User's Guide
SLLU216–July 2019

SPI to CAN FD SBC + LIN Transceiver BoosterPack User's Guide

The SPI to CAN FD SBC + LIN BoosterPack™ features the TCAN4550-Q1 CAN FD controller with integrated transceiver providing microcontrollers without an integrated CAN FD controller, or those needing additional channels, access to CAN FD applications through a SPI interface. LIN applications can also be developed using the TLIN2029-Q1 fault protected LIN transceiver and the MCU UART port making this BoosterPack an ideal starting point for any CAN FD or LIN system.

Contents

List of Figures

List of Tables

Trademarks

1 Introduction

The SPI to CAN FD SBC + LIN BoosterPack features the TCAN4550-Q1 CAN FD controller with integrated transceiver providing microcontrollers without an integrated CAN FD controller, or those needing additional channels, access to CAN FD applications through a SPI interface. LIN applications can also be developed using the TLIN2029-Q1 fault protected LIN transceiver and the MCU UART port making this BoosterPack an ideal starting point for any CAN FD or LIN system.

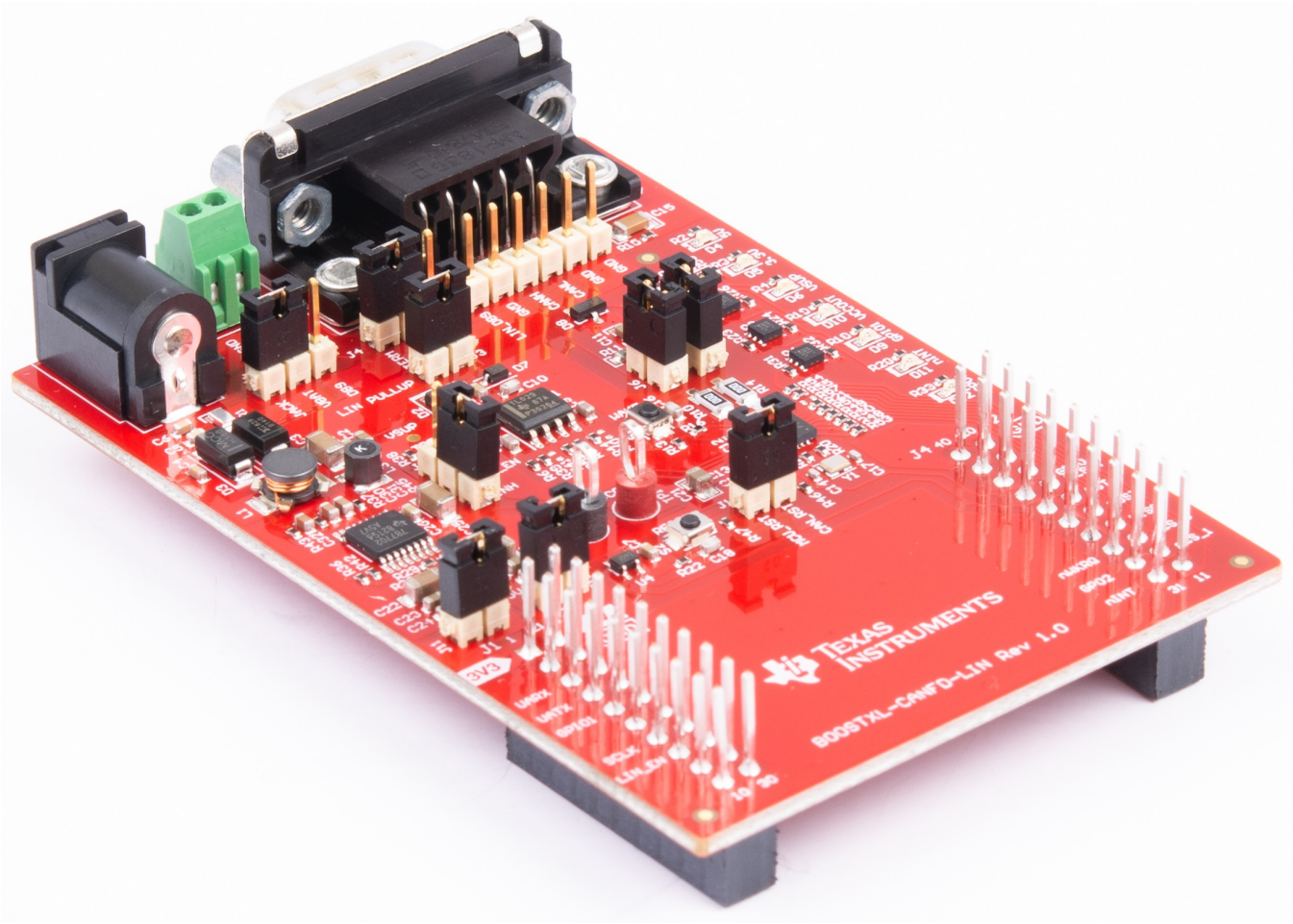
This user's guide describes features of the SPI to CAN FD SBC + LIN BoosterPack, explaining the board hardware details, functions and locations of jumpers and connectors.

