# TI Designs: TIDA-060001
# SunSpec® Rapid Shutdown Transmit and Receive Reference Design

## Description

This reference design interfaces an AFE031 Powerline Communications Analog Front End with a C2000™ MCU to send and receive data over a wired coupled interface using frequency shift keying (FSK). The design demonstrates the SunSpec® standard protocol transmitting the specific 33-bit word packet using FSK ($f_M$: 131.25 kHz and $f_S$: 143.75 kHz). The purpose is to provide a keepalive signal with an integrated system solution that communicates from the solar inverter to photovoltaic (PV) module.

## Resources

| | |
|---|---|
| TIDA-060001 | Design Folder |
| AFE031 | Product Folder |
| TMS320F28379D LaunchPad™ | Product Folder |
| TPS62177 | Product Folder |
| SN74LVC2G07 | Product Folder |

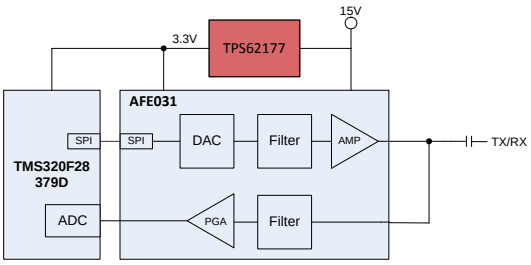
TI E2E™ Community — ASK Our E2E™ Experts

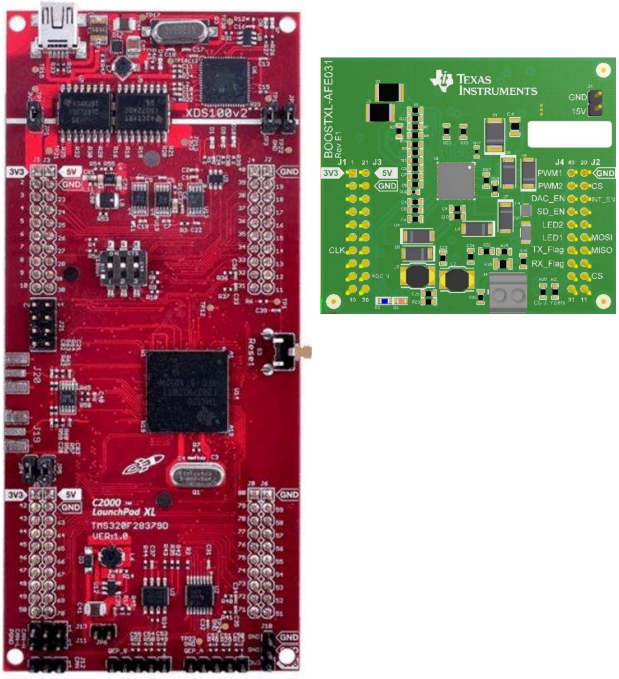
## Features

- Integrated powerline driver
- Programmable Tx/Rx filters and Tx/Rx gain control
- Preprogrammed open source example code for SunSpec protocol
- High-resolution PWM
- Supports EN50065 CENELEC bands A, B, C, D
- Supports FSK, S-FSK, and OFDM

## Applications

- Rapid Shutdown PLC for Solar Inverters
- Rapid Shutdown PLC for PV Modules
- PLC FSK on DC line



# 1 System Description

This reference design interfaces an AFE031 with a C2000 MCU to send and receive data over a wired coupled interface using frequency shift keying (FSK). The design demonstrates the SunSpec standard protocol transmitting the specific 33-bit word packet using FSK ($f_M$: 131.25 kHz and $f_S$: 143.75 kHz). The purpose is to provide a keepalive signal with an integrated system solution that communicates from the string inverter to a photovoltaic (PV) module. This design guide addresses component selection, design theory, and test results of the TIDA-060001 reference design. The following subsections describe the various devices within this reference design system and highlight the characteristics most critical in implementing the corresponding functions.

## *1.1 Key System Specifications*

Table 1 describes the parameters around the FSK signal that is generated in the software example. The end goal is to send the complete packet, wait the designated wait period, and repeat.

**Table 1. SunSpec® FSK Specifications**

| SYMBOL | TX SPECIFICATION | MIN | NOM | MAX | UNIT | COMMENT |
|---|---|---|---|---|---|---|
| W1 | Logic 1 code word | {{-1,-1,-1,+1,+1,+1,-1,+1,+1,-1,+1}} | | | | +1 = Mark, -1 = Space |
| W0 | Logic 0 code word | {{+1,+1,+1,-1,-1,-1,+1,-1,-1,+1,-1}} | | | | +1 = Mark, -1 = Space |
| packet_1 | Complete packet | ABC = [W1,W1,W1] | | | | |
| packet_0 | Complete packet | ABC = [W0,W0,W0] | | | | |
| Fm | Mark frequency | 131.236875 | 131.25 | 131.263125 | kHz | |
| Fs | Space frequency | 143.735625 | 143.75 | 143.764375 | kHz | |
| Ts | Bit period | 5.119488 | 5.12 | 5.120512 | ms | |
| Tt | Tx period | 168.943104 | 168.96 | 168.976896 | ms | 3 words |
| Tq | Quiet period | 901.029888 | 901.12 | 901.210112 | ms | 16 words |
| Tc | Cycle period | 1069.972992 | 1070.08 | 1070.187008 | ms | 19 words |

# 2 System Overview

## 2.1 Block Diagram

Figure 1 shows a block diagram of the system. In this example, not all of the connections are used, but all are present on the BoosterPack™ plug-in module for future development.

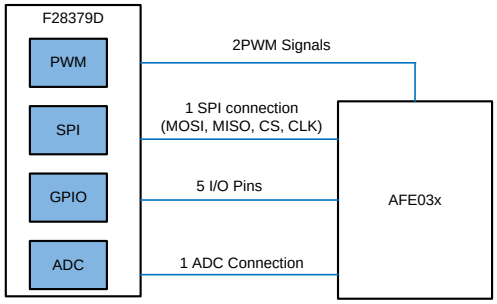



**Figure 1. C2000™ and AFE031 Block Diagram**

The AFE has multiple internal registers that allow configuration of the internal components of the AFE chip, including filter selection, gain selection, and mode selections. These registers can be accessed using the SPI peripheral.

The AFE also has various GPIOs that allow the MCU to set the AFE into certain modes, as well as receive interrupts for critical events on the AFE. The ADC connection allows the MCU to receive or sample an input signal. The PWM signals provide a way to create an output for the AFE. Currently, the AFE031 supports two modes of data transmission: PWM mode and DAC mode.

## 2.2 Highlighted Products

This reference design features the following devices:

- AFE031: Powerline Communication Analog Front-End
- LAUNCHXL-F28379D: C2000 Delfino MCUs F28379D LaunchPad Development Kit
- TPS62177: 28-V, 0.5-A, Step-Down Converter With Sleep Mode
- SN74LVC2G07: Dual Buffer and Driver With Open-Drain Outputs

For more information on each of these devices, see their respective product folders at TI.com.

### 2.2.1 AFE031

The AFE031 is a low-cost, integrated, powerline communication (PLC) analog front-end (AFE) device that is capable of capacitive or transformer coupled connections to the powerline while under control of a DSP or microcontroller (MCU). This device is ideal for driving low-impedance lines that require up to 1.5 A into reactive loads. The integrated receiver is able to detect signals down to 20 $\mu V_{RMS}$ and is capable of a wide range of gain options to adapt to varying input signal conditions. This monolithic integrated circuit provides high reliability in demanding PLC applications.

The AFE031 transmit power amplifier operates from a single supply in the range of 7 V to 24 V. At maximum output current, a wide output swing provides a 12-$V_{PP}$ ($I_{OUT}$ = 1.5 A) capability with a nominal 15-V supply. The analog and digital signal processing circuitry operates from a single 3.3-V power supply.

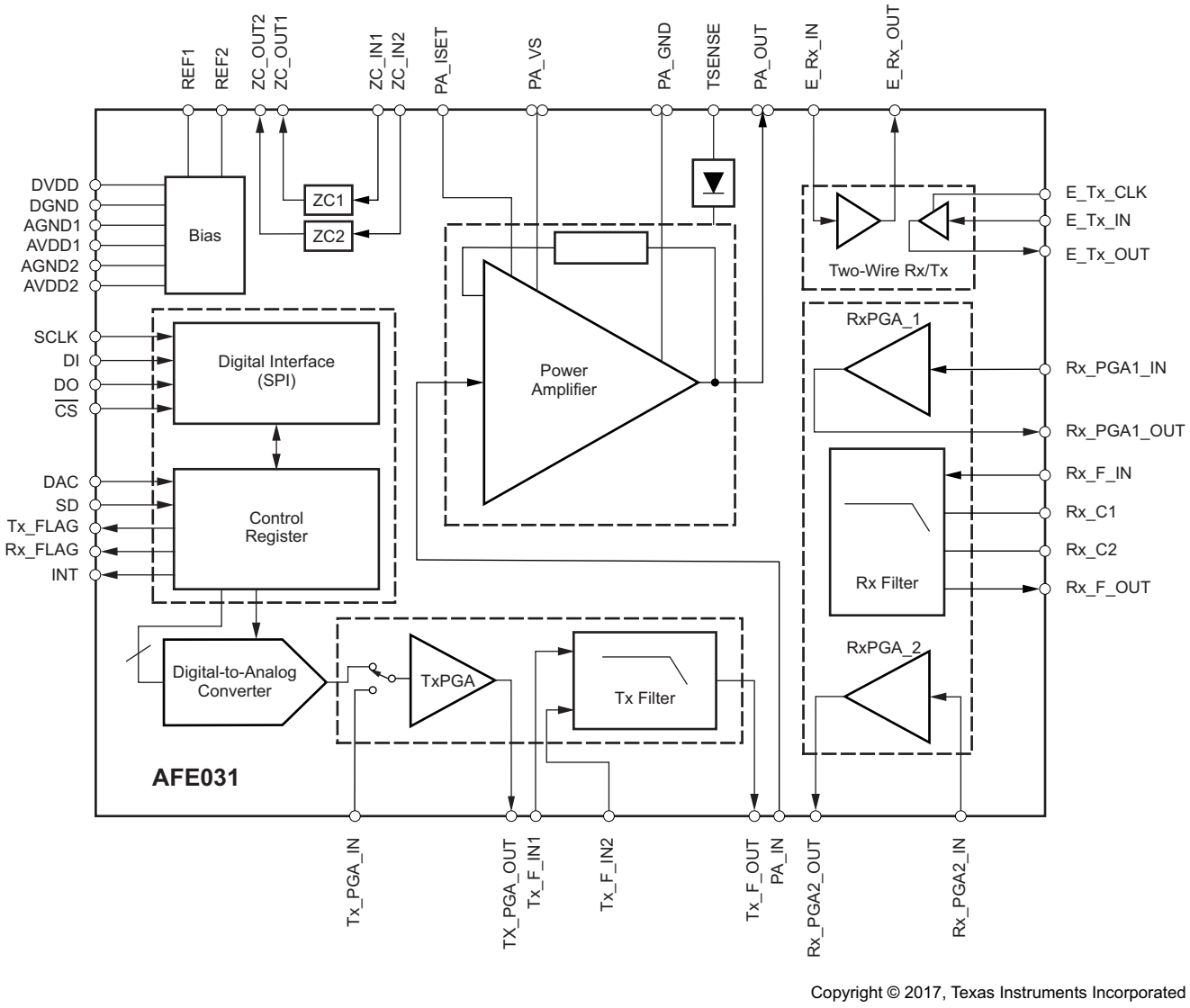





**Figure 2. Functional Block Diagram of AFE031**

### 2.2.2 LAUNCHXL-F28379D

The TMS320F2837xD MCU family, referred to as the F2837xD in this design guide, is a dual-core MCU design based on the TI 32-bit C28x CPU architecture. Each core is identical with access to its own local RAM and flash memory, as well as globally shared RAM. Sharing information between the two CPU cores is accomplished with an inter-processor communications (IPC) module. Additionally, each core shares access to a common set of highly integrated analog and control peripherals, providing a complete solution for demanding real-time high-performance signal processing applications, such as digital power, industrial drives, inverters, and motor control.

In addition to the high-performance CPUs, each core has a control law accelerator (CLA), which is an independent 32-bit floating-point processor designed to execute math-intensive calculations. The CLA runs concurrently and at the same speed of the main CPU, thereby effectively doubling the computational performance of each core. With each CPU running at 200 MHz, the CLAs can then effectively boost the total performance of the device to 800 MIPS.

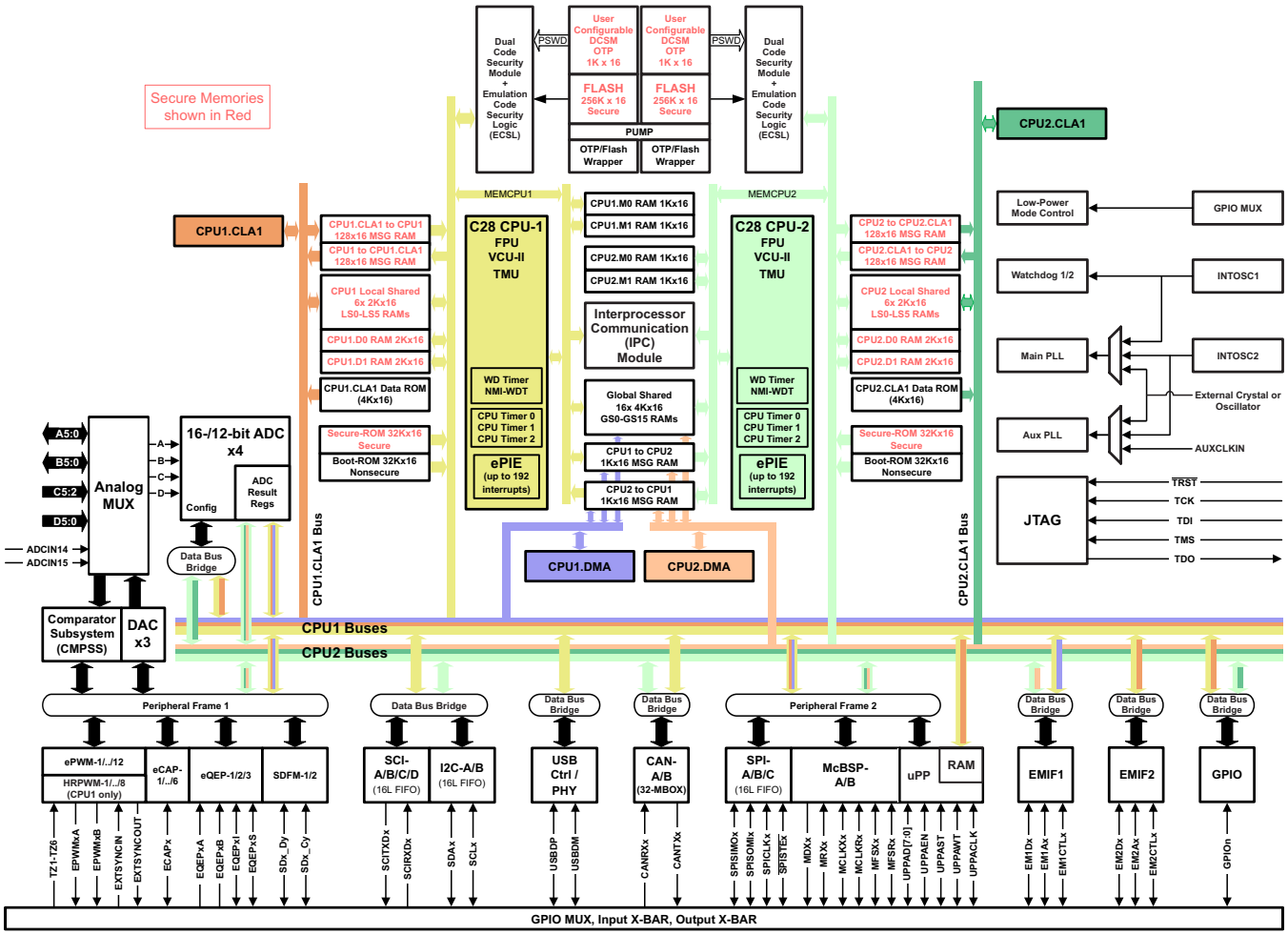





**Figure 3. Functional Block Diagram of LAUNCHXL-F28379D**

### 2.2.3 TPS62177

The TPS6217x is a high efficiency synchronous step-down DC/DC converter, based on the DCS-Control™ topology. With a wide operating input voltage range of 4.75 V to 28 V, the device is ideally suited for systems powered from multi-cell Li-Ion as well as 12 V and even higher intermediate supply rails, providing up to 500-mA output current.

