the first connection, the PC recognizes new hardware and begins an initialization process. The user may be prompted to identify the location of the drivers or to allow the PC to automatically search for them. If prompted, choose to allow the automatic detection option to locate the drivers.

After the TLV320AIC310xEVM-PDK software installation (described in Section 1.1) is complete, evaluation and development with the TLV320AIC310xEVM can begin.

# 2 TLV320AIC310xEVM Control Software

Section 2 discusses how to quickly start the control software and load a record or playback preset into the EVM using the TLV320AIC310x control software.

**Note**

For configuration of the codec, the TLV320AIC310x block diagram is located in the device data sheet. The block diagram is a good reference to help determine the signal routing.

## 2.1 Device Selection for Operation With the AIC310xEVM

The installed software provides operation for several devices in the TLV320AIC310x family. An initial window appears similar to Figure 2-1 when the software is run. As an example, for operation with the TLV320AIC3106EVM, select *AIC3106* from the drop-down menu and click **Accept**. The software takes a few seconds to be configured for operation before proceeding. A progress bar shows the status of the configuration. When configured, a default software screen (see Figure 2-2) displays the EVM GUI.

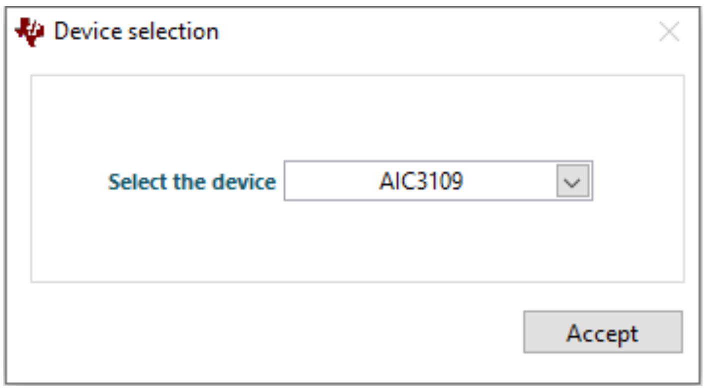



**Figure 2-1. Device Selection Window**

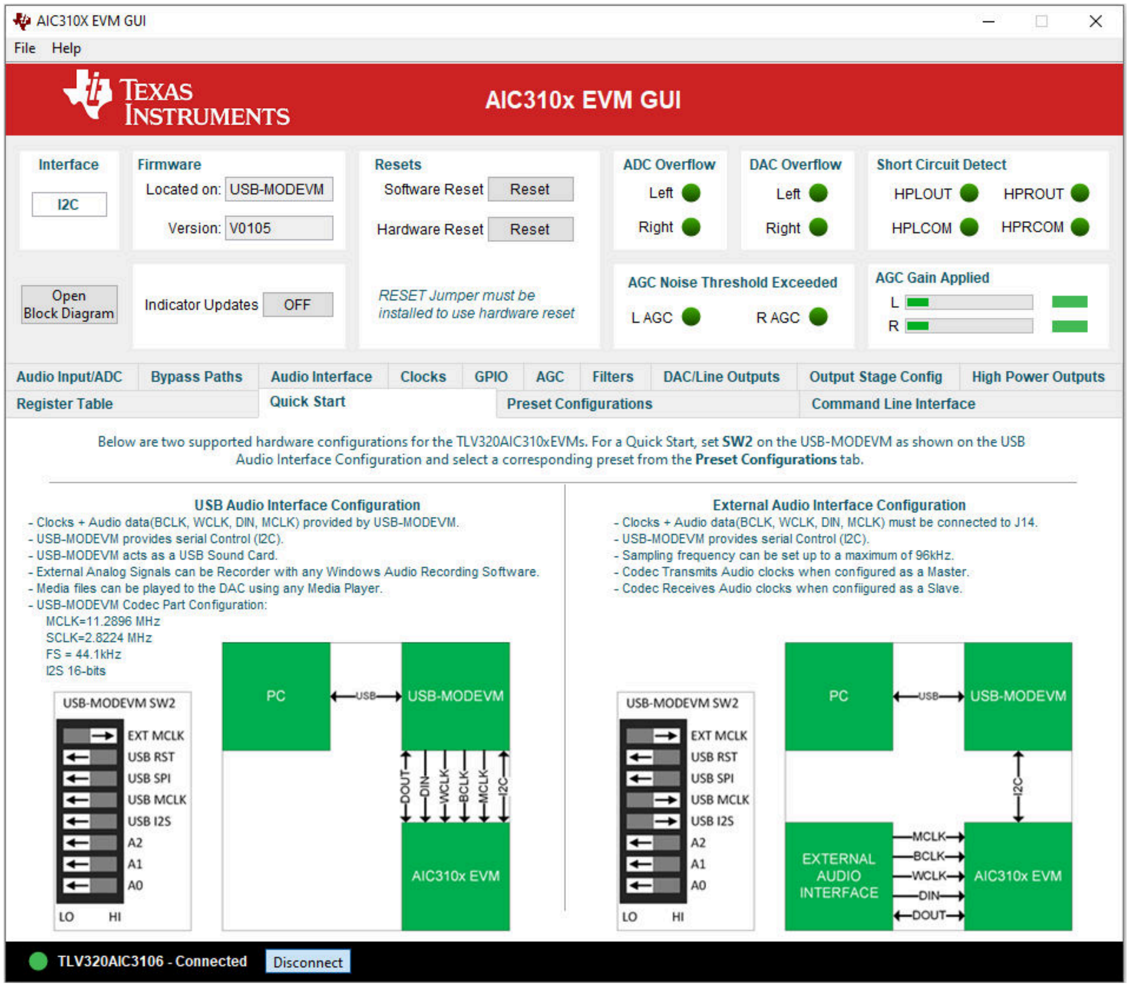


Figure 2-2. Default Software Screen

## 2.2 Front-Page Indicators and Functions

Figure 2-2 shows the main screen of the EVM software. The indicators and buttons located above the tabbed section of the front page are visible regardless of which tab is currently selected.

At the top left of the screen is an **Interface** indicator. This indicator shows which interface is selected for controlling the TLV320AIC310xEVM.

To the right of the **Interface** indicator is a group box called **Firmware**. This box indicates where the firmware being used is operating from—USB-MODEVM or AC-MODEVM. The GUI will auto detect which version of MODEVM is connected. In this example it is with the USB-MODEVM, so the user should see *USB-MODEVM* in the box labeled **Located On:**. The version of the firmware appears in the **Version** box below this box.

To the right, the next group box contains controls for resetting the TLV320AIC310x. A software reset can be done by writing to a register in the TLV320AIC310x, which is accomplished by pushing the button labeled **Software Reset**. The TLV320AIC310x also may be reset by toggling a pin on the TLV320AIC310x, which is done by pushing the **Hardware Reset** button.

**CAUTION**

In order to perform a hardware reset, the RESET jumper must be installed and SW2-7 on the USB-MODEVM must be turned OFF. Failure to do either of these steps results in not generating a hardware reset or causing unstable operation of the EVM, which may require cycling power to the USB-MODEVM.

On the lower left portion of the screen, the **Device Connected** LED must be green when the EVM is connected. If the indicator is red, the EVM is not properly connected to the PC. Disconnect the EVM and verify that the drivers were correctly installed, then reconnect and try restarting the software.

On the upper right portion of the screen, several indicators are located that provide the status of various portions of the TLV320AIC310x. These indicators are activated by pressing the **Indicator Updates** button below the *Firmware* section. These indicators, as well as the other indicators on this panel, are updated only when the front panel of the software is inactive, once every 20 ms.

The **ADC Overflow** and **DAC Overflow** indicators light up when the overflow flags are set in the TLV320AIC310x. Below these indicators are the **AGC Noise Threshold Exceeded** indicators that show when the AGC noise threshold is exceeded. To the far right of the screen, the **Short Circuit Detect** indicators show when a short-circuit condition is detected, if this feature is enabled. Below the short-circuit indicators, the **AGC Gain Applied** indicators use a bar graph to show the amount of gain applied by the AGC, and indicators that light when the AGC is saturated.

## 2.3 Default Configuration (Presets) Tab

The default configuration tab in Figure 2-3 provides several different preset configurations of the codec. The **Preset Configurations** buttons allow the user to choose from the provided defaults. When the selection is made, the **Preset Configuration Description** box shows a summary of the codec setup associated with the choice made. If the choice is acceptable, the **Load** button can be pressed and the preset configuration is loaded into the codec. The user can change to the **Command Line Interface** tab (see Figure 2-4) to view the actual settings that were programmed into the codec.