1.1 Features

- Operates from a batter voltage input V_{BAT} (6 V to 24 V)
  - Polarity protected and EMC filtered to create V_{SUP} supply rail
- TCAN4550-Q1 CAN FD controller with integrated transceiver
  - Supports Classical and CAN FD applications
  - SPI interface
  - Crystal oscillator (40 MHz)
  - Reset from push-button switch, MCU GPIO pin, or linked to LaunchPad™ global reset
  - Sleep (low power) mode
  - WAKE from push-button switch or CAN bus Wake Up Pattern (WUP)
  - V_{CCOUT} 5-V LDO output
- TLIN2029-Q1 Fault protected LIN transceiver
  - Supports Master or Slave mode configurations
  - MCU UART Interface
- TPS7B7702-Q1 Automotive dual-channel LDO
  - 3.3-V and 5-V supply rails
  - Can supply both the BoosterPack and LaunchPad
  - Enable linked to TCAN4550 Inhibit (INH) pin or pulled high
- Common DB9 connector for Supply Voltage, CAN FD, and LIN bus signals
  - Additional header for easy monitoring of bus signals
- Wire terminal for LIN bus signals
- CAN bus termination with disconnect headers
- Common-mode choke (bypassed with 0-Ω resistors by default)
- Status LEDs

2 Hardware description

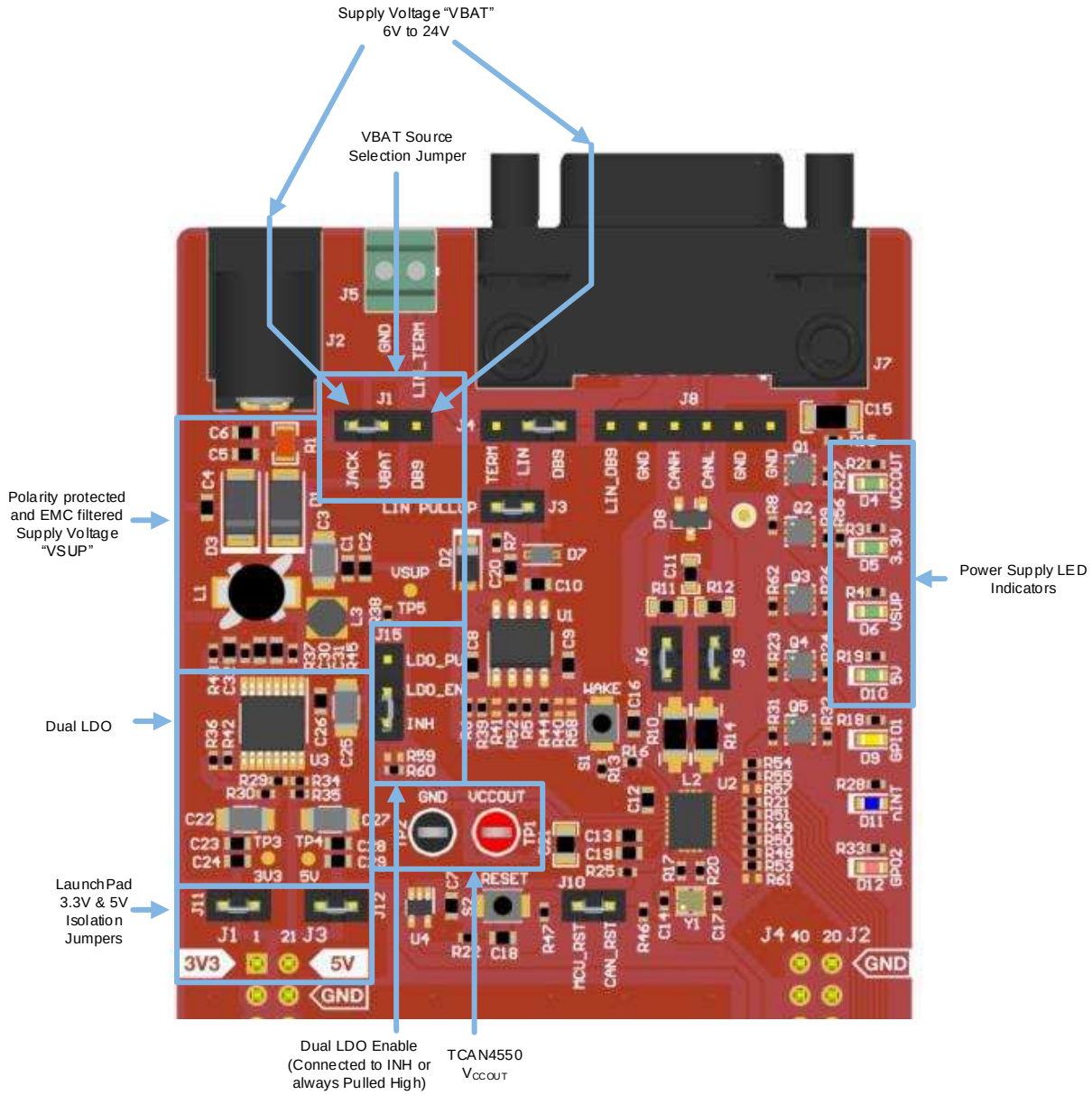
Figure 1. BOOSTXL-CANFN-LIN



2.1 Power

The following sections describe the power supply components and features.

Figure 2. Power components and features



2.1.1 BoosterPack supply voltage