The device features a typical quiescent current of 22 µA in normal mode and 4.8 µA in sleep mode. In sleep mode, the efficiency at very low load currents can be increased by as much as 20%. In shutdown mode, the shutdown current is less than 2 µA and the output is actively discharged.

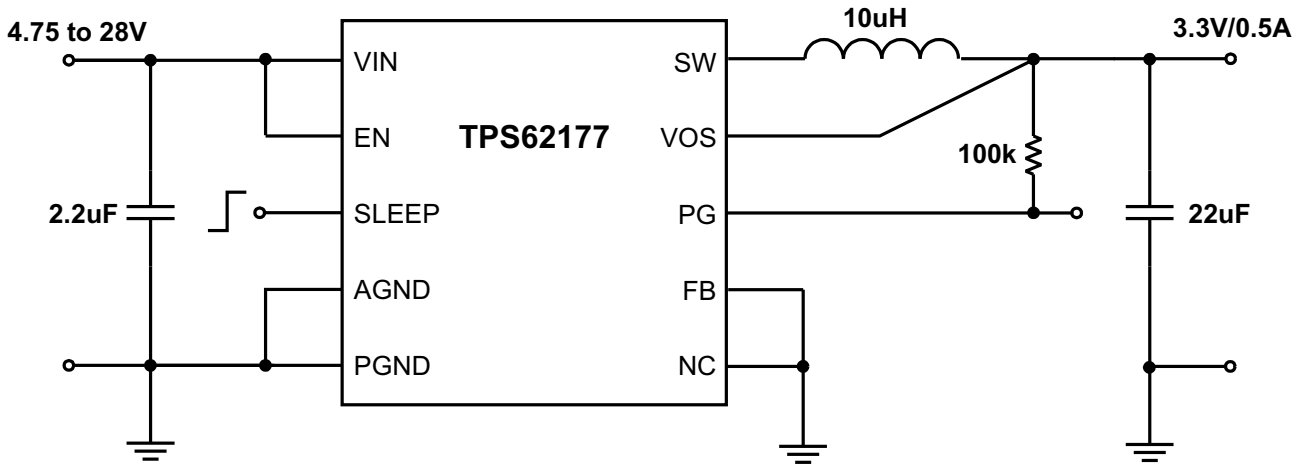



**Figure 4. Typical Application Schematic of TPS62177**

### 2.2.4 SN74LVC2G07

This dual buffer and driver is designed for 1.65-V to 5.5-V VCC operation. The output of the SN74LVC2G07 device is open drain and can be connected to other open-drain outputs to implement active-low wired-OR or active-high wired-AND functions. The maximum sink current is 32 mA.

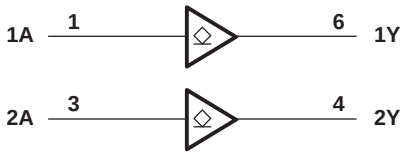



**Figure 5. Functional Block Diagram of SN74LVC2G07**

## 2.3 System Design Theory

FSK is a modulation scheme that utilizes discrete changes of frequency to transmit and receive digital data. One of the simplest subsections of this modulation scheme, as well as the modulation used in this demo, is called binary frequency shift keying (BFSK).

In this scheme, the system is switching between two discrete frequencies: the mark frequency ("1") and the space frequency ("0"). These frequencies correlate directly to the bit value of the transmitted data.

Figure 6 shows what this looks like in the time domain.

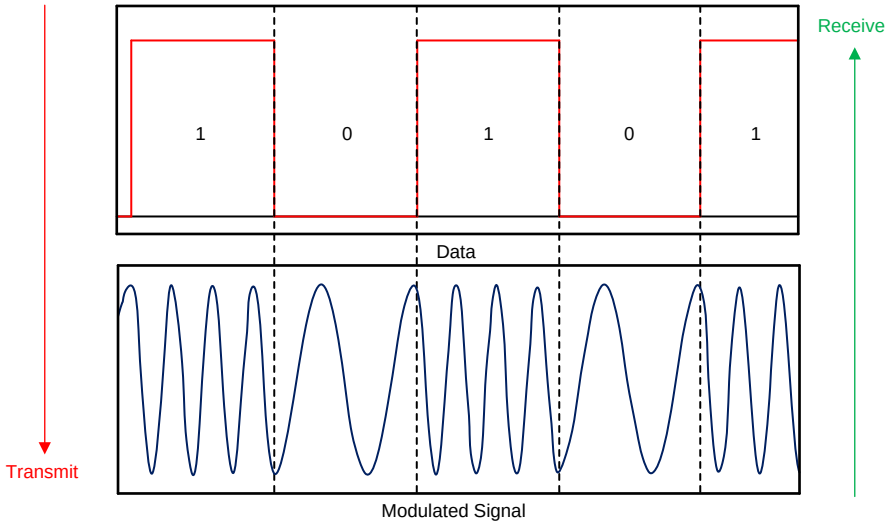



**Figure 6. BFSK in Time Domain**

Figure 7 shows an example of a simplified FSK transmitter where the block consists of two oscillators with an internal clock as well as an input binary sequence to control the position of the switch.

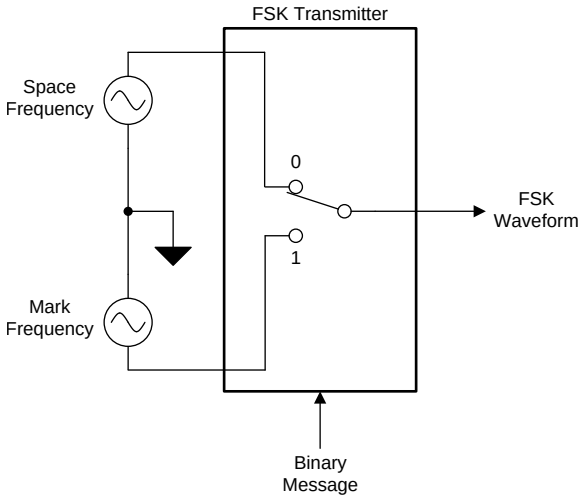



**Figure 7. Transmitter Example**

The two oscillators produce a higher (space) and a lower (mark) frequency signals. These oscillators are connected to a switch along with an internal clock. A clock is applied internally to both oscillators to avoid phase discontinuities of the output waveform during the transmission of the message. The binary input sequence is applied to choose the frequencies according to the binary input. In this case, binary "0" corresponds to the output of the space frequency and binary "1" corresponds to the output of the mark frequency.

Figure 8 shows an example of a simplified FSK receiver to convert a received signal back into the desired digital information.

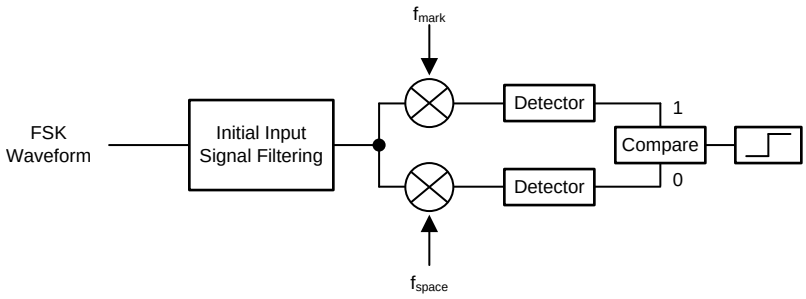



**Figure 8. Receiver Example**

An FSK waveform is initially filtered and then mixed with signals of the desired mark ($f_{mark}$) and space ($f_{space}$) frequencies. The output is run through a detector algorithm and the results are compared to decipher if the signal being received pertains to a mark, binary "1", or space, binary "0". Additional functionality is included to decipher received bits, based on the duration of the received mark or space signal, and handle the boundaries between consecutive bits.

This is a simple overview on how FSK works. The following sections discuss how this is implemented on a C2000 device.

# 3 Hardware, Software, Testing Requirements, and Test Results

## 3.1 Required Hardware and Software

### 3.1.1 Hardware

The system created for the SunSpec FSK transmitter or receiver is a combination of the following boards:

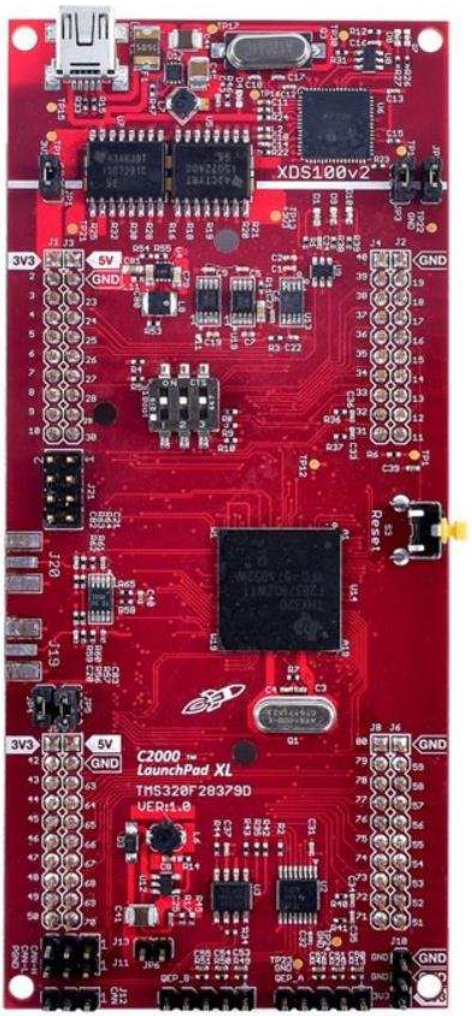



**Figure 9. TMDS28379D LaunchPad**

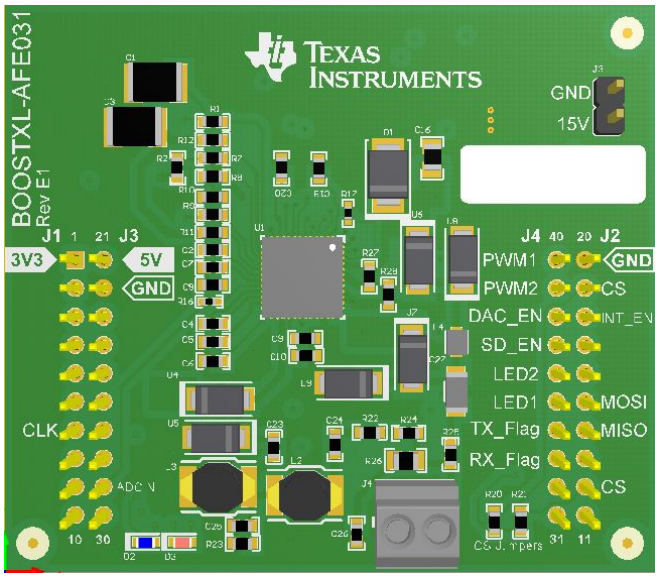



**Figure 10. BoosterPack Test Board**

Find the BoosterPack test board design files in C2000Ware.

### 3.1.1.1 Hardware Setup

There are a few things that must be done so that the hardware can be debugged correctly. Power is supplied to the BOOSTXL-AFE031 test board through the 15-V jumper. The BoosterPack has a regulator that supplies power to the LaunchPad connected. Figure 11 shows a close up of the 15-V headers.

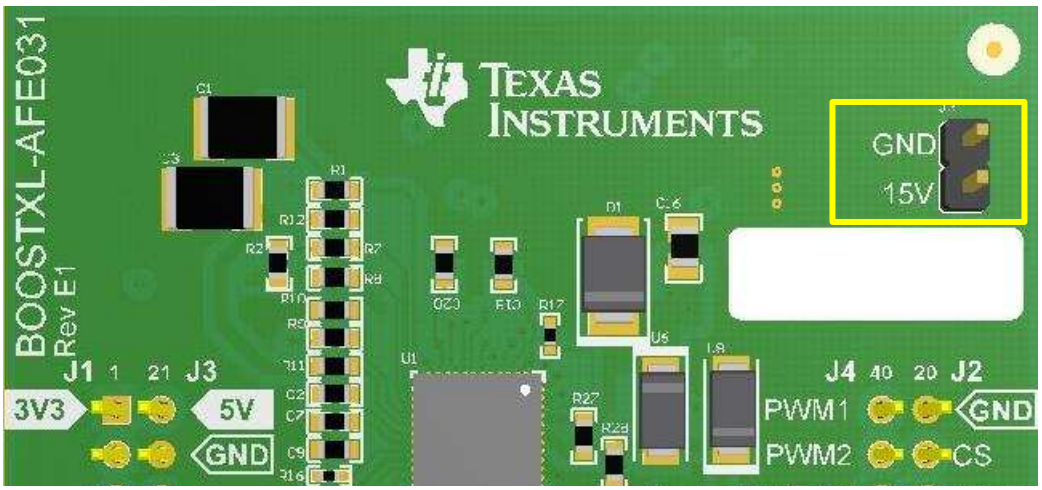



**Figure 11. BOOSTXL-AFE031 Power Pins**

The wire connected to the right input of the terminal block shown in Figure 12 is the line that the transmitted output or received input FSK signal resides on, depending on whether the system is set up as a transmitter or receiver. The wire connected to the left input of the terminal block is the ground line. These lines can then be connected to some coupling circuitry or directly to another BOOSTXL-AFE031 terminal block for controlled testing, TX/RX to TX/RX and GND to GND.