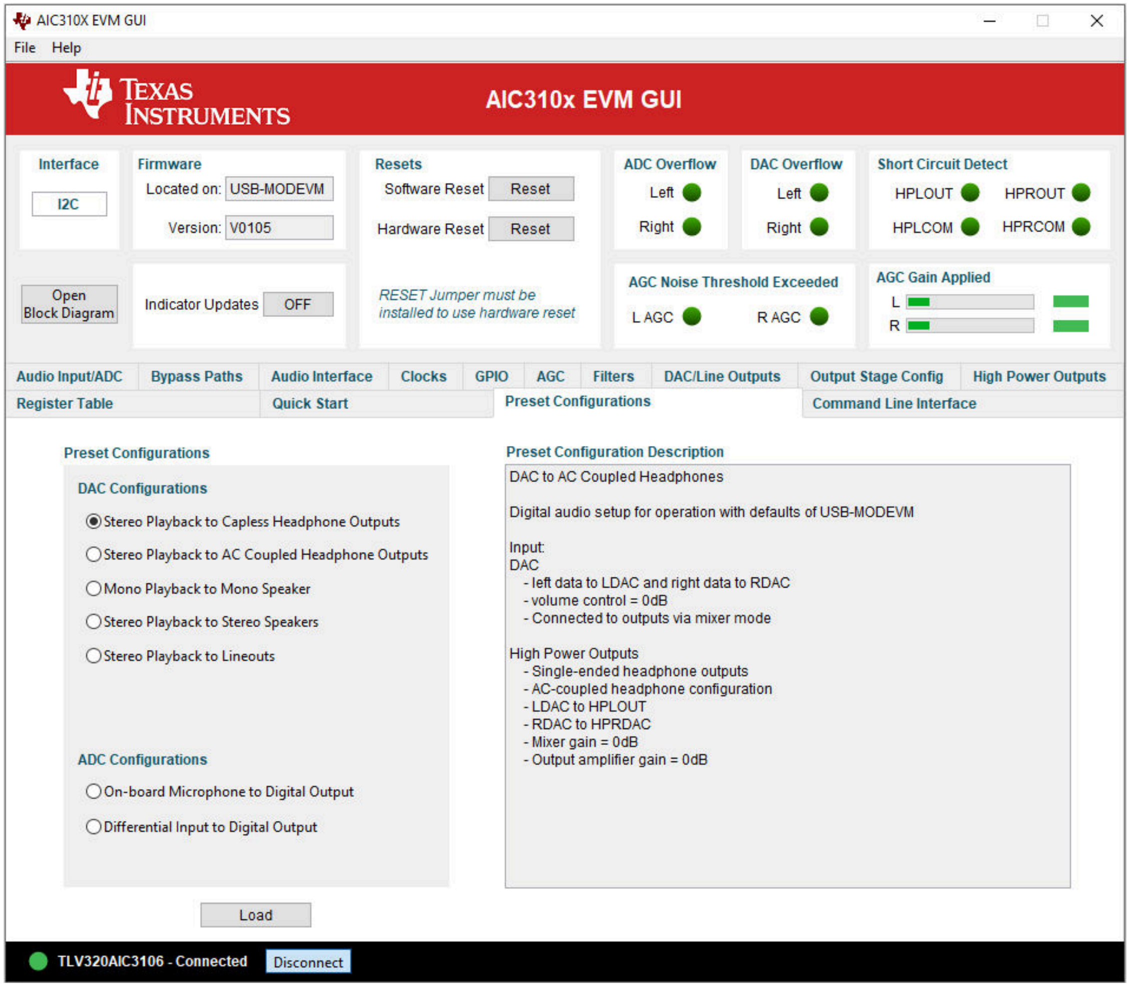



**Figure 2-3. Preset Configurations Tab**

## 2.4 Command Line Interface Tab

A simple scripting language controls the processor on the USB-MODEVM or AC-MODEVM from the LabView®-based PC software. The main program controls throughout the control software do nothing more than write a script that is then handed off to an interpreter that sends the appropriate data to the correct USB endpoint. Because this system is script-based, a provision is made in this tab (Figure 2-4) for the user to view the scripting commands created when the controls are manipulated, as well as load and execute other scripts that are written and saved. This design allows the software to be used as a quick test tool or to help provide troubleshooting information in the rare event that the user encounters a problem with this EVM.

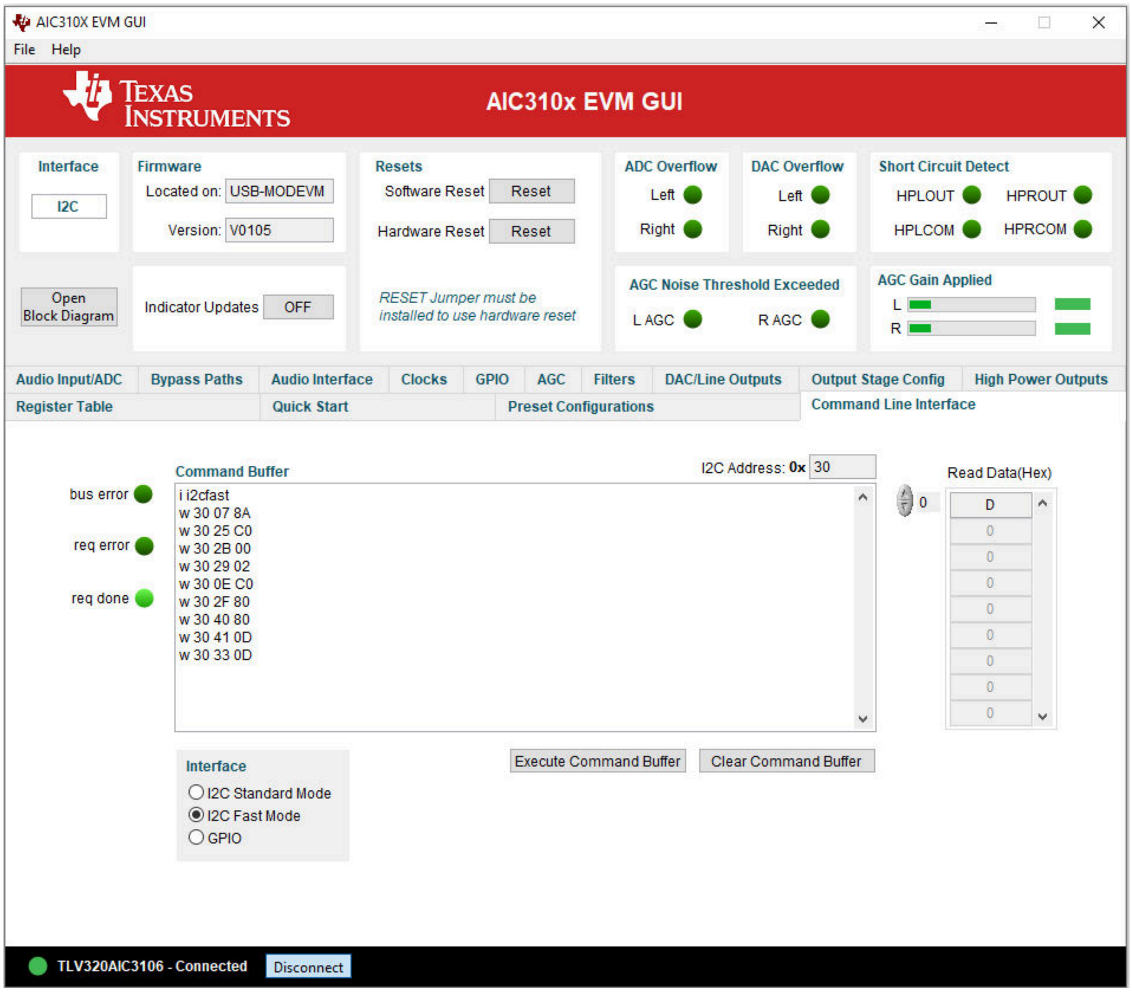



**Figure 2-4. Command Line Interface Tab**

A script is loaded into the command buffer, either by operating the controls on the other tabs or by loading a script file. When executed, the return packets of data that result from each command are displayed in the **Read Data** array control. When executing several commands, the read data control shows only the results of the last command. To see the results after every executed command, use the logging function described in this section.

The file menu (Figure 2-5) provides some options for working with scripts. The first option, *Open Command File...*, loads a command file script into the command buffer. This script can then be executed by pressing the **Execute Command Buffer** button. The second option, *Save Command File*, allows the user to save a script from the command line interface. The saved commands are then saved as a text file in a location specified by the user.

The third option is *Log Script and Results...*, which opens a file save dialog box. Choose a location for a log file to be written using this file save dialog. When the **Execute Command Buffer** button is pressed, the script runs and the script, along with the resulting data read back during the script, is saved to the file specified. The log file is a standard text file that can be opened with any text editor, and looks much like the source script file, but with the additional information of the result of each script command executed.