This BoosterPack requires an external Supply Voltage to be provided in the range of 6 V to 24 V and cannot be powered exclusively through the 3.3 V and 5 V supplies of the LaunchPad. CAN and LIN transceivers are commonly found in automotive applications and operate across a wide recommended supply voltage range corresponding to a vehicle battery. This supply voltage is used directly by the transceivers and converted down on the board to supply the digital voltage for the lower voltage digital devices which are typically ≤ 5 V. The TCAN4550-Q1 operates across a range of 6 V to 24 V with an absolute maximum voltage of 40 V, the TLIN2029-Q1 operates across a range of 4 V to 48 V with an absolute maximum voltage of 60 V, and the TPS7B7702-Q1 operates across a range of 4.5 V to 40 V with an absolute maximum voltage of 45 V. Since all three of these devices share this same supply, the voltage must be within all device limits for proper operation and is recommended to be in the range 6 V to 24 V.

The board uses a 30-V Zener diode as protection from excessive supply voltages, along with a reverse-blocking Schottky diode and other EMC-filtering components on this supply voltage commonly found in system applications. The board refers to the raw voltage supplied to the board as V_{BAT} to correspond with a system battery voltage, and the voltage after the supply filter as V_{SUP} which is the actual voltage supplied to the device pins. It is common to supply both a supply voltage and the bus communication through a single wire harness and this supply voltage can be supplied through pin 9 of the DB9 connector J7, or from a standalone supply through the DC barrel jack J2. Header J1 allows a shunt to be placed between the center pin and the selected voltage source.