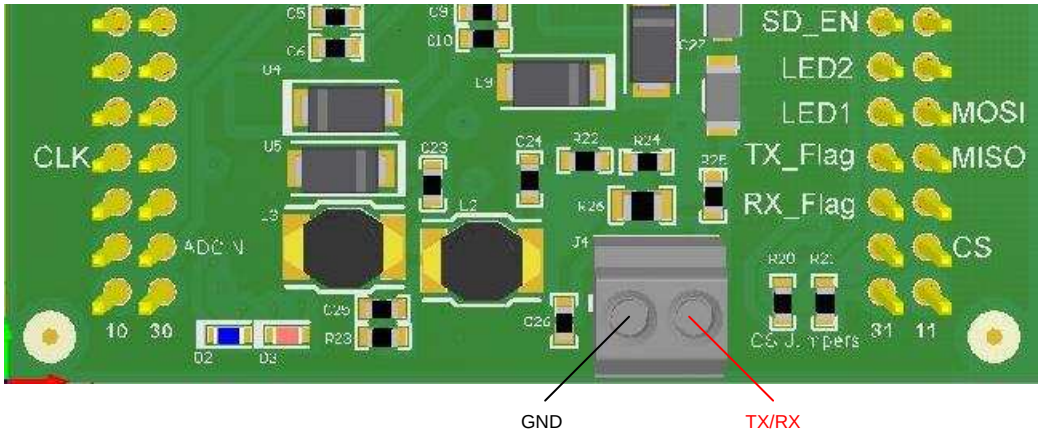



**Figure 12. BOOSTXL-AFE031 Terminal Block Connections**

Because the BoosterPack is supplying power to the LaunchPad, the PC USB interface must be isolated. Header jumpers JP1, JP2, and JP3 on the LaunchPad must be removed to enable electrical isolation. Figure 13 is a close up of the location of the headers.


**Figure 13. F28379D Jumper Configuration**

When using the system as a FSK receiver, the LaunchPad needs 5 V to power the C2000 ADC. With the LaunchPad being isolated from the PC USB interface, the 5-V power rail must be generated by stepping up the supplied 3.3 V. Header JP6 on the LaunchPad must be added to enable the step-up regulator. Figure 14 is a close up of the location of the header.

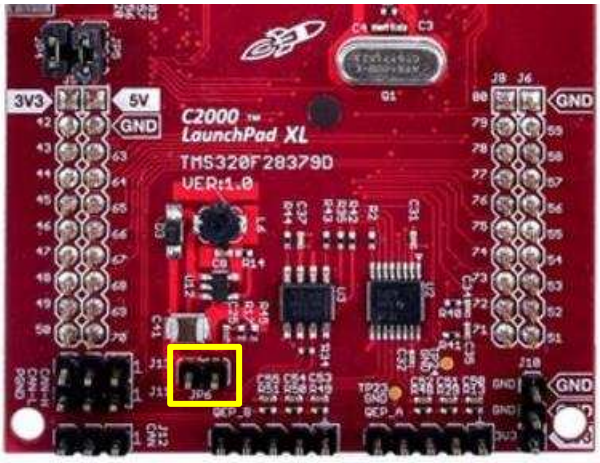



**Figure 14. F28379D JP6 Location**

*When using the system as a receiver, make sure that the F28379D LaunchPad is version 2.0 or greater.* Earlier versions (Ver 1.1 and 1.2) have an issue with the ADCIN pin that is connected to the BOOSTXL-AFE031. For more information, see the revision history of the *LAUNCHXL-F28379D Overview User's Guide* .

When using the transmitter solution in conjunction with the receiver solution, the complete system connection should look similar to what is shown in Figure 15.

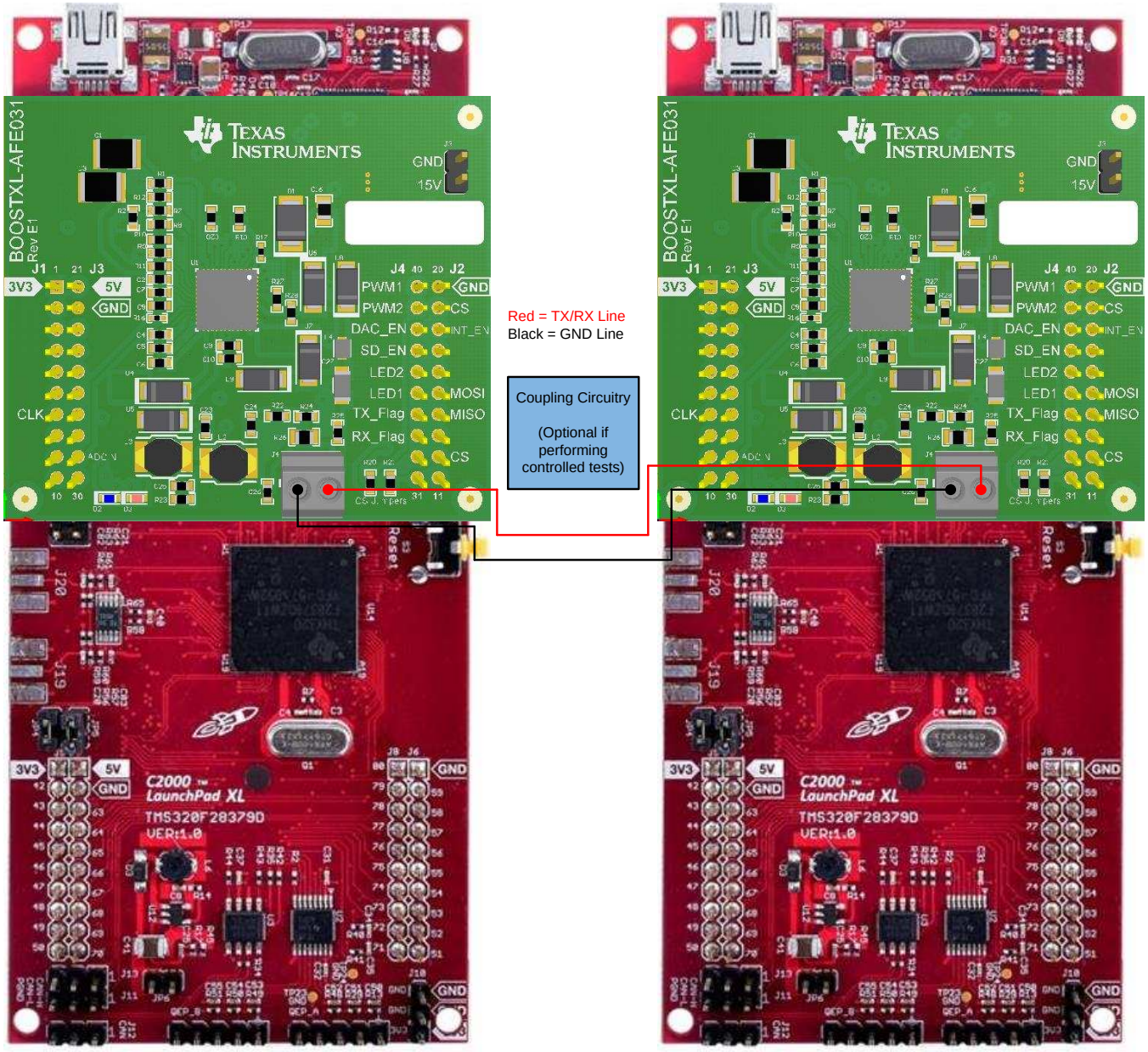



**Figure 15. TX-RX Solution System**

### 3.1.1.2 Interfacing With a Power Line

PLC is used with Texas Instruments' C2000 paired with the AFE031 device, which enables data to be sent over existing power cables. This pairing means that one can both power and control or retrieve data at the same time with just power cables running though these devices. This setup minimizes the overall cost that would be needed otherwise to create a communications path with extra cabling.

#### 3.1.1.2.1 Line Coupling

Line coupling is one of the most crucial segments of the PLC system, having two primary functions. The first function is to couple the signal from the AFE031 to and from the AC mains and DC bus. The second is to prevent the low frequency and high voltage of 50 or 60 Hz from the mains from damaging the PLC circuitry.

#### 3.1.1.2.2 Coupling to an AC Line

For coupling to the AC main, the following components are needed: a low-voltage capacitor, transformer, high-voltage capacitor, and an inductor. This section does not show the necessary protection circuitry; that information is discussed in Section 3.1.1.2.4. Figure 16 shows a simplified diagram.

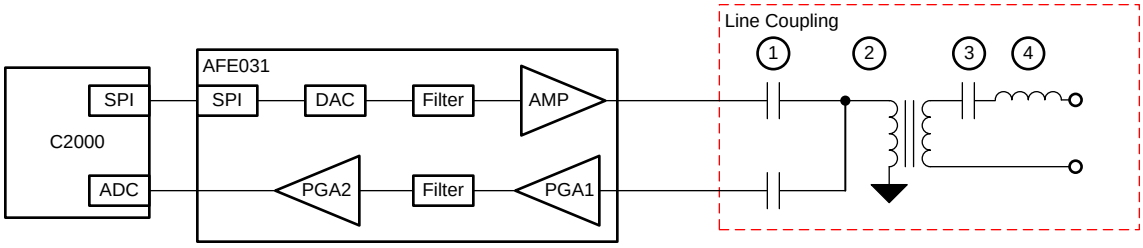



**Figure 16. Simplified Line Coupling Circuit for AC Main**

This section addresses the components from the left side (that connects to the TX and RX of the AFE031 device) to the right side (that connects to the power line). Consider the following key points during hardware design.

##### 3.1.1.2.2.1 LV Capacitor

The low-voltage (LV) capacitor couples the time-varying components of the PA output signal into the line coupling transformer. The LV capacitor should have a large enough capacitance to appear as a low-impedance throughout the signal band of interest. A value of 10 µF is a common value for signals ranging from 35 kHz to 150 kHz. The BoosterPack LV capacitor value is 10 µF because the space and mark frequencies (131.25 kHz and 141.75 kHz) lie within the range. The voltage rating of the LV capacitor must be sufficient to withstand the clamping voltage of the TVS diode (discussed in Section 3.1.1.2.4) operating under surge conditions. Generally, this limit is equal or slightly higher to the PA supply voltage.

##### 3.1.1.2.2.2 Ratio of Transformer

Most PLC transformers are compact, with turns ratios between 1:1 and 4:1, low leakage inductance, and approximately 1 mH of winding inductance. This inductance in series with the high-voltage capacitor results in a voltage divider attenuating the ac mains voltage down to negligible levels at the module output.

To determine the optimal turns ratio for the transformer, it must be based on the PA's capabilities of maximum output swing and maximum output current to achieve maximum power transfer into the load.

$$\frac{N_1}{N_2} = \sqrt{\frac{PA, V_{OUT\_Peak}}{PA, I_{OUT\_Peak} \times R_{Load}}} \quad (1)$$

There are three cases where a one limitation is dominant than the others.

- Case 1: If the turns ratio of the transformer is greater than the ideal calculated value, the TX output of the AFE031 is limited by the voltage swing of the PA.
- Case 2: If the turns ratio of the transformer is less than the ideal calculated value, the TX output of the AFE031 is limited by the maximum output current from the PA.
- Case 3: If the turns ratio of the transformer is equal to the ideal calculated value, the TX maximum output occurs as the amplifier approaches both its maximum output voltage and maximum output current, resulting in maximum power transfer to the load.

The transformer affects the coupling emission performance of the EN50065-1 when under a 2-MHz frequency. To compensate, TI recommends using products from Würth Electronik.

#### 3.1.1.2.2.3 HV Capacitor

The high-voltage (HV) capacitor blocks the low-frequency mains voltage by forming a voltage divider with the winding inductance of the line-coupling transformer. While using high-voltage CBB capacitors, the maximum voltage range must exceed the amplitude of the power grid AC voltage. Operating the capacitor at approximately 80% of its AC-rated voltage ensures a long component operating life. The next important requirement is a standard setting maximum reactive power (VA limit). For example, the European product standards for attaching a device to the power grid must have a reactive power of less than 10 VAR, resulting in a capacitor values less than 0.55 µF. Equation 2 shows how the value 0.55 µF is used to determine the HV capacitor value.

$$HVCap = \frac{VA_{LIMIT}}{VAC^2 \left(2\pi \times f\right)} \tag{2}$$

For a 240-V AC, 50-Hz application with a 10-VA limit,

$$HVCap \leq \frac{10}{240^2 \left(2\pi \times 50\right)} \cong 550\,nF \tag{3}$$

A metallized polypropylene electromagnetic interference and radio frequency interference (EMI/RFI) suppression capacitor is recommended because of the low loss factor associated with the dielectric, which results in minimal internal self-heating.

#### 3.1.1.2.2.4 HV Side Inductor

The inductor connected in series with the HV capacitor is required when driving low line impedances, and the HV capacitor is restricted to approximately 470 nF. In applications that operate in the CENELEC A band, the impedance of the 470-nF capacitance at 40 kHz is approximately 8.5 Ω. If the application requires the ability to drive a 2-Ω load, for example, this series impedance is restrictive. Adding the series inductor can mitigate this effect. To properly select the value of the inductance, the operating frequency range of the system must be known. A common example would be the PRIME frequency band, which is approximately 40 kHz to 90 kHz. Selecting the HV capacitor and inductor to have a resonant frequency in the center of the frequency band is recommended, and results in a series inductor value of 12.8 µH and HV capacitor value of 470 nF.

$$L = \frac{1}{HVCap \times \left(2\pi \times f\right)^2} \tag{4}$$

The inductor must be sized to be capable of withstanding the maximum load current without saturation.

### 3.1.1.2.3 Coupling to DC Line

Coupling to a DC line has similar components for protection with the lack of components, such as an HV inductor, metal-oxide varistor (MOV), and transformer. Figure 17 shows what is needed for a coupling circuit and transient protection.

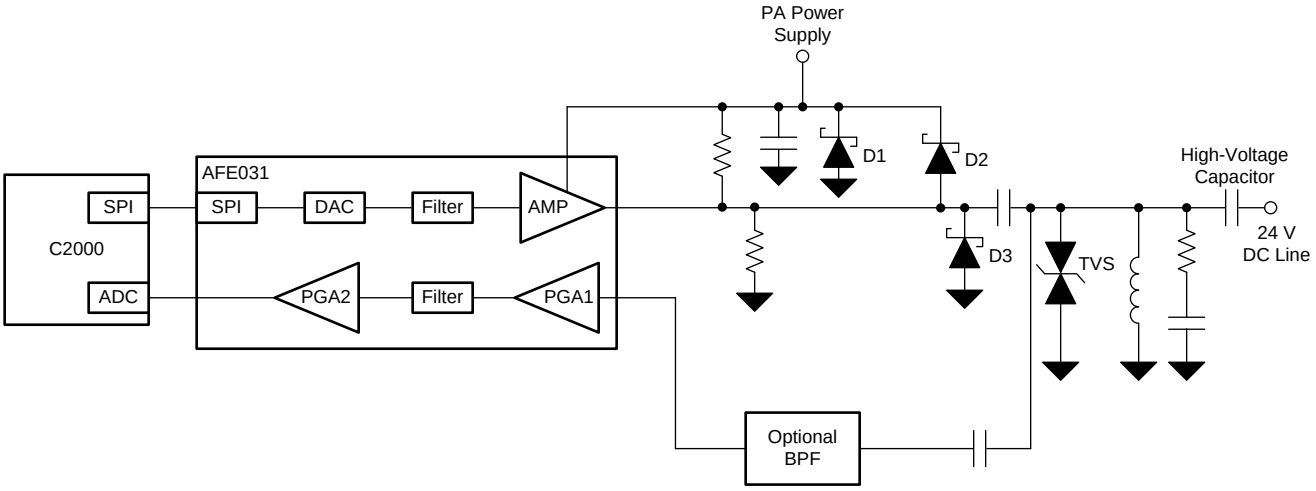



**Figure 17. Coupling Circuit for a DC Line**

A 10-µF capacitor is used to couple to the DC line. When coupling to a DC line, remember that the line is generally low impedance and can affect the output swing of the PA in the AFE031. Connecting an inductor in series with the line provides enough impedance to the PLC signal such that the power supply (possibly very low impedance) does not interfere with the PLC signal modulation. Figure 18 shows PLC modules coupled to a low-impedance DC line.