The fourth menu item is a submenu of *Recently Opened Files*. This option is simply a list of script files that have previously been opened, allowing fast access to commonly used script files. The final menu item is *Exit*, which terminates the TLV320AIC310xEVM control software.

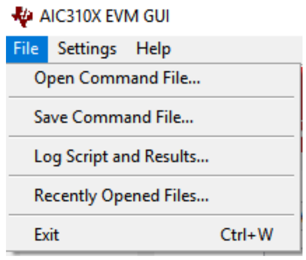



**Figure 2-5. File Menu**

Under the Help menu is an *About...* menu item that displays information about the TLV320AIC310xEVM control software.

# 3 MODEVM

## 3.1 MODEVM Operation

This section provides information on the analog input and output, digital control, power, and general connection of the TLV320AIC310xEVM using either the USB-MODEVM or AC-MODEVM.

### 3.1.1 TLV320AIC310xEVM-PDK Block Diagram

The TLV320AIC310xEVM-PDK consists of two separate circuit boards, the MODEVM and the TLV320AIC310xEVM. The MODEVM can either be the USB-MODEVM or AC-MODEVM. The USB-MODEVM is built around a TAS1020B streaming audio USB controller with an 8051-based core and the AC-MODEVM is based on XMOS xCORE multicore microcontrollers. The motherboard features two positions for modular EVMs, or one double-wide serial modular EVM can be installed. The TLV320AIC310xEVM is one of the double-wide modular EVMs that is designed to work with either of the MODEVM.

The simple diagram in Figure 3-1 shows how the TLV320AIC310xEVM is connected to the MODEVM. The MODEVM interface board is intended to be used in USB mode, where control of the installed EVM is accomplished using the onboard USB controller device. Provisions are made, however, for driving all the data buses ($I^2$C, SPI®, $I^2$S, and AC97) externally. The source of these signals is controlled by SW2 on the MODEVM. See Table 3-1 for details on the switch settings for USB-MODEVM. For AC-MODEVM switching SW2 (S0) to OFF position will allow user to use external audio interface through J10 header.

The USB-MODEVM has two EVM positions that allow for the connection of two small evaluation module or one larger evaluation module. The TLV320AIC310xEVM is designed to fit over both of the smaller evaluation module slots as illustrated in Figure 3-1.

#### 3.1.1.1 MODEVM Interface Board

The simple diagram in Figure 3-1 shows only the basic features of the MODEVM interface board.

Because the TLV320AIC310xEVM is a double-wide modular EVM, the TLV320AIC310xEVM is installed with connections to both EVM positions, which connects the TLV320AIC310x digital control or digital audio interface to either the TAS1020B/XMOS microcontroller.

In the factory configuration, the board is ready to use with the TLV320AIC310xEVM. To view all functions and configuration options available on the MODEVM board, see the USB-MODEVM interface board schematic in Section 3 or AC-MODEVM in Section 4.

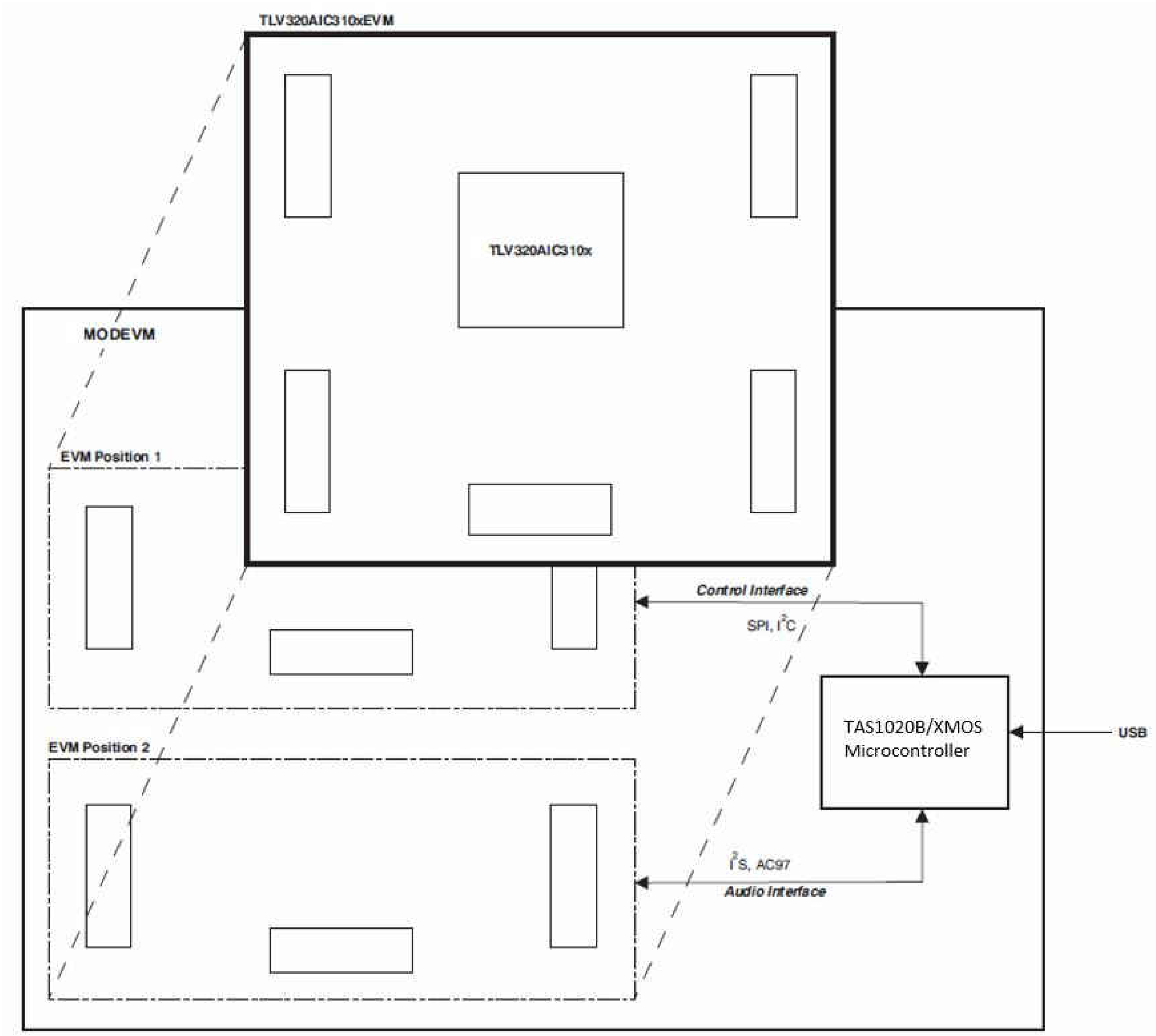



**Figure 3-1. TLV320AIC310xEVM-PDK Block Diagram**

### 3.1.2 Default Configuration and Connections

#### 3.1.2.1 USB-MODEVM SW2 Settings

Table 3-1 provides a list of the SW2 settings on the USB-MODEVM. For use with the TLV320AIC310xEVM, SW-2 positions 1 through 7 must be set to ON, while SW-2.8 must be set to OFF.

**Table 3-1. USB-MODEVM SW2 Settings**

| SW-2 Switch Number | Label | Switch Description |
| --- | --- | --- |
| 1 | A0 | USB-MODEVM EEPROM $I^2C$ address A0<br>ON: A0 = 0<br>OFF: A0 = 1 |
| 2 | A1 | USB-MODEVM EEPROM $I^2C$ address A1<br>ON: A1 = 0<br>OFF: A1 = 1 |
| 3 | A2 | USB-MODEVM EEPROM $I^2C$ address A2<br>ON: A2 = 0<br>OFF: A2 = 1 |
| 4 | USB $I^2S$ | $I^2S$ bus source selection<br>ON: $I^2S$ bus connects to TAS1020<br>OFF: $I^2S$ bus connects to USB-MODEVM J14 |
| 5 | USB MCK | $I^2S$ bus MCLK source selection<br>ON: MCLK connects to TAS1020<br>OFF: MCLK connects to USB-MODEVM J14 |
| 6 | USB SPI | SPI bus source selection<br>ON: SPI bus connects to TAS1020<br>OFF: SPI bus connects to USB-MODEVM J15 |
| 7 | USB RST | RST source selection<br>ON: EVM reset signal comes from TAS1020<br>OFF: EVM reset signal comes from USB-MODEVM J15 |
| 8 | EXT MCK | External MCLK selection<br>ON: MCLK signal is provided from USB-MODEVM J10<br>OFF: MCLK signal comes from either selection of SW2-5 |

#### 3.1.2.2 USB-MODEVM Operation

The USB-MODEVM interface board can be powered from several different sources:

- USB
- 6 Vdc–10 Vdc AC/DC external wall supply (not included)
- Lab power supply

When powered from the USB connection, JMP6 must have a shunt from pins 1–2 (the default factory configuration). When powered from 6 V–10 Vdc, either through the J8 terminal block or J9 barrel jack, JMP6 must have a shunt installed on pins 2–3. If power is applied in any of these ways, onboard regulators generate the required supply voltages and no further power supplies are necessary.