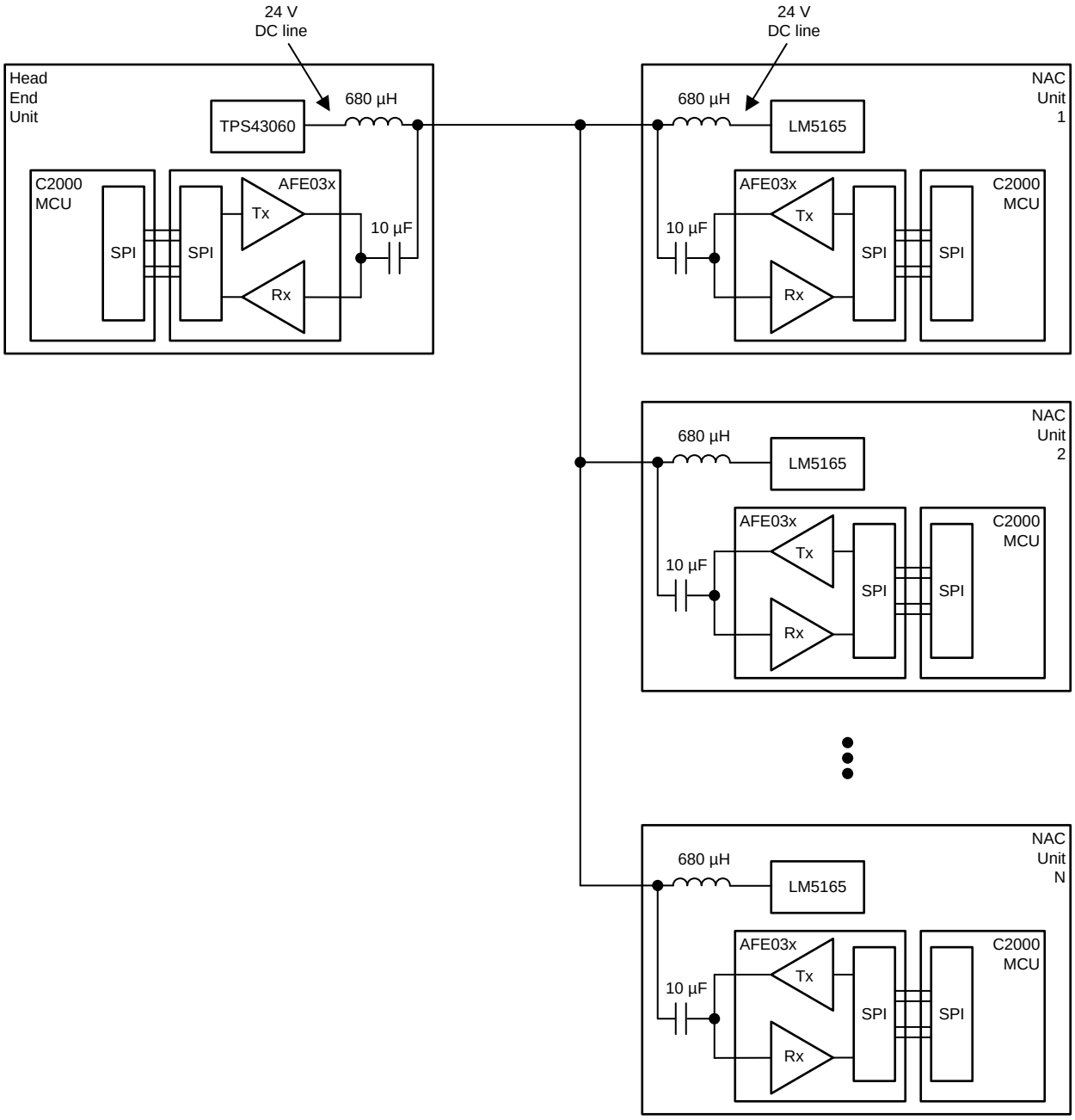



**Figure 18. Example of Multiple PLC Modules Coupled to a DC Line**

For detailed test data on this circuit, see the *DC Power-Line Communication Reference Design* .

#### *3.1.1.2.4 Protection Circuit*

PLC are often located in operating environments that are harsh for electrical components connected to the AC line. Noise or surges from electrical anomalies such as lightning, capacitor bank switching, inductive switching, or other grid fault conditions can damage high-performance integrated circuits if they are not properly protected. The AFE031 can survive even the harshest conditions if several recommendations are followed: MOVs, transient voltage suppression diodes (TVSs), Schottky diodes, and a Zener diode.

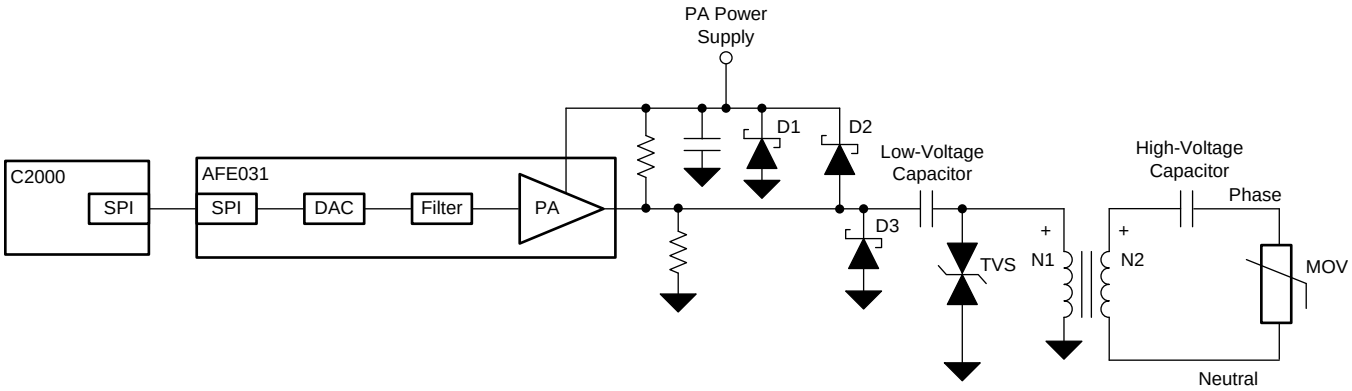



**Figure 19. Recommended Transient Protection**

#### *3.1.1.2.4.1 Metal Oxide Varistors*

There are several factors to consider when selecting an MOV:

- Working voltage
- Required amount of transient energy to be absorbed by the MOV
- Peak transient current
- Power dissipation

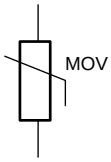



**Figure 20. Metal Oxide Varistor (MOV)**

An MOV is a device that has high resistance until its triggering voltage is exceeded. Once this voltage level has been exceeded, the MOV reduces its resistance and absorbs the energy from the pulse. Figure 21 shows the I/V characteristic of a typical MOV.

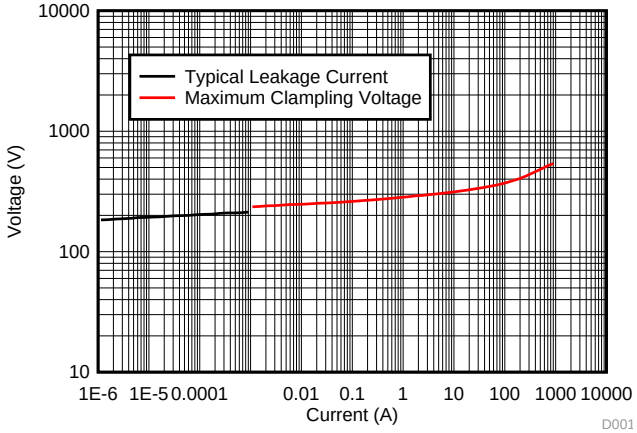



**Figure 21. Typical MOV I/V Characteristic**

By the nature of the materials and techniques used in the construction of these components, MOVs respond quickly to a fast transient pulse, have high instantaneous power ratings, and are well-suited for protection on the AC line. The maximum clamping voltages are typically specified in response to a high-speed transient similar to that shown in Figure 22. The 8/20-µs waveform is commonly associated as a waveform that represents the spectral content of lightning strikes.

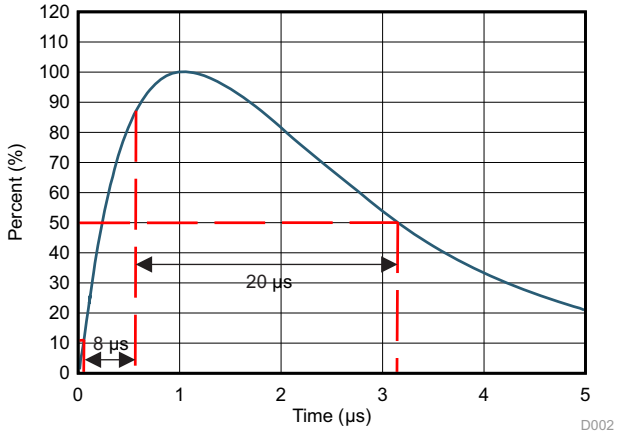



**Figure 22. Typical 8/20-µs Pulse for MOV and TVS Performance Specification**

### *3.1.1.2.4.2 Transient Voltage Suppressors*

A TVS is a very fast-acting clamping device that turns on in the case of an overvoltage condition, shunting the surge of current into ground. TVSs are rated primarily by the power handling capability and the clamping voltage. TVSs are available in either unidirectional or bidirectional configurations.

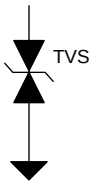



**Figure 23. Bidirectional TVS Diode**

For PLC applications, a bidirectional TVS is recommended; the component is placed next to the line coupling transformer. Figure 24 shows the I/V characteristics for a typical bidirectional TVS.

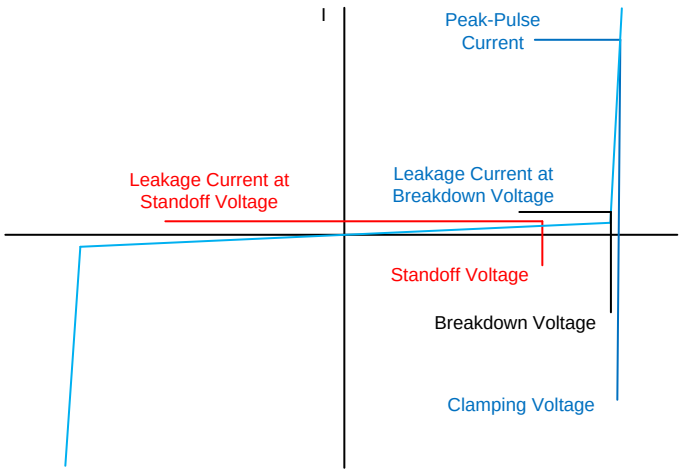



**Figure 24. Typical Bidirectional TVS I/V Characteristics**

As a surge or pulse on the AC line occurs, the voltage rises across the TVS. If the voltage rises higher than the TVS breakdown voltage, the TVS turns on and rapidly changes from high impedance to low impedance, shunting current into ground. The low-voltage capacitor between the PA output and the TVS blocks any DC voltage at the TVS. As a result, the normal FSK or OFDM signal from the PA appears to be centered around ground at the TVS. This condition requires the TVS to be bidirectional.

Figure 25 illustrates this concept.

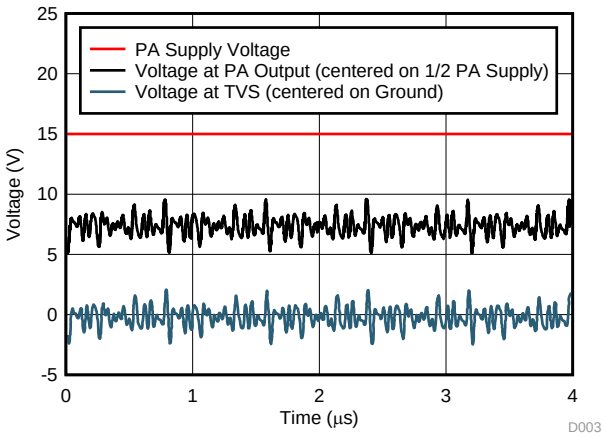



**Figure 25. Typical Signal at TVS**

Because the signal is symmetric around ground at the TVS, the TVS breakdown voltage must be equal to approximately one-half of the PA power-supply voltage. It is important for the TVS to remain off during normal operation to avoid clipping and introducing distortions to the output signal. It is also important that the TVS turn on and clamp at the lowest possible voltage beyond normal operation to provide maximum protection.

This BoosterPack is designed with surge protection parts. A bidirectional TVS diode with a shunt connection to attach on the LV side of the transformer is used to clamp the voltage of a surge. The stable voltage of the TVS must be exactly half of the PA supply voltage of the AFE, meaning a 15-V AFE must use a 7.5-V TVS. This ½ ratio is based on the fact that:

- The PA of the AFE is a type AB, which uses a single power rail. Thus, the 12-V powered PA TX output is at a 6-V bias with a ±6-V amplitude, meaning that the signal has a ±6-V range. While on transmission, the TVS must not be allowed to saturate the signal, so the stable voltage must be kept to ≥ ½ of the PA power rail.
- If a 12-V PA uses a 7.5-V TVS, during the arrival of a surge pulse, the LV side bias is locked-on at 7.5 V, but if the surge occurs at the exact moment of a TX maximum amplitude (which is 6 V), then the signal on the TX route is 7.5 V + 6 V = 13.5 V, which is higher than the power rail and causes damage to the PA of the AFE. So, the TVS voltage must be ≤ ½ of the PA power rail.