If lab supplies are used to provide the individual voltages required by the USB-MODEVM interface, JMP6 must have no shunt installed. Voltages are then applied to J2 (+5VA), J3 (+5VD), J4 (+1.8VD), and J5 (+3.3VD). The +1.8VD and +3.3VD can also be generated on the board by the onboard regulators from the +5VD supply; to enable this configuration, the switches on SW1 must be set to enable the regulators by placing them in the ON position (lower position, looking at the board with text reading right-side up). If +1.8VD and +3.3VD are supplied externally, disable the onboard regulators by placing SW1 switches in the OFF position.

Each power-supply voltage has an LED (D1-D7) that lights when the power supplies are active.

## 3.2 USB-MODEVM Schematic

The schematic diagrams (see Figure 3-2 and Figure 3-3) for the USB-MODEVM interface board are provided for reference.

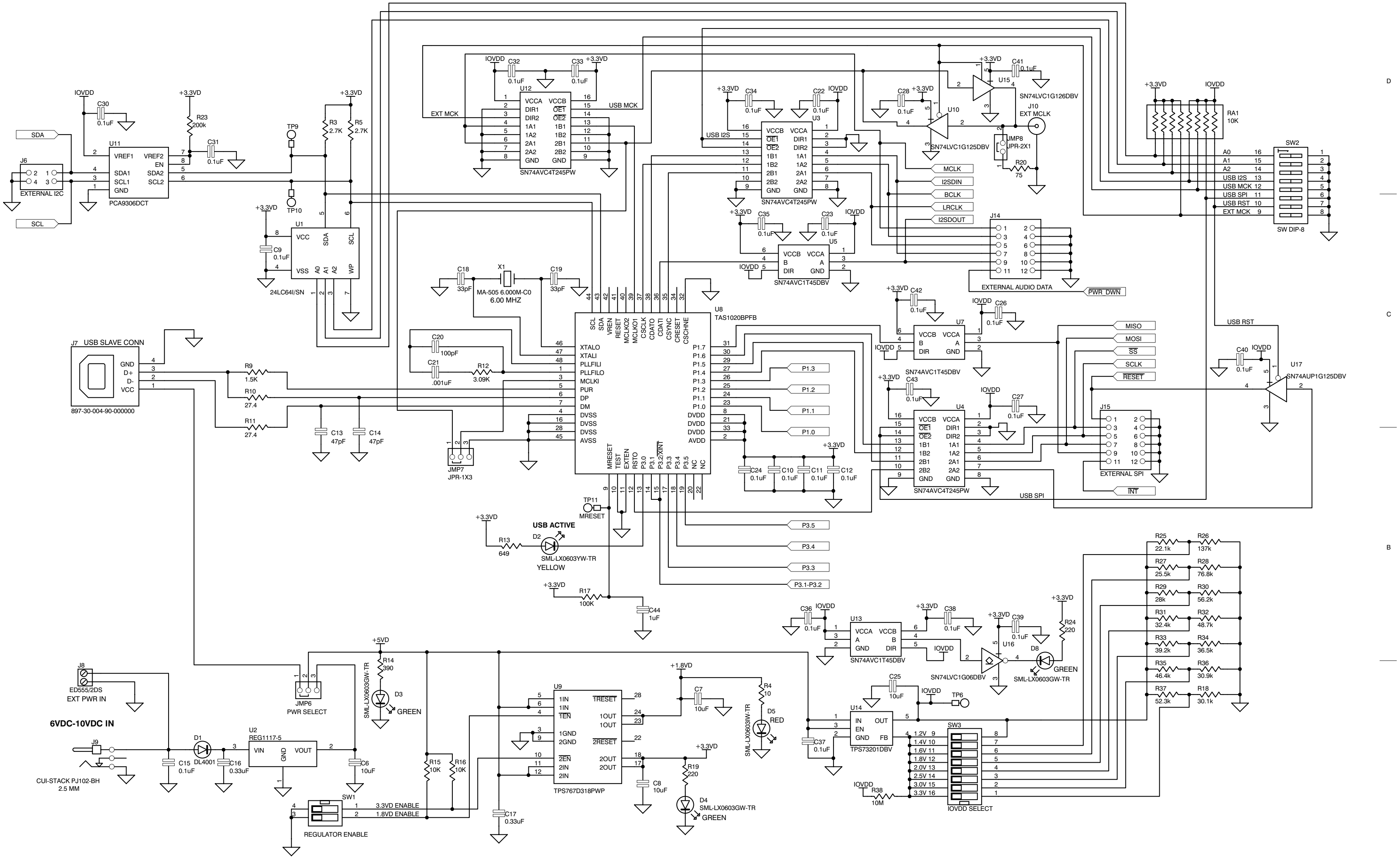



**Figure 3-2. USB-MODEVM Schematic**

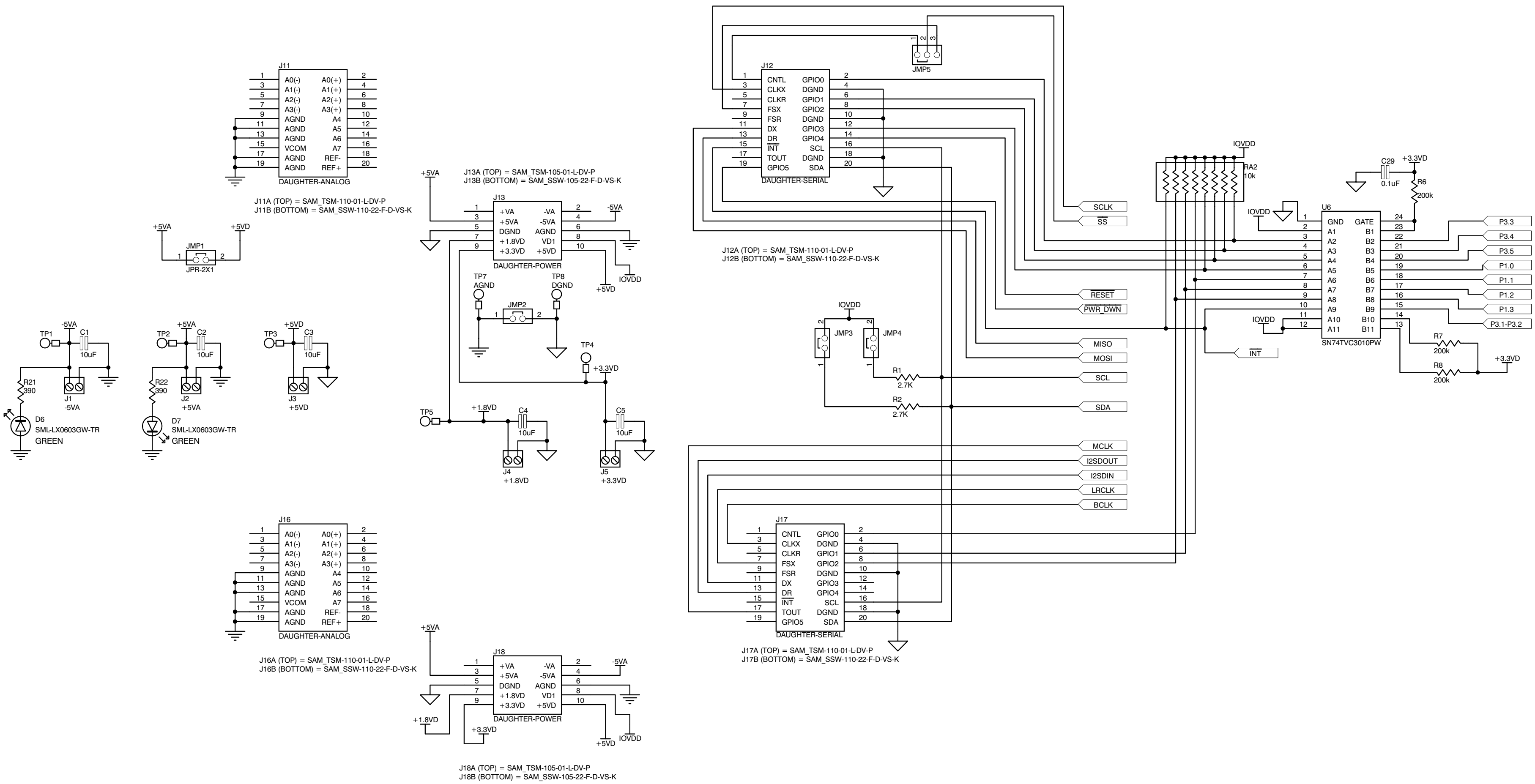



**Figure 3-3. USB-MODEVM Schematic 2**

### 3.3 USB-MODEVM Bill of Materials

The complete bill of materials, listed in Table 3-2, for the USB-MODEVM interface board (included only in the TLV320AIC3106EVM-PDK) is provided as a reference.

**Table 3-2. USB-MODEVM Bill of Materials**

| Designators | Description | Manufacturer | Mfg. Part Number |
|---|---|---|---|
| R4 | 10Ω 1/10W 5% chip resistor | Panasonic | ERJ-3GEYJ1300V |
| R10, R11 | 27.4Ω 1/16W 1% chip resistor | Panasonic | ERJ-3EKF27R4V |
| R20 | 75Ω 1/4W 1% chip resistor | Panasonic | ERJ-14NF75R0U |
| R19 | 220Ω 1/10W 5% chip resistor | Panasonic | ERJ-3GEYJ221V |
| R14, R21, R22 | 390Ω 1/10W 5% chip resistor | Panasonic | ERJ-3GEYJ391V |
| R13 | 649Ω 1/16W 1% chip resistor | Panasonic | ERJ-3EKF6490V |
| R9 | 1.5KΩ 1/10W 5% chip resistor | Panasonic | ERJ-3GEYJ1352V |
| R1–R3, R5–R8 | 2.7KΩ 1/10W 5% chip resistor | Panasonic | ERJ-3GEYJ272V |
| R12 | 3.09KΩ 1/16W 1% chip resistor | Panasonic | ERJ-3EKF3091V |
| R15, R16 | 10KΩ 1/10W 5% chip resistor | Panasonic | ERJ-3GEYJ1303V |
| R17, R18 | 100kΩ 1/10W 5% chip resistor | Panasonic | ERJ-3GEYJ1304V |
| RA1 | 10KΩ 1/8W octal isolated resistor array | CTS Corporation | 742C163103JTR |
| C18, C19 | 33pF 50V ceramic chip capacitor, ±5%, NPO | TDK | C1608C0G1H330J |
| C13, C14 | 47pF 50V ceramic chip capacitor, ±5%, NPO | TDK | C1608C0G1H470J |
| C20 | 100pF 50V ceramic chip capacitor, ±5%, NPO | TDK | C1608C0G1H101J |
| C21 | 1000pF 50V ceramic chip capacitor, ±5%, NPO | TDK | C1608C0G1H102J |
| C15 | 0.1µF 16V ceramic chip capacitor, ±10%, X7R | TDK | C1608X7R1C104K |
| C16, C17 | 0.33µF 16V ceramic chip capacitor, ±20%, Y5V | TDK | C1608X5R1C334K |
| C9–C12, C22–C28 | 1µF 6.3V ceramic chip capacitor, ±10%, X5R | TDK | C1608X5R0J1305K |
| C1–C8 | 10µF 6.3V ceramic chip capacitor, ±10%, X5R | TDK | C3216X5R0J1306K |
| D1 | 50V, 1A, diode MELF SMD | Micro Commercial Components | DL4001 |
| D2 | Yellow light emitting diode | Lumex | SML-LX0603YW-TR |
| D3– D7 | Green light emitting diode | Lumex | SML-LX0603GW-TR |
| D5 | Red light emitting diode | Lumex | SML-LX0603IW-TR |
| Q1, Q2 | N-channel MOSFET | Zetex | ZXMN6A07F |
| X1 | 6MHz crystal SMD | Epson | MA-505 6.000M-C0 |
| U8 | USB streaming controller | Texas Instruments | TAS1020BPFB |
| U2 | 5V LDO regulator | Texas Instruments | REG1117-5 |
| U9 | 3.3V/1.8V dual output LDO regulator | Texas Instruments | TPS767D318PWP |
| U3, U4 | Quad, 3-state buffers | Texas Instruments | SN74LVC125APW |
| U5–U7 | Single IC buffer driver with open drain o/p | Texas Instruments | SN74LVC1G07DBVR |
| U10 | Single 3-state buffer | Texas Instruments | SN74LVC1G125DBVR |
| U1 | 64K 2-wire serial EEPROM I2C | Microchip | 24LC64I/SN |
| | USB-MODEVM PCB | Texas Instruments | 6463995 |
| TP1–TP6, TP9–TP11 | Miniature test point terminal | Keystone Electronics | 5000 |
| TP7, TP8 | Multipurpose test point terminal | Keystone Electronics | 5011 |
| J7 | USB Type B slave connector thru-hole | Mill-Max | 897-30-004-90-000000 |

**Table 3-2. USB-MODEVM Bill of Materials (continued)**

| Designators | Description | Manufacturer | Mfg. Part Number |
|---|---|---|---|
| J13, J2–J5, J8 | 2-position terminal block | On Shore Technology | ED555/2DS |
| J9 | 2.5mm power connector | CUI Stack | PJ-102B |
| J130 | BNC connector, female, PC mount | AMP/Tyco | 414305-1 |
| J131A, J132A, J21A, J22A | 20-pin SMT plug | Samtec | TSM-110-01-L-DV-P |
| J131B, J132B, J21B, J22B | 20-pin SMT socket | Samtec | SSW-110-22-F-D-VS-K |
| J133A, J23A | 10-pin SMT plug | Samtec | TSM-105-01-L-DV-P |
| J133B, J23B | 10-pin SMT socket | Samtec | SSW-105-22-F-D-VS-K |
| J6 | 4-pin double row header (2x2) 0.1" | Samtec | TSW-102-07-L-D |
| J134, J135 | 12-pin double row header (2x6) 0.1" | Samtec | TSW-106-07-L-D |
| JMP1–JMP4 | 2-position jumper, 0.1" spacing | Samtec | TSW-102-07-L-S |
| JMP8–JMP14 | 2-position jumper, 0.1" spacing | Samtec | TSW-102-07-L-S |
| JMP5, JMP6 | 3-position jumper, 0.1" spacing | Samtec | TSW-103-07-L-S |
| JMP7 | 3-position dual row jumper, 0.1" spacing | Samtec | TSW-103-07-L-D |
| SW1 | SMT, half-pitch 2-position switch | C&K Division, ITT | TDA02H0SK1 |
| SW2 | SMT, half-pitch 8-position switch | C&K Division, ITT | TDA08H0SK1 |
| | Jumper plug | Samtec | SNT-100-BK-T |

## 3.4 USB-MODEVM Protocol

The USB-MODEVM is defined as a vendor-specific class, and is identified on the PC system as an NI-VISA device. Because the TAS1020 has several routines in its ROM that are designed for use with HID-class devices, HID-like structures are used, even though the USB-MODEVM is not an HID-class device. Data are passed from the PC to the TAS1020 using the control endpoint.

As Table 3-3 describes, data are sent in an HIDSETREPORT.

**Table 3-3. USB Control Endpoint HIDSETREPORT Request**

| Part | Value | Description |
|---|---|---|
| bmRequestType | 0x21 | 00100001 |
| bRequest | 0x09 | SET_REPORT |
| wValue | 0x00 | don't care |
| wIndex | 0x03 | HID interface is index 3 |
| wLength | calculated by host | |
| Data | | Data packet as described below |

Table 3-4 lists the bytes that comprise the data packet.

**Table 3-4. Data Packet Configuration**

| Byte Number | Type | Description | |
|---|---|---|---|
| 0 | Interface | Specifies the serial interface and operation. The two values are logically OR'd. | |
| | | Operation: | |
| | | READ | 0x00 |
| | | WRITE | 0x10 |
| | | Interface: | |
| | | GPIO | 0x08 |
| | | SPI_16 | 0x04 |
| | | I2C_FAST | 0x02 |
| | | I2C_STD | 0x01 |
| | | SPI_8 | 0x00 |
| 1 | $I^2C$ slave address | Slave address of the $I^2C$ device or the MSB of the 16-bit register address for SPI | |
| 2 | Length | Length of data to write/read (number of bytes) | |
| 3 | Register address | Address of the register for $I^2C$ or 8-bit SPI; LSB of the 16-bit address for SPI | |
| 4...64 | Data | Up to 60 data bytes can be written at a time. EP0 maximum length is 64. The return packet is limited to 42 bytes, so advise only sending 32 bytes at any one time. | |

Example usages:

Write two bytes (AA, 55) to the device starting at register 5 of an $I^2C$ device with address A0:

```
[0] 0x11
[1] 0xA0
[2] 0x02
[3] 0x05
[4] 0xAA
[5] 0x55
```

Do the same with a fast mode $I^2C$ device:

```
[0] 0x12
[1] 0xA0
[2] 0x02
[3] 0x05
[4] 0xAA
[5] 0x55
```