#### *3.1.1.2.4.3 Current Steering Diodes*

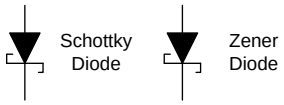



**Figure 26. Current Steering Diodes**

While the MOV and TVS components clamp the pulse and either dissipate or re-direct most of the energy to ground, TI also recommends placing current-steering (Schottky) diodes at the output of the PA section of the AFE031. In the unlikely event a transient surge increases the PA output pin beyond its power-supply rail, low-drop Schottky diodes can steer the current around the AFE031 safely to ground. Maintaining a low (less than 0.8 V) forward voltage drop on the Schottky diode is recommended for

maximum protection. If the Schottky diode that connects the output of the PA to the power-supply rail turns on and becomes forward-biased, it is important to steer the current to ground without significantly disturbing the PA power-supply voltage. Placing a Zener diode at the PA power-supply pins to ground provides a low-impedance path for surges that attempt to raise the power-supply voltage beyond the absolute maximum rated voltage for the AFE031.

#### 3.1.1.2.5 Determining PA Power Supply Requirements

Calculating the minimum power-supply requirements for the PA, the desired load voltage, load impedance, and available power-supply voltage or desired transformer ratio are all the parameters that must be known. For this FSK PLC example, similar to PRIME, the goal is to drive a 1-$V_{RMS}$ signal into a 2-Ω load. The minimum power-supply voltage required is calculated by adding the peak-to-peak load voltage; the voltage dropped across the HV capacitor and inductor, V2; the voltage dropped across the LV Cap, V1; and twice the output swing to rail limit of the PA, VSWING. For FSK and SFSK systems, the peak-to-average ratio is √2, while this ratio is approximately 3:1 for OFDM systems.

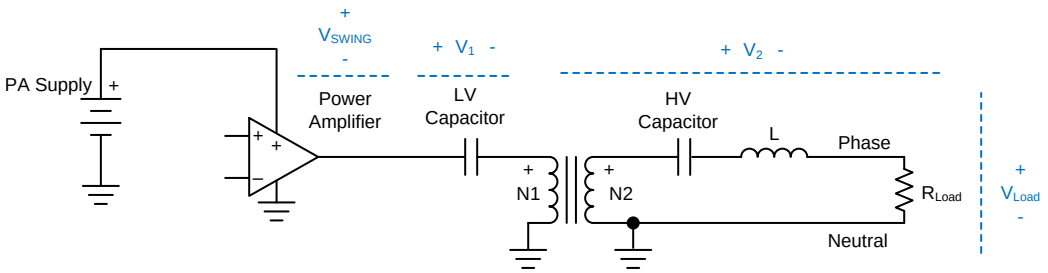



**Figure 27. Typical Line Coupling Circuit**

Consider these ratios when performing calculations that relate the RMS voltages and peak voltages during an analysis. Choosing a large value for the LV capacitor results in the voltage drop (V1) becoming negligible in most circumstances. The losses in the transformer are also negligible, even at high load currents, if the proper transformer with a low DCR is used. For FSK and SFSK systems, the voltage drop across the HV capacitor and inductor, V2, is also usually negligible.

For the AFE031 BoosterPack, an FSK signal with a 2-Ω load and 1-$V_{RMS}$ load voltage:

$$PA_{Supply} = V_{Load} \times \text{Turns Ratio} \times \left(2 \times V_{Swing}\right) \quad (5)$$

$$PA_{Supply} = 2.878\ \text{V} \times 1.5 + \left(2 \times 2\ \text{V}\right) = 8.25\ \text{V} \quad (6)$$

For more information on power dissipation of the AFE031, see *Analog Front-End Design for a Narrowband Power-Line Communications Modem Using the AFE031* .

### 3.1.2 Software

The target processor for the software is the TMDS320F28379D, but the software can be ported to other devices. For the project collateral and example code discussed in this guide, refer to the latest C2000Ware release within the directory:

C:\ti\c2000\C2000Ware_x_xx_xx_xx\device_support\f2837xd\examples\cpu1

The available example projects are:

- boostxl_afe031_f28379d_pwmmode
- boostxl_afe031_f28379d_dacmode
- boostxl_afe031_f28379d_rx

#### 3.1.2.1 Interfacing With AFE03x for Transmitting SunSpec® Protocol

When interfacing with the AFE03x as a transmitter, first decide which mode of operation to use for the AFE03x: DAC mode or PWM mode.

In DAC mode, the C2000 MCU sends SPI data to update an internal DAC on the AFE031; the DAC value output is filtered and amplified. In this mode, a sine table is cycled through, and each point is sent to the AFE031 over SPI.

In PWM mode, the SPI connection is only used to configure the registers for the AFE. Two PWM signals are generated and supplied to the AFE031 at the desired frequency for FSK transmission. The AFE adds these signals together to create a waveform that creates less harmonics than a single PWM signal.

These two modes require different hardware configurations, so be sure to choose one or the other. If using the BOOSTXL-AFE031 test board as a reference, the hardware has been designed so that both modes can be used by simply adding or removing components.

When interfacing with the AFE03x as a receiver, the SPI connection is only used to set the registers of the AFE, much like the PWM transmit mode. The receive path of the AFE is enabled to allow the input signal to reach the ADC input of the C2000 MCU. Along this path lies some filters and two programmable-gain amplifiers (PGA) to step up or step down the input signal before entering the C2000 ADC.

#### *3.1.2.1.1 Configuring the AFE031*

There are two main steps to configure the AFE device. First, ensure that the GPIOs are configured correctly. On the AFE device, two main GPIOs are set up: DAC Mode Select and System Shutdown. Both of these pins are pulled low. All other GPIOs connected can also be pulled low.

The next step is to configure the device over SPI. In the software example, the *HAL_afe031Init()* function configures the AFE031 and is defined within AFE03x_Config.c.

The following steps show the sequence the function follows to correctly configure the device:

1. Configure GPIOs.
   a. Set up SD and DAC pins on the AFE device. Both of these pins must be brought low.
   - Software example function: *HAL_afe031_cfgGpio();*
2. Configure SPI.
   a. Configure SPI module for 16-bit characters. For more information regarding SPI requirements, see *AFE031 Powerline Communications Analog Front-End* .
   - Software example function: *HAL_spi_cfg();*
3. Perform a soft reset on the AFE device:
   a. Write 0x14 to the reset register.
   - Software example function: *HAL_afe031_softReset();*
4. Enable the bias:
   a. Write 0x03 to the enable2 register.
   - Software example function: HAL_afe031_biasEnable();
5. Select a frequency band:
   a. Write either 1 or 0 to the CA_CBCD bit inside the control1 register. If a 1 is written, the frequency response of the TX and RX filters will be configured to CENELEC B, C, and D. If a 0 is written, then CENELEC A will be configured.
   - Software example function: *HAL_afe031_bandSelect(1);*
6. Clear all interrupts:
   a. Write 0x00 to the control2 register.
   b. Software example function: *HAL_afe031_clrAllInt();*
7. Configure all interrupts:
   - Interrupts can be configured by writing to the control2 register. The software example function currently only enables the T_flag that indicates thermal overload.
   - Software example function: *HAL_afe031_cfgInt();*
8. Enable ZC:
   a. Write a 1 to the ZC bit in the enable2 register.
   b. Software example function: *HAL_afe031_zcEnable();*

9. Write the TX gain if transmitting:
   a. Write either a 0, 1, 2, or 3 to the TXG bits in the gain select register. The gains are as follows:
      - 0 = 0.25 V/V
      - 1 = 0.5 V/V
      - 2 = 0.707 V/V
      - 3 = 1 V/V
   b. Software example function: *HAL_afe031_writeTxGain(UINT16 gain);*
      - Note that the software example function uses an array to write the 0 to 3. Thus when using the function, the input parameter corresponds to the index of the desired *HAL_afe031_txGainLut* array element. For example, passing 0 to the function corresponds to a gain of 0.25 V/V.
10. Write the RX gain if receiving:
   a. Write 0 to 15 to the RXG bits in the gain select register. The gains range from 0.25 V/V when RXG is set to 0x0, to 128 V/V when RXG is set to 0xF.
      - For specific gains, see Table 4.
   b. Software example function: *HAL_afe031_writeRxGain(UINT16 gain);*
      - Note that the software example function uses an array to write 0 to 15. Thus, when using the function, the input parameter corresponds to the index of the desired *HAL_afe031_rxGainLut* array element. The included array does not have values for all RX gain configurations.

When these steps are completed, begin the final few configurations needed based on the transmit or receive implementation being used.

## 3.2 Transmit Path

When the AFE is initialized correctly, enable the system to transmit based on which transmit method is chosen: PWM mode or DAC mode.

### 3.2.1 PWM Mode

In PWM mode, the C2000 F28379D generates two symmetric PWM signals that go directly into the AFE device. The two symmetric PWM signals are 66% and 33% duty cycle. These signals are added together inside the AFE device and create a waveform that has the least amount of noise. Figure 28 shows how this addition works.

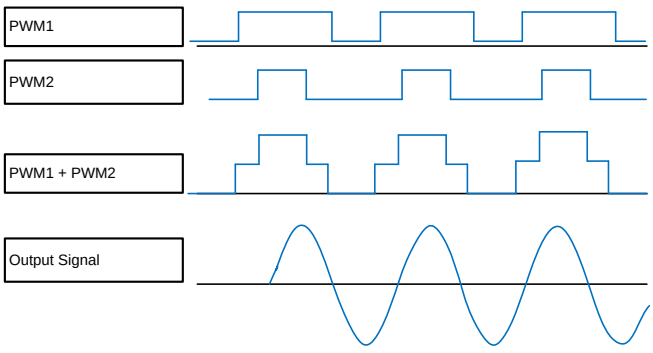



**Figure 28. PWM Addition**

Figure 29 shows the path of the PWM signals. The PWM signals go into the low-pass filter internal to the AFE030/AFE031 device, and are added together to create the above PWM1+PWM2 waveform. Figure 30 shows the gains witnessed at certain frequencies at the output of the internal TX low-pass filter.

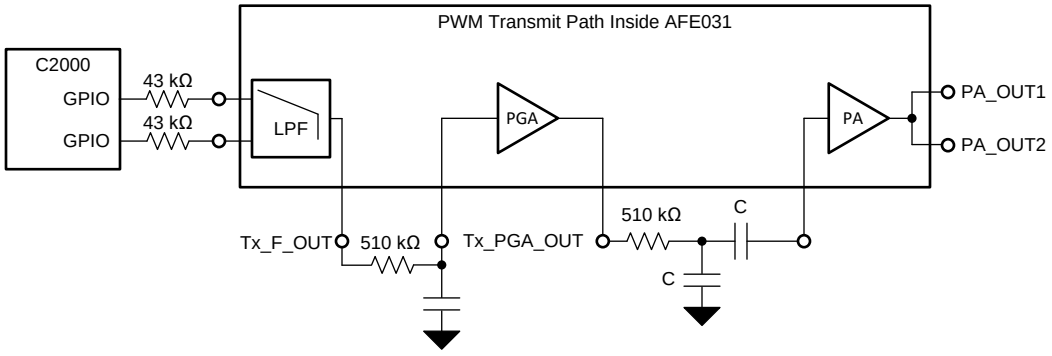



**Figure 29. PWM Transmit Path**

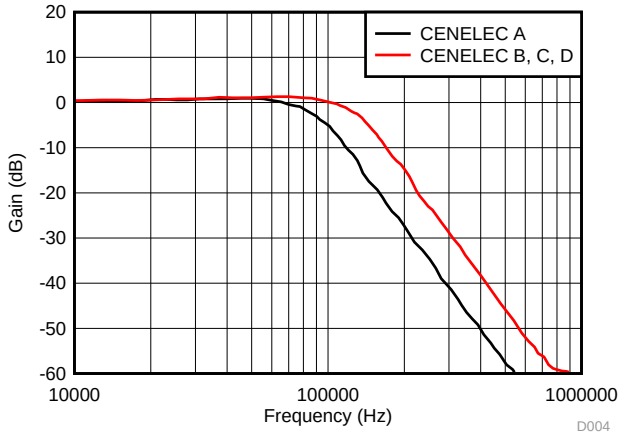



**Figure 30. TX Filter Gain vs Frequency**

Next, the signal goes through a low-pass filter, PGA, another low-pass filter, and finally out of the PA. The external low-pass filters can be tuned to filter the desired frequencies, as shown in Table 2.

**Table 2. External R and C Values to Increase Filter Response in PWM Applications**

| FREQUENCY BAND | R (Ω) | C (nF) |
|---|---|---|
| SFSK: 63 kHz, 74 kHz | 510 | 2.7 |
| CENELEC A | 510 | 1.5 |
| CENELEC B, C, D | 510 | 1 |

### 3.2.2 HRPWM vs EPWM

Unique to the C2000 device is the ability to use high-resolution PWM (HRPWM). HRPWM enables increased resolution for both the duty cycle and period of the PWM signals. In this example, HRPWM is used to generate both the mark and the space frequencies.

The HRPWM is based on micro edge positioner (MEP) technology. MEP logic is capable of positioning an edge very finely by sub-dividing one coarse system clock of a conventional PWM generator. The time step accuracy is on the order of 150 ps.

Table 3 shows the resolutions possible with and without HRPWM.

**Table 3. Resolution for PWM and HRPWM**

| PWM FREQUENCY (kHz) | REGULAR RESOLUTION (PWM) | | HIGH RESOLUTION (HRPWM) | |
|---|---|---|---|---|
| | 100-MHz EPWM CLK | | | |
| | BITS | % | BITS | % |
| 20 | 12.3 | 0.02 | 18.1 | 0.000 |
| 50 | 11 | 0.05 | 16.8 | 0.001 |
| 100 | 10 | 0.1 | 15.8 | 0.002 |
| 150 | 9.4 | 0.15 | 15.2 | 0.003 |
| 200 | 9 | 0.2 | 14.8 | 0.004 |
| 250 | 8.6 | 0.25 | 14.4 | 0.005 |
| 500 | 7.6 | 0.5 | 13.4 | 0.009 |
| 1000 | 6.6 | 1 | 12.4 | 0.018 |
| 1500 | 6.1 | 1.5 | 11.9 | 0.027 |
| 2000 | 5.6 | 2 | 11.4 | 0.036 |

For example, the mark frequency is generated at a 131.25-kHz signal. Using PWM, only a 131.2-kHz or a 131.3-kHz signal can be generated. This limitation is due to the resolution that is available for the PWM module. If HRPWM is added, a 131.250-kHz signal can be effectively generated.

If this amount of accuracy is not necessary or the desired frequency can be reached with a normal PWM, HRPWM is not required. HRPWM is an add-on to PWM. To disable this add-on in the software example, remove the code associated with it inside the main.c file.

### 3.2.3 PWM Mode Software Implementation

The example program referenced is: boostxl_afe031_f28379d_pwmmode.

To enable PWM mode in software, complete the following flow:

1. Enable PWM transmit mode:
   - To enable the PWM transmit mode, set the TX and PA bits in the enable register to 1, and set the DAC bit to 0.
   - Software example function: *HAL_afe031_txPWMEnable();*
2. Disable the DAC transmit mode.

Two PWM sources create the two PWM signals: one PWM source is used to set the frequency of the two outputted signals and the other controls the bit rate for the sent data. In the software example, PWM2 is used to control the bit rate and generate an interrupt to determine the frequency that must be outputted.

In the software example, the PWM2 interrupt handles all of the FSK protocol requirements. The implemented protocol is a repeatable pattern, which allows the software to be based on a cycle count. One cycle count is the time period for one bit. In this implementation, 33 bits (11 bits per word, three words) are sent. During each cycle, it checks the value of the next bit and the PWM frequency changes to either the mark or space frequencies. After 33 cycles, the system stops sending PWM signals and enters the quiet mode. After 209 cycles, the cycle count is reset and the software starts sending the packet again. With the FSK transmission being handled by the PWM2 interrupt, the CPU's main function is free to be used for other applications. By default, the software example transmits a packet_1 referenced in Table 1, but this can be changed to a packet_0 by setting the packet_to_send variable to a zero.

### 3.3 Testing and Results

Figure 31 and Figure 32 show that the system created both the mark and space frequencies. Figure 33 shows the complete packet being sent out of the system.

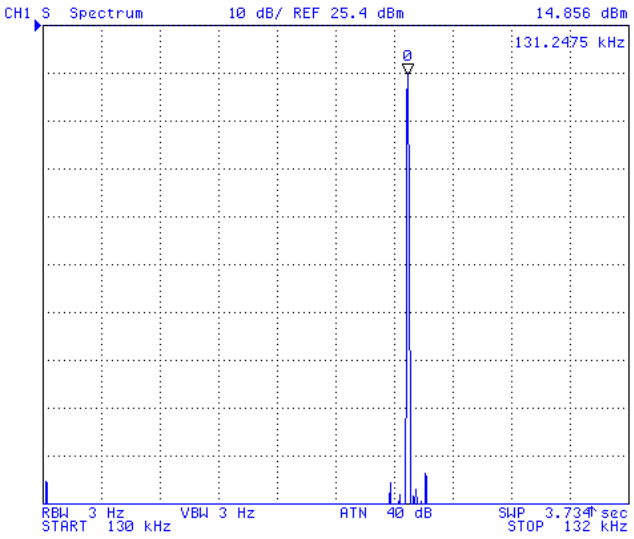



Figure 31. PWM Mode Mark Frequency, 131.25 kHz

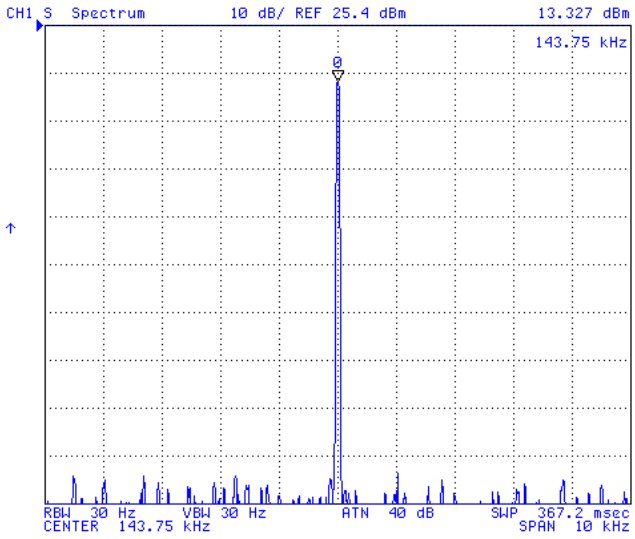



Figure 32. PWM Mode Space Frequency, 143.75 kHz

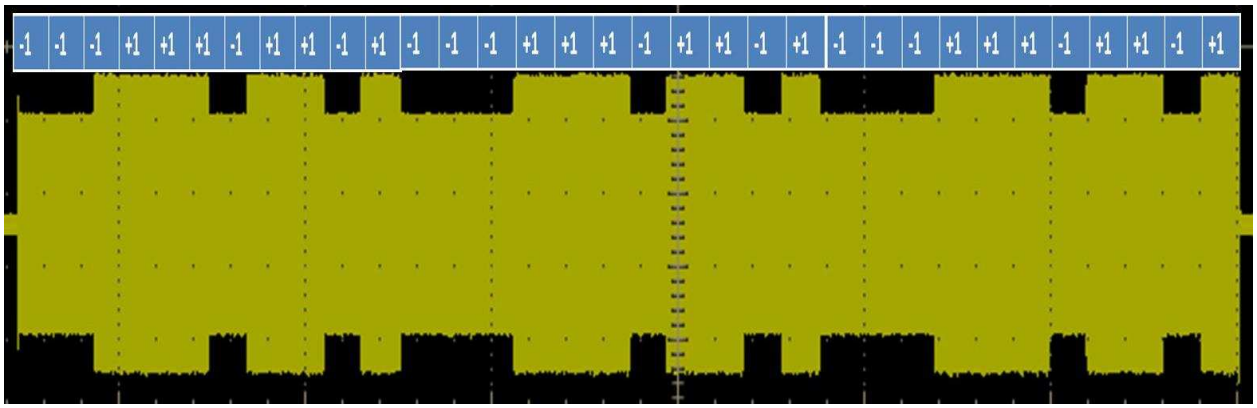


Figure 33. Full Packet Transmission Waveform

### 3.4 DAC Mode

This section describes one way to create a DAC mode FSK transmitter. DAC mode is very similar to the PWM mode, in that PWM interrupts are used to accomplish FSK transmission. Figure 34 shows the transmit path used when using DAC mode.

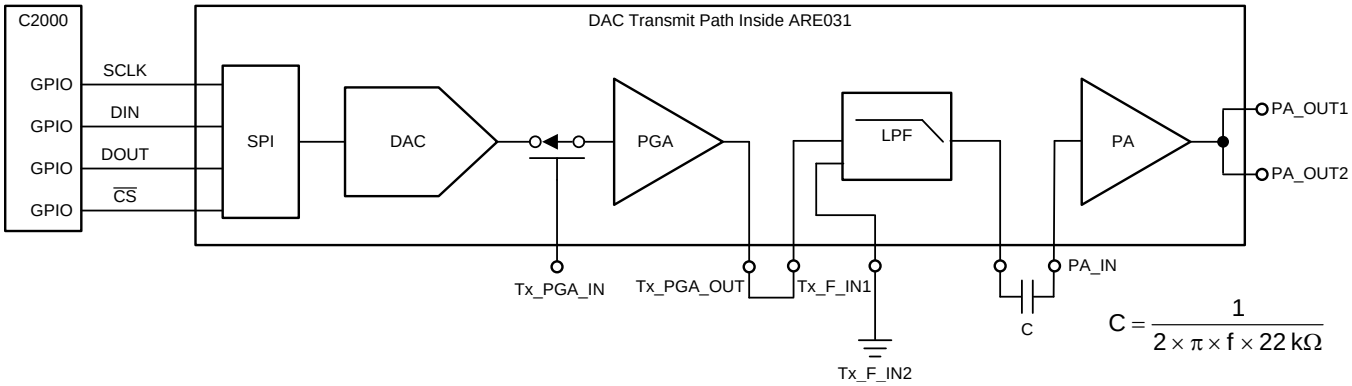



**Figure 34. DAC Transmit Path**

In DAC mode, the software sends data over SPI to the internal DAC to set the output value. To accomplish sending a single tone or sine wave, a ramp of DAC values at a given frequency are sent. Using a PWM interrupt occurring every period, send the updated DAC value at a frequency determined by the PWM period. The number of points in the sine table and desired frequency determine the frequency of the PWM signal. For example, if given a 100-kHz sine wave that has ten points, all ten points must be sent within 100 kHz. This means that the PWM must generate an interrupt that follows Equation 7:

PWM Frequency = Points of Sine Table × Desired Frequency (7)

Figure 35 shows what this looks like in the time domain.

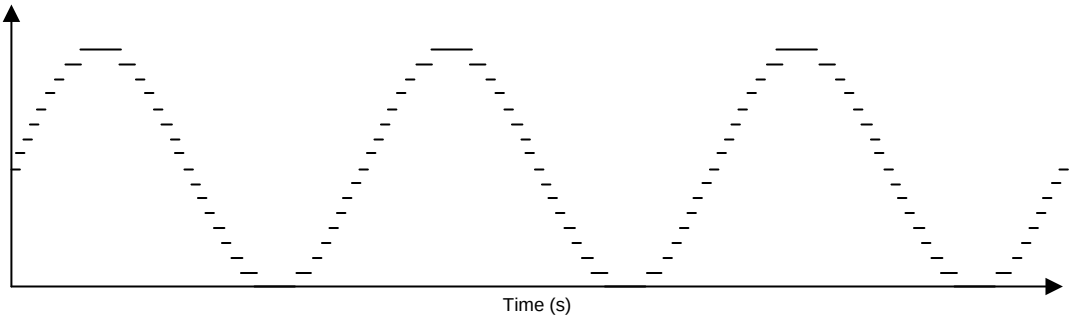



**Figure 35. DAC Sine Wave Ramp**

### 3.4.1 DAC Mode Software Implementation

The example program referenced is: boostxl_afe031_f28379d_dacmode.

To enable DAC mode in software, complete the following flow:

1. Enable DAC transmit mode internal to AFE:
   - To enable the PWM transmit mode, set the TX and PA bits in the enable register to 1, and set the DAC bit to 0.
   - Software example function: *HAL_afe031_txDACEnable();*
2. Enable the DAC transmit mode with GPIO toggle and Configure word length for SPI:
   - Set the GPIO connected the the DAC pin to 1.
   - Set the word length to 10 bytes for SPI communication.
   - Software example function: *HAL_afe031_dacEnable();*

Sending information in DAC mode can be accomplished similarly to how the PWM mode operates. One PWM source is used to set the DAC mode value to the correct value of the sine ramp that gets sent out. The second PWM is used for bit rate to generate an interrupt and determine what frequency must be outputted.

A problem can occur if attempting to reach vary precise frequencies with this implementation. The problem can be looked at with Equation 8.

$$\text{Number of steps in Sine Table} = \frac{\text{Frequency of Interrupt}}{\text{Frequency of Desired Signals}} \quad (8)$$

If trying to generate 131.25 kHz, then only two variables can change. One approach is to set the number of steps in the sine table. For example, if there are 10 steps in the sine table the frequency of interrupt is:

Frequency of Interrupt = (Number of Steps in Sine Table) × (Frequency of Desired Signal)
Frequency of Interrupt = 1.3125 MHz

With a 200-MHz clock, even if an interrupt occurs at either 152 or 153 CPU cycles, the interrupt frequencies would be 1.31579 MHz and 1.30719 MHz, respectively. These frequencies do not fall within spec for the frequency tolerances in Table 1. This result means setting the step size cannot be the correct implementation for precise frequency generation.

The other way to think of this is to set the frequency of interrupt. For example, set the interrupt for 1 MHz, which is possible to generate. The number of steps in the sine wave would then be:

Number of Steps in Sine Table = (1 MHz) / (131.25 kHz) = 7.61905

Using the floating-point capability of the F28379D, the processor can keep track of that remainder, and now the accuracy depends on the sine table, not the 1-MHz clock. The step size can be found using Equation 9:

Step Size = (Points in Sine Table)/ (Number of Steps in Sine Table) (9)

Continuing with the previous example, a 4096 sine table gives a step size of 537.6. This result means in every interrupt, the sine table steps another 537.6. Because the program is sorting through an array, this number is rounded off to 537. However, as this number gets added and the program shifts through the sine table, the next step will vary off the decimal step size. An example interrupt routine is as follows:

- //Transmit next data point in sine wave. Also convert float SinePosition to unsigned int:
  - Uint16 temp = sinePosition;
  - HAL_spi_xmt((sineTable[temp]));
- //Calculate next step:
  - sinePosition += sineStep; }
- //Check for overflow:
  - if(sinePosition > 4095)
  - { sinePosition -= 4095; }

In this implementation, a 1-MHz Interrupt is created, which only moves data from one memory address to another. A way to make this less CPU intensive is to use the C2000's direct memory access (DMA). The DMA peripheral moves data from one memory address to another based on a trigger event. Pre-fill two buffers with the correct data that is chosen to send out over SPI, and use the DMA to switch between the two. When switching from one buffer to the other, the older buffer refills with the values.

In the DAC mode software, a 1-MHz PWM signal is used to generate a DMA event that moves data from one of these buffers to the SPI TX buffer. Because the older buffer must be refilled after each use, the DMA triggers an event each time it finishes reading an entire buffer. During this interrupt, the buffers are switched, and the old one is refilled. By following this implementation, CPU utilization versus memory trade-offs can be weighed and the buffers can be sized accordingly. If extra memory is available, using a bigger buffer reduces the CPU overhead. If not a lot of memory is available, then the smaller buffer size increases the CPU overhead.

Exactly like the PWM mode software implementation, the PWM2 interrupt handles all of the FSK protocol needs. The protocol being implemented is a repeatable pattern, which allows the software to be based on a cycle count. One cycle count is the time period for one bit. In this implementation, 33 bits (11 bits per word, three words) are being sent. During each cycle, check to see the value of the next bit and the step size changes to allow the sine table to be sent out at the mark or space frequencies. After 33 cycles, the

system stops sending DAC values and enters the quiet mode. After 209 cycles, the cycle count is reset and the software starts sending the packet again. With the FSK transmission being handled by the PWM2 interrupt, the CPU's main function is free to be used for other applications. By default, the software example transmits a packet_1 referenced in Table 1, but this can be changed to a packet_0 by setting the packet_to_send variable to a zero.

## 3.5 Test Results

Figure 36 and Figure 37 show that the system created both the mark and space frequencies. Figure 38 shows the complete packet being sent out of the system.