Now do the same with an SPI device that uses an 8-bit register address:

```
[0] 0x10
[1] 0xA0
[2] 0x02
[3] 0x05
[4] 0xAA
[5] 0x55
```

Do the same process for a 16-bit register address, as found on parts like the TSC2101. Assume the register address (command word) is **0x10E0**:

```
[0] 0x14
[1] 0x10
```

**Note**

The $I^2C$ address now serves as the MSB of the register address.

```
[2] 0x02
[3] 0xE0
[4] 0xAA
[5] 0x55
```

In each case, the TAS1020 returns, in an HID interrupt packet, the following:

| | |
|---|---|
| [0] | interface byte \| status |
| status: | |
| | `REQ_ERROR 0x80` |
| | `INTF_ERROR 0x40` |
| | `REQ_DONE 0x20` |
| [1] | For $I^2C$ interfaces, the $I^2C$ address is as sent |
| | For SPI interfaces, the read back data from the SPI line for transmission of the corresponding byte is as sent |
| [2] | Length is as sent |
| [3] | For $I^2C$ interfaces, the register address is as sent |
| | For SPI interfaces, the read back data from the SPI line for transmission of the corresponding byte is as sent |
| [4..60] | The echo of the data packet is sent |

If the command is sent with no problem, the returning byte [0] must be the same as the sent one logically or'd with 0x20. In the first example usage, the returning packet is:

```
[0] 0x31
[1] 0xA0
[2] 0x02
[3] 0x05
[4] 0xAA
[5] 0x55
```

If for some reason the interface fails (for example, the $I^2C$ device does not acknowledge), the returning byte comes back as:

`[0] 0x51` → interface | INTF_ERROR

```
[1] 0xA0
[2] 0x02
[3] 0x05
[4] 0xAA
[5] 0x55
```

If the request is malformed, that is, the interface byte (byte [0]) takes on a value that is not described above, the return packet is:

`[0] 0x93` → the user sent 0x13, which is not valid, so 0x93 is returned

```
[1] 0xA0
[2] 0x02
[3] 0x05
[4] 0xAA
[5] 0x55
```

The examples above used writes. Reading is similar:
Read two bytes from the device starting at register 5 of an $I^2C$ device with address A0:

```
[0] 0x01
[1] 0xA0
[2] 0x02
[3] 0x05
```

The return packet is:

```
[0] 0x21
[1] 0xA0
[2] 0x02
[3] 0x05
[4] 0xAA
[5] 0x55
```

This result is assuming that the values written above starting at register 5 were actually written to the device.

### 3.5 GPIO Capability

The USB-MODEVM has seven GPIO lines. Access them by specifying the interface to be 0x08, and then using the standard format for packets—but addresses are unnecessary. The GPIO lines, as shown in Table 3-5 are mapped into one byte.

**Table 3-5. GPIO Pin Assignments**

| Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| x | P3.5 | P3.4 | P3.3 | P1.3 | P1.2 | P1.1 | P1.0 |

Example: write P3.5 to a 1, set all others to 0:

`[0] 0x18` → Write, GPIO

`[1] 0x00` → This value is ignored

`[2] 0x01` → Length is always a 1

`[3] 0x00` → This value is ignored

`[4] 0x40` → 01000000

The user may also read back from the GPIO to see the state of the pins. For this example, assume the previous example was written to the port pins.

Example: read the GPIO

`[0] 0x08` → Read, GPIO

`[1] 0x00` → This value is ignored

`[2] 0x01` → Length is always a 1

`[3] 0x00` → This value is ignored

The return packet is:

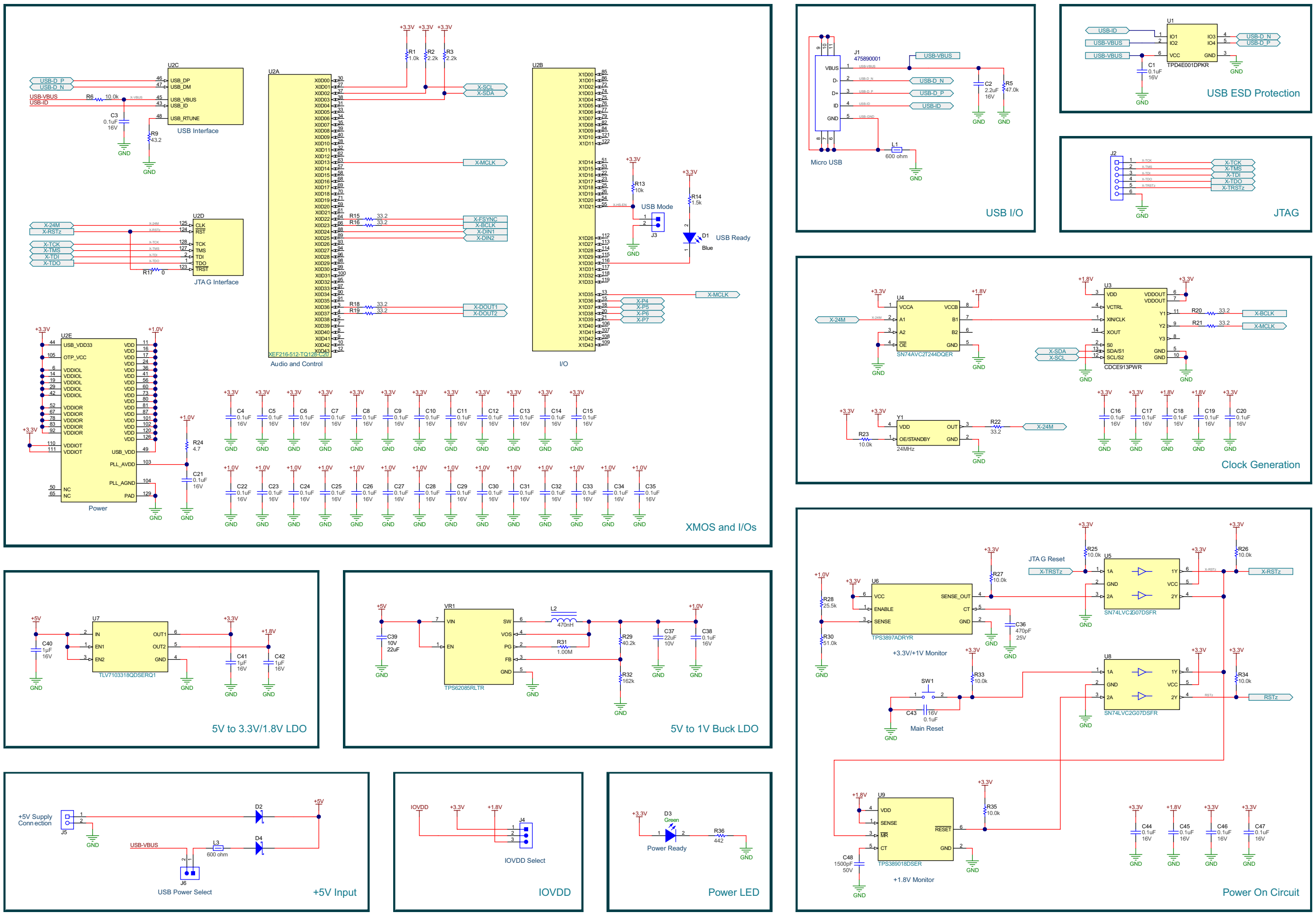
```
[0] 0x28
[1] 0x00
[2] 0x01
[3] 0x00
[4] 0x40
```

## 3.6 AC-MODEVM Schematic

The schematic diagrams for the AC-MODEVM interface board are provided for reference.

AC-MODEVM-LPA001A

XMOS, USB and Power



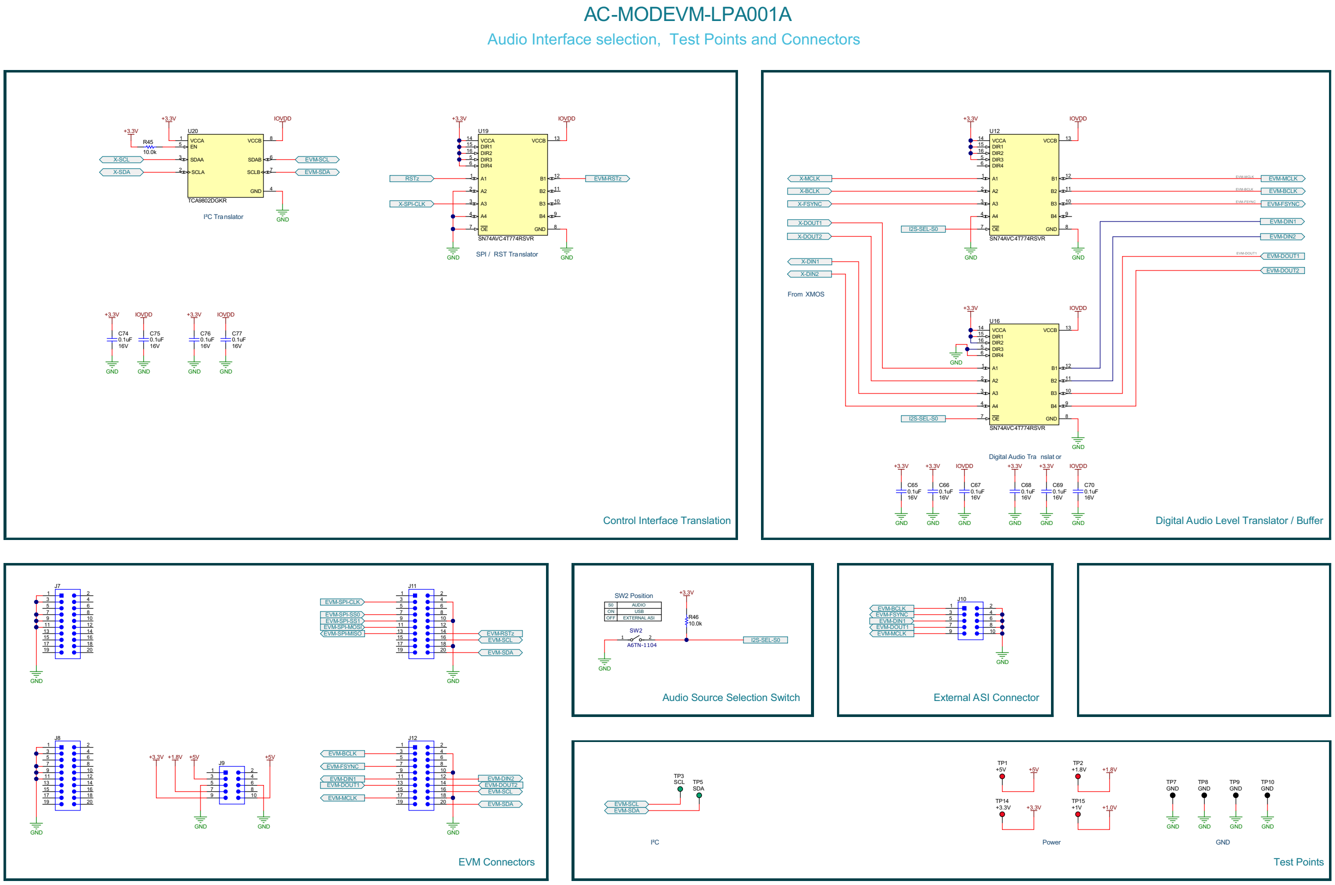



**Figure 3-4. AC-MODEVM Schematic**

## 3.7 AC-MODEVM Bill of Materials

The complete bill of materials, listed in , for the AC-MODEVM interface board (included only in the TLV320AIC3106EVM-PDK) is provided as a reference.

**Table 3-6. AC-MODEVM Bill of Materials**

| Designators | Description | Manufacturer | Mfg. Part Number |
|---|---|---|---|
| C1, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C16, C17, C18, C19, C20, C21, C22, C23, C24, C25, C26, C27, C28, C29, C30, C31, C32, C33, C34, C35, C38, C43, C44, C45, C46, C47, C65, C66, C67, C68, C69, C70, C74, C75, C76, C77 | CAP, CERM, 0.1 uF, 16 V, +/- 10%, X7R, 0402 | Wurth Elektronik | 885012205037 |
| C2 | CAP, CERM, 2.2 uF, 16 V, +/- 10%, X7R, 0603 | Taiyo Yuden | EMK107BB7225KA-T |
| C36 | CAP, CERM, 470 pF, 25 V, +/- 5%, C0G/NP0, 0402 | MuRata | GRM1555C1E471JA01D |
| C37, C39 | CAP, CERM, 22 uF, 10 V, +/- 20%, X7R, 0805 | MuRata | GRM21BZ71A226ME15L |
| C40, C41, C42 | CAP, CERM, 1 µF, 16 V,+/- 10%, X7R, AEC-Q200 Grade 1, 0603 | TDK | CGA3E1X7R1C105K080AC |
| C48 | CAP, CERM, 1500 pF, 50 V, +/- 10%, X7R, 0402 | MuRata | GRM155R71H152KA01D |
| D1 | LED, Blue, SMD | Wurth Elektronik | 150060BS75000 |

**Table 3-6. AC-MODEVM Bill of Materials (continued)**

| Designators | Description | Manufacturer | Mfg. Part Number |
|---|---|---|---|
| D2, D4 | Diode, Schottky, 20 V, 1 A, SOD-123FL | ON Semiconductor | MBR120LSFT1G |
| D3 | LED, Green, SMD | Lite-On | LTST-C170KGKT |
| FID4, FID5, FID6 | Fiducial mark. There is nothing to buy or mount. | N/A | N/A |
| H9, H10, H11, H12 | Bumpon, Hemisphere, 0.44 X 0.20, Clear | 3M | SJ-5303 (CLEAR) |
| J1 | Connector, Receptacle, Micro-USB Type AB, R/A, Bottom Mount SMT | Molex | 475890001 |
| J2 | Receptacle, 50mil, 6x1, Gold, R/A, TH | Sullins Connector Solutions | LPPB061NGCN-RC |
| J3, J6 | Header, 2.54 mm, 2x1, Tin, TH | Samtec | TSW-102-07-T-S |
| J4 | Header, 100mil, 3x1, Gold, TH | Samtec | TSW-103-07-G-S |
| J5 | Terminal Block, 3.5mm Pitch, 2x1, TH | On-Shore Technology | ED555/2DS |
| J7, J8, J11, J12 | Header, 2.54mm, 10x2, Tin, TH | Sullins Connector Solutions | PEC10DAAN |
| J9, J10 | Header, 2.54mm, 5x2, Gold, Black, TH | Samtec | TSW-105-07-F-D |
| L1, L3 | Ferrite Bead, 600 ohm @ 100 MHz, 2 A, 0805 | TDK | MPZ2012S601AT000 |
| L2 | Inductor, Shielded, Ferrite, 470 nH, 2 A, 0.059 ohm, SMD | TDK | VLS2012ET-R47N |
| R1 | RES, 1.0 k, 5%, 0.063 W, AEC-Q200 Grade 0, 0402 | Vishay-Dale | CRCW04021K00JNED |
| R2, R3 | RES, 2.2 k, 5%, 0.063 W, AEC-Q200 Grade 0, 0402 | Vishay-Dale | CRCW04022K20JNED |
| R5 | RES, 47.0 k, 1%, 0.0625 W, 0402 | Yageo America | RC0402FR-0747KL |
| R6 | RES, 10.0 k, 1%, 0.1 W, 0402 | Panasonic | ERJ-2RKF1002X |
| R9 | RES, 43.2, 1%, 0.063 W, AEC-Q200 Grade 0, 0402 | Vishay-Dale | CRCW040243R2FKED |
| R13 | RES, 10 k, 5%, 0.063 W, AEC-Q200 Grade 0, 0402 | Vishay-Dale | CRCW040210K0JNED |
| R14 | RES, 1.5 k, 5%, 0.1 W, AEC-Q200 Grade 0, 0603 | Vishay-Dale | CRCW06031K50JNEA |
| R15, R16, R18, R19, R20, R21, R22 | RES, 33.2, 1%, 0.05 W, 0201 | Yageo America | RC0201FR-0733R2L |
| R17 | RES, 0, 5%, .05 W, AEC-Q200 Grade 0, 0201 | Panasonic | ERJ-1GN0R00C |
| R23, R25, R26, R27, R33, R34, R35, R45, R46 | RES, 10.0 k, 1%, 0.05 W, 0201 | Vishay-Dale | CRCW020110K0FKED |
| R24 | RES, 4.7, 5%, 0.1 W, AEC-Q200 Grade 0, 0603 | Vishay-Dale | CRCW06034R70JNEA |
| R28 | RES, 25.5 k, 1%, 0.05 W, 0201 | Yageo America | RC0201FR-0725K5L |
| R29 | RES, 40.2 k, 1%, 0.063 W, AEC-Q200 Grade 0, 0402 | Vishay-Dale | CRCW040240K2FKED |
| R30 | RES, 51.0 k, 1%, 0.05 W, 0201 | Yageo America | RC0201FR-0751KL |
| R31 | RES, 1.00 M, 1%, 0.125 W, AEC-Q200 Grade 0, 0805 | Vishay-Dale | CRCW08051M00FKEA |
| R32 | RES, 162 k, 1%, 0.063 W, AEC-Q200 Grade 0, 0402 | Vishay-Dale | CRCW0402162KFKED |
| R36 | RES, 442, 1%, 0.1 W, AEC-Q200 Grade 0, 0603 | Vishay-Dale | CRCW0603442RFKEA |
| SH-J1, SH-J2 | Shunt, 100mil, Gold plated, Black | Samtec | SNT-100-BK-G |
| SW1 | Switch, Tactile, SPST-NO, 0.05A, 12V, SMT | E-Switch | TL1015AF160QG |
| SW2 | Dip Switch SPST 1 Position Through Hole Slide (Standard) Actuator 25mA 24VDC | Omron Electronics Inc-EMC Div | A6TN-1104 |
| TP1, TP2, TP14, TP15 | Test Point, Miniature, Red, TH | Keystone | 5000 |
| TP3, TP5 | Test Point, Miniature, Green, TH | Keystone | 5116 |
| TP7, TP8, TP9, TP10 | Test Point, Multipurpose, Black, TH | Keystone | 5011 |