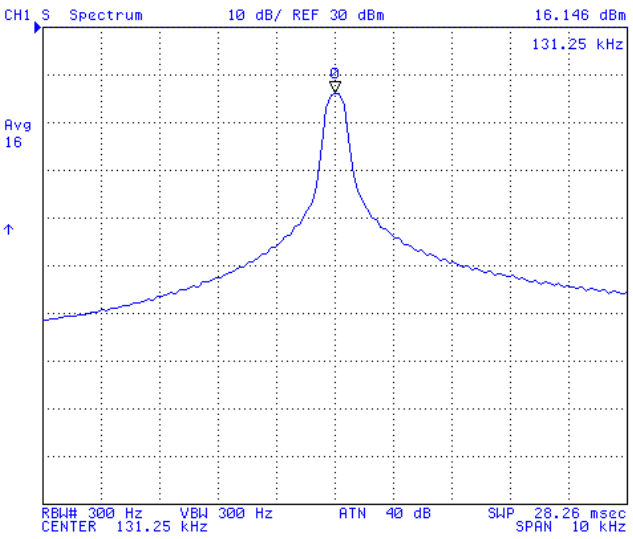



**Figure 36. DAC Mode Mark Frequency Spectrum Analyzer**

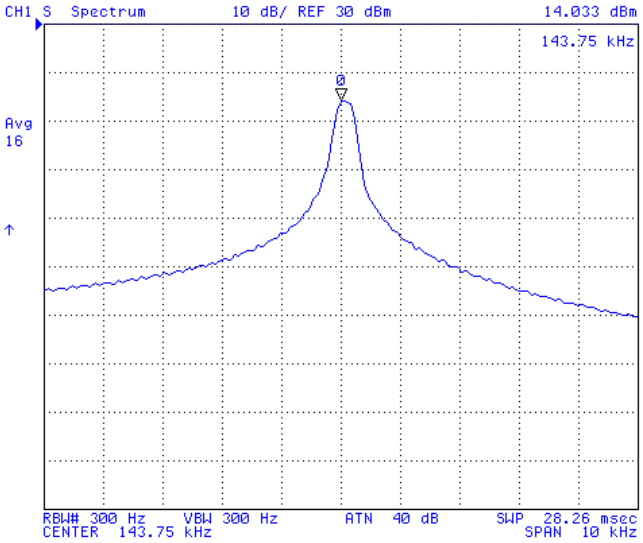



**Figure 37. DAC Mode Space Frequency Spectrum Analyzer**

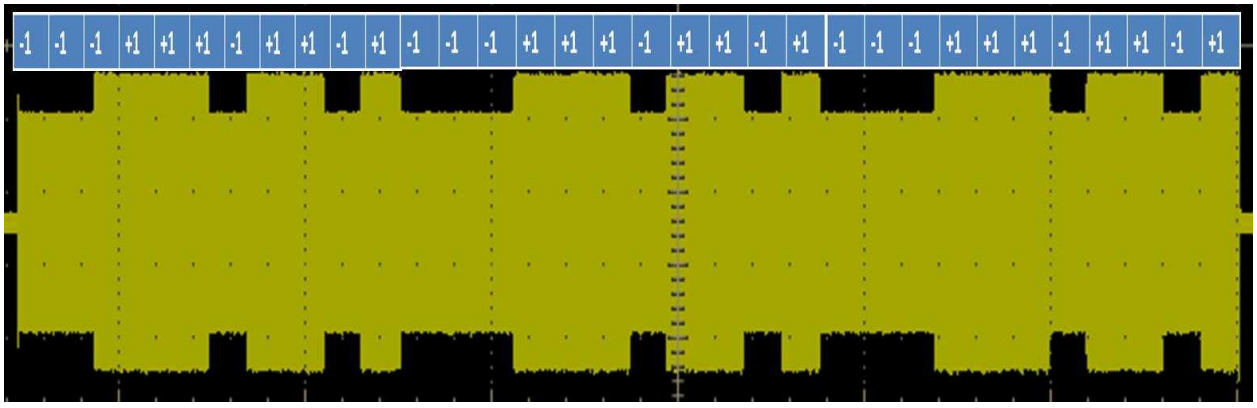



**Figure 38. Full Packet Transmission Waveform**

## 3.6 Receive Path

The C2000 AFE031 interface can also be used as a FSK receiver to translate the transmitted frequency shifted signal into digital data. This section describes one way to create an FSK receiver using the C2000 AFE031 interface.

### 3.6.1 Receive Path Overview

Figure 39 shows the C2000 AFE031 system receive path. There is a significant amount of filtering the input signal must traverse along the path from the transformer on the right to the input of the C2000 ADC on the left. On the AFE side, the AFE031 has vast filtering abilities for these signals.

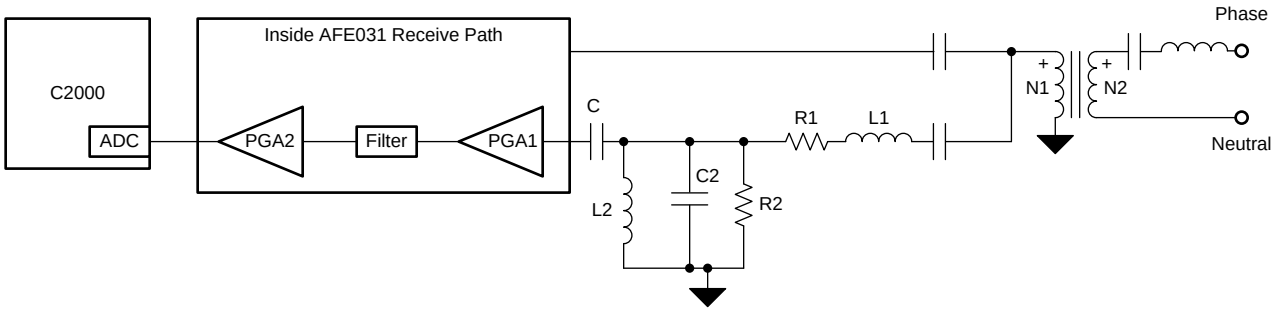



**Figure 39. AFE031 Receive Path Interfaced With C2000™ ADC**

The AFE031 RX path consists of the Rx PGA1, the Rx low-pass filter, and Rx PGA2. Both Rx PGA1 and Rx PGA2 are high-performance PGAs that can be configured through SPI. Rx PGA1 can operate as either an attenuator, providing loss, or an amplifier, providing gain. The gain steps of the Rx PGA1 are 0.25 V/V, 0.5 V/V, 1 V/V, and 2 V/V. The gain steps of the Rx PGA2 are 1 V/V, 4 V/V, 16 V/V, and 64 V/V. For specific Rx PGA gain select register values, see Table 4. Configuring the Rx PGA1 as an attenuator (at gains less than 1 V/V) is useful for applications where the presence of large interference signals are present within the signal band. Attenuating the large interference allows these signals to pass through the analog Rx signal chain without causing an overload; the interference signal can then be processed and removed within the MCU, as necessary.

**Table 4. AFE031 RX PGA Gain Settings**

| BIT NAME | LOCATION (0 = LSB) | DEFAULT | R/W | FUNCTION |
|---|---|---|---|---|
| RX1G-0, RX1G-1 | 0, 1 | 0, 1 | R/W | This bit is used to set the gain of the RX PGA1.<br><br>00 = 0.25 V/V<br>01 = 0.5 V/V<br>10 = 1 V/V<br>11 = 2 V/V |
| RX2G-0, RX2G-1 | 2, 3 | 0, 0 | R/W | This bit is used to set the gain of the RX PGA2.<br><br>00 = 1 V/V<br>01 = 4 V/V<br>10 = 16 V/V<br>11 = 64 V/V |

The Rx filter is a very low noise, unity-gain, fourth-order low-pass filter. The Rx filter cutoff frequency is selectable between CENELEC A or CENELEC B, C, and D modes that is set within the control register. Because the Rx filter is a very low-noise analog filter, two external capacitors, shown in Figure 40, are required to properly configure the Rx filter. Table 5 shows the proper capacitance values for CENELEC A and CENELEC B, C, and D bands.

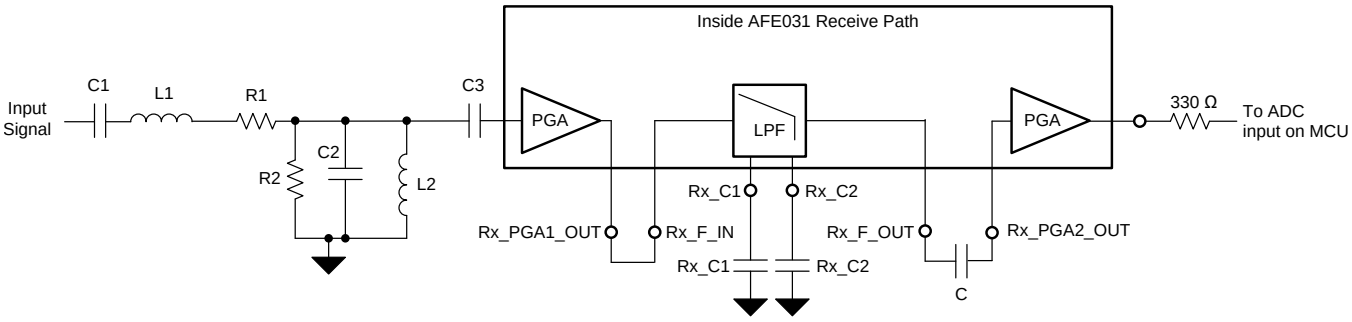



**Figure 40. External Filtering for the AFE031 Receive Path**

**Table 5. Recommended Rx Filter External Capacitor Values**

| FREQUENCY BAND | Rx C1, PIN 24 (pF) | Rx C2, PIN 23 (pF) | CUTOFF FREQUENCY (kHz) |
|---|---|---|---|
| CENELEC A | 680 | 680 | 90 |
| CENELEC B, C, D | 270 | 560 | 145 |

Capacitor Rx C1 is connected between pin 24 and ground, and Rx C2 is connected between pin 23 and ground. For the capacitors shown, TI recommends that these components be rated to withstand the full AVDD power-supply voltage.

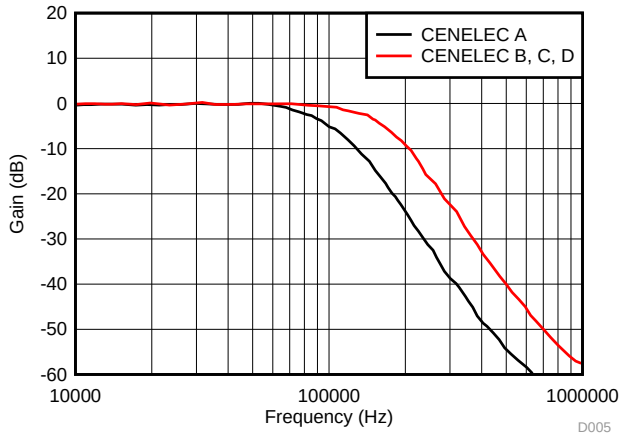



**Figure 41. RX Filter Gain vs Frequency**

Figure 41 shows the gains witnessed at certain frequencies at the output of the RX low-pass filter. This response displayed is under normal AFE03x operating conditions. Attenuation begins at frequencies slightly before the actual cutoff frequencies in Table 5.

An external fourth-order passive passband filter is optional, but recommended for applications where high performance is required. The external passive pass-band filter removes any unwanted, out-of-band signals from the signal path, and prevents them from reaching the active internal filters within the AFE031. Table 6 shows the values needed for CENELEC A or CENELEC B, C, D with a 0-dB passband. The component values used on the BoosterPack are for a CENELEC B, C, D with a 0-dB passband.

**Table 6. Recommended Component Values for Fourth-Order Passive Bandpass Filter (0-dB Passband Attenuation)**

| FREQUENCY BAND | FREQUENCY RANGE (kHz) | CHARACTERISTIC IMPEDANCE (Ω) | R1 (Ω) | R2 (Ω) | C1 (nF) | C2 (nF) | L1 (µH) | L2 (µH) |
|---|---|---|---|---|---|---|---|---|
| CENELEC A | 35 to 95 | 1k | 1k | 10k | 4.7 | 1.5 | 1500 | 4700 |
| CENELEC B,C,D | 95 to 150 | 1k | 1k | 10k | 1.7 | 1 | 1200 | 1500 |
| SFSK | 63 to 74 | 1k | 1 | 10k | 2.7 | 2.2 | 2200 | 2200 |

For other band-pass filter component values with a different passband attenuation, see the *Powerline Communications Analog Front-End* .

### 3.6.2 Receiver Software Implementation

The example program referenced is: boostxl_afe031_f28379d_rx \.

The software example is designed to do the following:

- Continuously sample the received FSK signal after it has traveled through the AFE031 receive path
- Execute a correlation based algorithm on the sampled values to detect if a mark or space frequency is being received
- Decipher mark and space bits based on the frequency detected and its duration
- Store the received bits and packetize them into a desired and usable format

To enable the system to receive in software, complete the following:

1. Set the RX bit in the enable register to 1 to open the AFE031 receive path.
   - Software example function: HAL_afe031_rxEnable();
2. Include the fsk_corr_detect library and related header files in the project:
   - fsk_corr_detect.lib
   - fsk_corr_detector.h
   - fsk_packetization.h

#### *3.6.2.1 Initial Setup and Parameters*

The FSK signals being received follow a set of communication parameters that must be designed around within software. Figure 42 shows the communication parameters of interest.

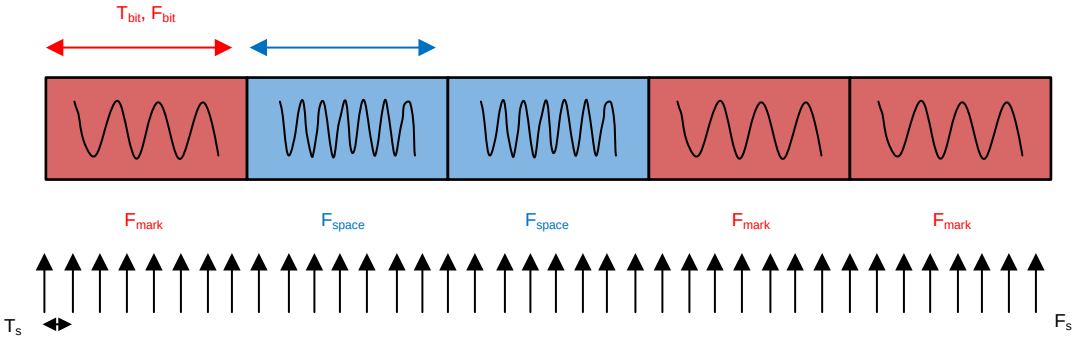



**Figure 42. Received FSK Signal**

The labeled parameters represent the following:

- $T_{bit}$: Bit period
- $F_{bit}$: Bit frequency
- $F_{mark}$: Mark frequency
- $F_{space}$: Space frequency
- $T_S$: Sampling period
- $F_S$: Sampling frequency

For the example program, the communication parameters being followed by default are stated in Table 1. Create a *FSK_CORR_DETECTOR* structure, declared in fsk_corr_detector.h, to hold the parameters necessary for accurate receiving.

```
volatile FSK_CORR_DETECTOR FSK_struct1; // FSK structure
```

The example software and fsk_corr_detect library are designed to detect a set of user-specified frequencies, one mark frequency and one space frequency. These frequencies must be within the frequency band ranges of the AFE031's CENELEC A or CENELEC B, C, D configurations. The example program uses a mark frequency of 131.25 kHz and a space frequency of 143.75 kHz and is meant to be used with the CENELEC B, C, D configuration. Set the *mark_freq* and *space_freq* members of the *FSK_CORR_DETECTOR* structure with these frequencies.

```
FSK_struct1.mark_freq = 131250;    // Mark Frequency Detected
FSK_struct1.space_freq = 143750;   // Space Frequency Detected
```

The C2000's ADC is used to sample the FSK input signal. The sampling frequency, $f_S$, must follow the Nyquist theorem. The input signal must be sampled at a rate of at least 2× the highest signal frequency trying to be detected. That is, if the highest signal frequency to be detected is 100 kHz, $f_S$ must be at least 200 kHz. In the example program, the highest frequency being detected is a 143.75-kHz space frequency and the sampling rate is set to 300 kHz, which is more than the required rate. Set the *isr_freq* member of the *FSK_CORR_DETECTOR* structure to the acceptable $f_S$.

```
FSK_struct1.isr_freq = 300000;     // ADC Sampling frequency
```

A bit decision algorithm is intended to be run at three times the bit frequency. For example, if each bit period is 1 ms long, the bit frequency is 1 kHz, making the desired bit decision frequency 3 kHz. The example program is detecting bits with a period of 5.12 ms making the bit frequency 195.3125 Hz and the desired bit decision frequency 585.9375 Hz. The bit decision frequency in software should be as close as possible to the desired frequency to prevent bit boundary issues. Set the *bit_freq* member of the *FSK_CORR_DETECTOR* structure with this bit decision frequency.

```
FSK_struct1.bit_freq = 586; // Bit decision frequency, 3x bit frequency
```

In summary, Table 7 lists the frequency parameters set for the example program.