**Table 3-6. AC-MODEVM Bill of Materials (continued)**

| Designators | Description | Manufacturer | Mfg. Part Number |
|---|---|---|---|
| U1 | Low-Capacitance + / - 15-kV ESD Protection Array for High-Speed Data Interfaces, 4 Channels, -40 to +85 degC, 6-pin USON (DPK), Green (RoHS & no Sb/Br) | Texas Instruments | TPD4E001DPKR |
| U2 | IC MCU 512KB RAM, 128TQFP | XMOS semiconductor | XEF216-512-TQ128-C20 |
| U3 | Programmable 1-PLL VCXO Clock Synthesizer with 2.5-V or 3.3-V LVCMOS Outputs, PW0014A (TSSOP-14) | Texas Instruments | CDCE913PWR |
| U4 | Dual-Bit Dual-Supply Bus Transceiver, DQE0008A, LARGE T&R | Texas Instruments | |
| U5, U8 | Enhanced Product Dual Buffer/Driver with Open-Drain Output, DCK0006A (SOT-SC70-6) | Texas Instruments | SN74LVC2G07DSFR |
| U6 | Single-Channel Ultra-Small Adjustable Supervisory Circuit With Active-High Open-Drain Output, DRY0006A (USON-6) | Texas Instruments | TPS3897ADRYR |
| U7 | Automotive Catalog, Dual, 200mA, Low-IQ Low-Dropout Regulator for Portable Devices, DSE0006A (WSON-6) | Texas Instruments | TLV7103318QDSERQ1 |
| U9 | Low-Quiescent-Current 1% Accurate Supervisor With Programmable Delay, DSE0006A (WSON-6) | Texas Instruments | TPS389018DSER |
| U12, U16, U19 | 4-Bit Dual-Supply Bus Transceiver With Configurable Voltage-Level Shifting and 3-State Outputs, RSV0016A (UQFN-16) | Texas Instruments | SN74AVC4T774RSVR |
| U20 | Level-Translating I2C Bus Buffer/Repeater, DGK0008A (VSSOP-8) | Texas Instruments | TCA9802DGKR |
| VR1 | 3-A Step-Down Converter with DCS-Control and Hiccup Short Circuit Protection in 2x2 HotRod Package, RLT0007A (VSON-HR-7) | Texas Instruments | TPS62085RLTR |
| Y1 | OSC, 24 MHz, 2.25 - 3.63 V, SMD | Abracon Corporation | ASTMLPA-24.000MHZ-EJ-E-T |
| FID1, FID2, FID3 | Fiducial mark. There is nothing to buy or mount. | N/A | N/A |

## 3.8 Writing Scripts

A script is simply a text file that contains data to send to the serial control buses. The scripting language is quite simple, as is the parser for the language. Therefore, the program is not very forgiving about mistakes made in the source script file, but the formatting of the file is simple. Consequently, mistakes tend to be rare.

Each line in a script file is one command. There is no provision for extending lines beyond one line. A line is terminated by a carriage return.

The first character of a line is the command. Commands are listed in Table 3-7.

**Table 3-7. Commands**

| Command | Description |
|---|---|
| **i** | Set interface bus to use |
| **r** | Read from the serial control bus |
| **w** | Write to the serial control bus |
| **#** | Comment |
| **b** | Break |
| **d** | Delay |

The first command, **i**, sets the interface to use for the commands to follow. This command must be followed by one of the parameters listed in Table 3-8.

**Table 3-8. Command Parameters**

| Command | Description |
|---|---|
| **i2cstd** | Standard mode $I^2C$ Bus |
| **i2cfast** | Fast mode $I^2C$ bus |
| **spi8** | SPI bus with 8-bit register addressing |
| **spi16** | SPI bus with 16-bit register addressing |
| **gpio** | Use the USB-MODEVM GPIO capability |

For example, if a fast mode $I^2C$ bus is used, the script begins with **i i2cfast**.

No data follows the break command. The text following a comment command is ignored by the parser, provided that the text is on the same line. The delay command allows the user to specify a time, in milliseconds, that the script pauses before proceeding.

**Note**

Unlike all other numbers used in the script commands, the delay time is entered in a decimal format. Also, because of latency in the USB bus as well as the time required for the processor on the USB-MODEVM to handle requests, the delay time may not be precise.

A series of byte values follows either a read or write command. Each byte value is expressed in hexadecimal, and each byte must be separated by a space. Commands are interpreted and sent to the TAS1020 by the program using the protocol described in Section 3.4.

The first byte following a read or write command is the $I^2C$ slave address of the device (if $I^2C$ is used) or the first data byte to write (if SPI is used, SPI interfaces are not standardized on protocols, so the meaning of this byte varies with the device being addressed on the SPI bus). The second byte is the starting register address that data are written to (with $I^2C$ the SPI varies, see Section 3.4 for additional information about what variations may be necessary for a particular SPI mode). If reading, data follow these two bytes. If reading, the third byte value is the number of bytes to read, (expressed in hexadecimal).

For example, to write the values 0xAA 0x55 to an $I^2C$ device with a slave address of 0x90, starting at a register address of 0x03, write:

```
#example script
i i2cfast
w 90 03 AA 55
r 90 03 2
```

This script begins with a comment, specifying that a fast $I^2C$ bus will be used, then writes *0xAA 0x55* to the $I^2C$ slave device at address 0x90, and writes the values into registers 0x03 and 0x04. The script then reads back two bytes from the same device starting at register address 0x03. The slave device value does not change. The R/$\overline{W}$ bit does not need to be set for $I^2C$ devices in the script; the read or write commands will do that.

Here is an example of using an SPI device that requires 16-bit register addresses:

```
# setup TSC2101 for input and output

# uses SPI16 interface

# this script sets up DAC and ADC at full volume, input from onboard mic

#

# Page 2: Audio control registers

w 10 00 00 00 80 00 00 00 45 31 44 FD 40 00 31 C4

w 13 60 11 20 00 00 00 80 7F 00 C5 FE 31 40 7C 00 02 00 C4 00 00 00 23 10 FE 00 FE 00
```

Blank lines are allowed. However, be sure that the script does not end with a blank line. Although ending with a blank line does not cause the script to fail, the program executes that line, and therefore, may prevent the user from seeing data written or read back on the previous command.

In this example, the first two bytes of each command are the command word to send to the TSC2101 (0x1000, 0x1360); these are followed by data to write to the device starting at the address specified in the command word. The second line may wrap in the viewer being used to look like more than one line; careful examination will show, however, that there is only one carriage return on that line, following the last **00**.

Any text editor may be used to write these scripts; Jedit is an editor that is highly recommended for general usage. For more information, go to: http://www.jedit.org.

Once the script is written, this script can be used in the command window by running the program, and then selecting *Open Command File...* from the File menu. Locate and open the script. The script is then displayed in the command buffer. The user may also edit the script when in the buffer, but saving of the command buffer is not possible at this time (this feature may be added at a later date).

Once the script is in the command buffer, the script may be executed by pressing the *Execute Command Buffer* button. If there are breakpoints in the script, the script executes to that point, and the user is presented with a dialog box with a button to press to continue executing the script. When ready to proceed, push that button and the script continues.

Here an example of a (partial) script with breakpoints:

```
# setup AIC33 for input and output
# uses I2C interface
i i2cfast
# reg 07 - codec datapath
w 30 07 8A
r 30 07 1
d 1000
# regs 15/16 - ADC volume, unmute and set to 0dB
w 30 0F 00 00
r 30 0F 2
b
```

This script writes the value 8A at register 7, then reads the register back to verify that the write was good. A delay of 1000 ms (one second) is placed after the read to pause the script operation. When the script continues, the values **00 00** are written starting at register 0F. This output is verified by reading two bytes, and pausing the script again, this time with a break. The script does not continue until the user presses *OK* in the dialog box that will be displayed because of the break.

# 4 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.