**Table 7. Software Frequency Parameters**

| PARAMETER | FREQUENCY |
|---|---|
| Detected mark frequency | 131.25 kHz |
| Detected space frequency | 143.75 kHz |
| Input signal sampling frequency | 300 kHz |
| Bit decision algorithm frequency | 586 Hz (rounded up) |

Set the *detection_threshold* member of the *FSK_CORR_DETECTOR* structure. This value plays a role in tuning the bit detection sensitivity.

```
#define FSK_BIT_DETECTION_THRESHOLD 0.1 // Bit detection threshold value
FSK_struct1.detection_threshold = FSK_BIT_DETECTION_THRESHOLD; // Set threshold
```

Complete the fsk_corr_detect library's initialization based on the member values inputted by calling the corresponding init function.

```
FSK_CORR_DETECTOR_INIT(&FSK_struct1); // Initialize FSK structure
```

Additionally, the format of received information is taken into account by setting the following parameters within software.

- The number of bits that make up a word, #define within fsk_packetization.h:
  - #define NUMBER_OF_BITS_PER_WORD 11
- The number of words that make up a packet, #define within fsk_packetization.h:
  - #define NUMBER_OF_WORDS 3
- The number of total bits within a packet, #define within fsk_corr_detector.h:
  - #define RX_MESSAGE_SIZE 33

#### 3.6.2.2 Interrupt Service Routines

ISRs running at these predetermined frequencies carry out the main functions of the receiver solution. The example program makes use of the C2000's EPWMs and CPU timer to trigger necessary interrupts.

An ADC sampling ISR, configured using EPWM1, is set to trigger at the 300-kHz input signal sampling frequency. The ISR function samples the ADC, scales the sampled value, and passes the scaled value to a library function for signal processing.

A bit-decision ISR, configured using EPWM2, is set to trigger at 585.92 Hz. This frequency is as close to the desired 585.9375 Hz the EPWM module could achieve. The ISR function takes the signal processing work of the previous ADC sampling ISR and checks if a mark or space bit has been detected. If a mark or space bit is detected, then the detected bit is placed into a received message buffer. When the message buffer is full, a flag is set to signify that a full packet has been received.

A message timeout ISR, configured using CPU timer 2, is set to trigger if the user-specified time limit is reached while receiving. By default, this time limit is set to 3 seconds by the RX_MESSAGE_TIMEOUT #define. The timer begins right before the system starts receiving and resets if a packet is received. If a packet is not received within the specified time limit, the ISR triggers and causes the system to stop receiving.

#### 3.6.2.3 Run Time Operation

During the run time, the receiver operates in the following way:

1. The system begins to continuously receive incoming data by starting the EPWMs and CPU timer to trigger interrupts.
   - Example program function: *Start_Receiving();*
2. The ISRs run until the full packet of data is received or the specified timeout is reached.
   - The *rxMessage[]* buffer is filled with the received bits while the ISRs are running.
3. When the packet of data is received or the specified timeout is reached, the system stops receiving data to reduce CPU usage during quiet periods.
   - Example program function: *Stop_Receiving();*
   - The function stops the EPWMs and stops and resets the CPU timer.
4. The received data is then packetized into the desired format.
   - Example program function: *Packetize(int message[], int packet[]);*
   - The function takes the *rxMessage[]* buffer containing 33 received bits and fills the *packet[]* buffer with three 11-bit words by summing up the received bits for each word. The *rxMessage[]* buffer contents are then set to zero.
     - +1 equates to a W1 following the specifications in Table 1
     - -1 equates to a W0 following the specifications in Table 1
   - The function sums up the values of the *packet[]* and saves the sum to the *packet_sum* variable.
     - +3 equates to a packet_1 following the specifications in Table 1
     - -3 equates to a packet_0 following the specifications in Table 1
5. The packetized data is used before the receive process restarts.
   - Example program function: *Visual_Indication();*

- The function causes the LEDs on the BOOSTXL-AFE031 to blink accordingly:
  - Blue LED blinks if a packet_1 is received.
  - Red LED blinks if a packet_0 is received.
  - Red and blue LEDs blink if neither a packet_1 or packet_0 are received, or if a timeout occurs.

## 3.7 Testing Results

The performance of the FSK receiver solution is tested in a controlled lab environment using either the PWM or DAC mode transmitter solution to provide the input FSK signal. The established setup is similar to what is shown in Figure 15 without any coupling circuitry. The purpose of the tests were to confirm that the receiver solution could effectively translate a received FSK signal into the original digital information.

When the guidelines discussed in Section 3.6.2 are appropriately followed, a C2000 MCU can take an FSK input signal and accurately decode each bit of data it contains. Figure 43 shows the results, where a packet of thirty-three unique bits are successfully captured.

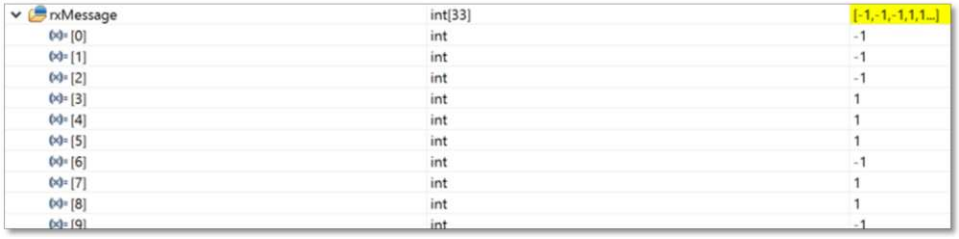



**Figure 43. Packet of Received Bits of Data**

This captured data is in the form of an 11-bit code word transmitted three consecutive times. Each code word having a sum of one is then packetized in the format shown in Figure 44.

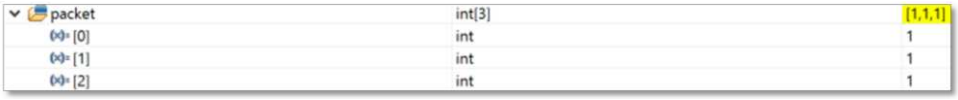



**Figure 44. Packetized Data**

## 3.8 System Utilization

Table 8 lists the resources consumed by the C2000 MCU when being used as an FSK receiver.

**Table 8. C2000™ Resources Used**

| RESOURCE NAME | TYPE | PURPOSE | USAGE AND RESTRICTIONS |
|---|---|---|---|
| ADCINA | Module/IO | ADC input for sampling input signal | Restricted to ADC inputs that are available an accessible |
| EPWM1 | Module | Triggers the interrupt for the signal sampling routine | Can be configured a number of ways to meet desired ISR frequency |
| EPWM2 | Module | Triggers the interrupt for running the bit decision routine | Can be configured a number of ways to meet desired ISR frequency |
| SPI | Module/IO | For accessing the AFE03x's registers during initialization | Only needed for initialization for the RX solution |
| CPU Timer 2 | Module | Can be used to create a timeout when no information has been received for a certain amount of time | ISR may have lower priority than other set ISRs |

Table 9 lists the number of CPU cycles consumed by each fsk_corr_detect library function.

**Table 9. Library Function CPU Cycles**

| FUNCTION NAME | DESCRIPTION | CPU CYCLES | TYPE |
|---|---|---|---|
| FSK_CORR_DETECTOR_INIT | Initializes variables used by the FSK library, based off the frequency parameters the user sets within a fsk_corr_detector structure | 60 | Initialization |
| FSK_CORR_DETECTOR_RUN | Performs necessary calculations on the sampled ADC values to demodulate the input signal | 59 | Run time |
| FSK_CORR_DETECTOR_OverSampl_RUN | Performs the logic to decipher if a bit has been received | 134 | Run time |
| Packetize | Takes a received message data buffer and builds usable code words and packets | 1381 | Run time |

The run-time CPU utilization of the software solution, when being used in its default state and receiving information specified in Table 1, can be calculated using the information in Table 10.

**Table 10. Software ISR and Function Usage**

| ISR OR FUNCTION | AVERAGE CYCLES | FREQUENCY OF EXECUTION |
|---|---|---|
| ADC Sampling ISR | 74 | 300 kHz |
| Bit-decision ISR | 175 | Approximately 586 Hz |
| Packetization Function | 1381 | Approximately 1 Hz |

CPU utilization equation:

$$\text{CPU Utilization} = ((74 \times f_S + 175 \times 3 \times f_{bit} + 1381 \times f_{Packetization}) / f_{CPU}) \times 100\% \quad (10)$$

CPU utilization at F28379D's 200-MHz clocking frequency:

$$\text{CPU Utilization} = ((74 \times 300\ \text{kHz} + 175 \times 586\ \text{Hz} + 1381 \times 1\ \text{Hz}) / 200\ \text{MHz}) \times 100\% = 11.15\% \quad (11)$$

## 3.9 Device Dependency and Porting

While the FSK receiver solution is built for and tested on the F2837xD, the solution should directly port to devices that have floating point unit (FPU) and trigonometric math unit (TMU) support. The fsk_corr_detect library's functionality is dependent on the FPU and TMU being present. Devices with both FPU and TMU support include the F2837xD, F2837xS, F2807x, and F28004x.

## 3.10 Tuning and Calibration

The C2000's AFE031 system can be tuned and calibrated for receiving within software and hardware. These concepts are discussed in the following subsections.

## 3.11 Setting the AFE03x's PGAs

The AFE031's PGAs along the RX path are used to amplify or attenuate the input signal to be within the desired voltage range before entering the C2000's ADC. It is desired for the signal at the ADC pin to be within the range defined by the ADC's reference voltage. For example, the F28379D LaunchPad's reference voltage is 3 V; therefore, the signal should be within a range of 0 V to 3 V to prevent clipping.

For best results, this signal should be close to this voltage range, without going over or under, to use the full resolution of the ADC. Use the AFE031's PGAs to tune the signal to meet this criteria using the gain value settings in Table 4. For how to set the RX PGA values within software, see Step 10 of Section 3.1.2.1.1.

## 3.12 Automatic Gain Control (AGC)

If the receiver system is to be used within an application that has an input signal with inconsistent amplitudes, it may be necessary to implement some form of automatic gain control (AGC) to manage the signal before it reaches the C2000's ADC input. The AGC would be a closed-loop feedback system that measures the amplitude of the input signal, or a related response, and based on what is observed amplify or attenuate the signal.

The AFE031's internal PGAs could be leveraged to dynamically amplify or attenuate the signal to be within the desired amplitude range. For gain select times of the two PGAs along the RX path, see *Powerline Communications Analog Front-End* .

Another approach would be to perform the AGC externally before the signal enters the C2000 AFE031 system. However, this requires additional circuitry involving a variable gain amplifier (VGA) to be added.

## 3.13 Setting the Bit Detection Threshold

The *detection_threshold* member of the FSK_CORR_DETECTOR structure is in direct relation to the accuracy of bit detections. The threshold must be set to a value between 0 and 1. If set to a value too high, then bits will not be detected at all, but if set too low then there could be confusion as to what constitutes a mark or space bit and cause bit errors. It may take some trial and error to find a value that consistently detects bits without errors.

## 3.14 Setting the Bit Detection Threshold (Filtered)

The detection_threshold member of the FSK_CORR_DETECTOR structure is directly related to the accuracy of bit detections. The threshold must be set to a value between 0 and 1. If set to a value too high, then bits will not be detected at all; however, if set to a value too low, then there could be confusion as to what constitutes a mark or space bit resulting in bit errors.

In the software example, the *bit_detect_strength* variable that is assigned a value by the FSK_CORR_DETECT_STRENGTH library function provides some insight into what the detection_threshold member should be set to. The bit_detect_strength variable is set to a value after each execution of the library function. The detection_threshold member needs to be given a value at or below the bit_detect_strength calculated while receiving a mark or space signal. Compare the bit_detect_strength value witnessed when receiving a mark signal to the value witnessed when receiving a space signal. The detection_threshold should be set to a value at or below the lower of the two values witnessed. It may take some trial and error to find a value that consistently detects bits without errors.

## 3.15 FSK Correlation Detector Library

For inquiries related to the FSK correlation detector library used in the software example, email the following address:
C2000-fsk_rx_source_access@list.ti.com

To request access to the library's source code (contingent on end application review and export control approval from Texas Instruments, Inc), follow the instructions at:
https://www.ti.com/licreg/docs/swlicexportcontrol.tsp?form_id=268791&prod_no=C2000-AFE031_FSK_RX_SOURCE&ref_url=c2000.

# 4 Design Files

## 4.1 Schematics

To download the schematics, see the design files at TIDA-060001.

## 4.2 Bill of Materials

To download the bill of materials (BOM), see the design files at TIDA-060001.

## 4.3 PCB Layout Recommendations

For thermal considerations for PCB layout, see *Powerline Communications Analog Front-End* .

### 4.3.1 Layout Prints

To download the layer plots, see the design files at TIDA-060001.

## 4.4 Altium Project

To download the Altium Designer® project files, see the design files at TIDA-060001.

## 4.5 Gerber Files

To download the Gerber files, see the design files at TIDA-060001.

## 4.6 Assembly Drawings

To download the assembly drawings, see the design files at TIDA-060001.

# 5 Software Files

To download the software files, see the design files at C2000Ware.

# 6 Related Documentation

1. Texas Instruments, *Spread-Frequency Shift Keying Power Line Modem Software Architecture Application Report*
2. SunSpec Alliance, SunSpec Homepage
3. Texas Instruments, *Analog Front-End Design for a Narrowband Power-Line Communications Modem Using the AFE031 Application Report*
4. Texas Instruments, *AFE031 Powerline Communications Analog Front-End Data Sheet*
5. Texas Instruments, *DC Power-Line Communication Reference Design*

## 6.1 Trademarks



# 7 Terminology

**PLC—** Powerline communications

**FSK—** Frequency shift keying

**HRPWM—** High-resolution pulse width modulation

**AFE—** Analog front end

## 8 About the Authors

**VINCENT RODRIQUEZ** is an applications engineer within the SimpleLink™ Wi-Fi® group, where he works on hardware and software development, creating collateral and reference designs showcasing the CC31xx/CC32xx Wi-Fi devices. Vince graduated from Texas A&M University with a bachelor's of science in electronic systems engineering technology in 2016.

**ERROL LEON** is an applications engineer that specializes in analog signal chain design at Texas Instruments, where he is responsible for developing application notes, evaluation modules, and reference design solutions. Errol earned his master of science in electrical engineering (MSEE) and bachelor (BSEE) from California Polytechic University San Luis Obispo. Errol is also a member of the Institute of Electrical and Electronics Engineers (IEEE).

**KEVIN ALLEN** is an applications engineer within the C2000 Group, where he works on creating collateral and supporting C2000 devices. Kevin graduated from the University of Texas Dallas with a bachelor's of science in electrical engineering.

## 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