2.1.2 TCAN4550-Q1

2.1.2.1 V_{SUP} Pin

The TCAN4550-Q1 is powered through the V_{SUP} pin and supplies the internal regulators for the digital core, CAN transceiver, and optional V_{CCOUT}. A 330-nF capacitor is required on the LDO Filter pin FLTR for proper operation as described in the data sheet and seen in the schematics.

2.1.2.2 V_{IO} Pin

The V_{IO} pin provides the digital I/O voltage to match the MCU I/O voltage to prevent the need for level-shifting of signals across different MCUs. The V_{IO} pin supports the SPI I/O pins, GPIO1 and GPO2 pins, as well as the oscillator block supporting the crystal oscillator or CLKIN pins.

2.1.2.3 V_{CCOUT} Pin

The TCAN4550-Q1 devices provide up to 70 mA of current on the V_{CCOUT} pin from the 5-V internal LDO without impacting the CAN transceiver performance, except when in sleep mode and when the regulator is disabled. This voltage can be used to power an MCU or other supporting circuitry that is not needed during sleep mode. At least 10 µF of capacitance to ground is required on the V_{CCOUT} pin, as described in the data sheet and seen in the schematic.

2.1.2.4 GND Pin

The thermal pad and pin 13 of the TCAN4550-Q1 device are ground and are connected to the ground plane of the board to support heat dissipation.

2.1.3 TLIN2029-Q1

2.1.3.1 V_{SUP} Pin

The TLIN2029-Q1 is powered through the V_{SUP} pin which is typically connected to the battery through an external blocking diode (D1) and is included in this design. If there is a loss of power on the board, or the board is put into Sleep Mode, the device has extremely low leakage from the LIN pin and does not load the bus down. This is optimal for LIN systems in which some of the nodes are unpowered (ignition supplied) while the rest of the network remains powered (battery supplied).

2.1.3.2 GND Pin

GND is the TLIN2029-Q1 ground connection. The device can operate with a ground shift as long as the ground shift does not reduce the V_{SUP} below the minimum operating voltage. If there is a loss of ground at the ECU level, the device has extremely low leakage from the LIN pin, and does not load the bus down. This is optimal for LIN systems in which some of the nodes are unpowered (ignition supplied) while the rest of the network remains powered (battery supplied).

2.1.4 TPS7B7702-Q1 (External LDO)

Although it is not required for all applications, the BoosterPack includes a dual-channel LDO to provide the 3.3-V V_{IO} voltage reference for the TCAN4550-Q1 and a 5-V utility voltage to source the LEDs. The LaunchPad containing the MCU can be powered through the USB debugger which produces both 3.3-V and 5-V rails. However, once the debugger is no longer needed, the LaunchPad 3.3-V and 5-V rails can be isolated from the debugger and supplied from the BoosterPack through this LDO as is common for real system applications and allowing for the entire system to be placed into a low power sleep mode.

The Enable pin of the TPS7B7702-Q1 is connected to the Inhibit pin of the TCAN4550-Q1 and is pulled low when the TCAN4550-Q1 is placed into Sleep Mode. When a valid wake-up request is seen by the TCAN4550-Q1, either through a Wake-Up Packet (WUP) on the CAN bus or through a state change on the WAKE pin, the TCAN4550-Q1 transitions to Standby Mode and drive the Inhibit pin high re-enabling the LDO output 3.3-V and 5-V output channels which in turn sources power to MCU on the LaunchPad.

CAUTION

The BoosterPack's 3.3-V and 5-V rails should not be connected to the LaunchPad if the LaunchPad is powered from the debugger. Ensure the shunts on J11 and J12 are removed when the LaunchPad is powered from the debugger.

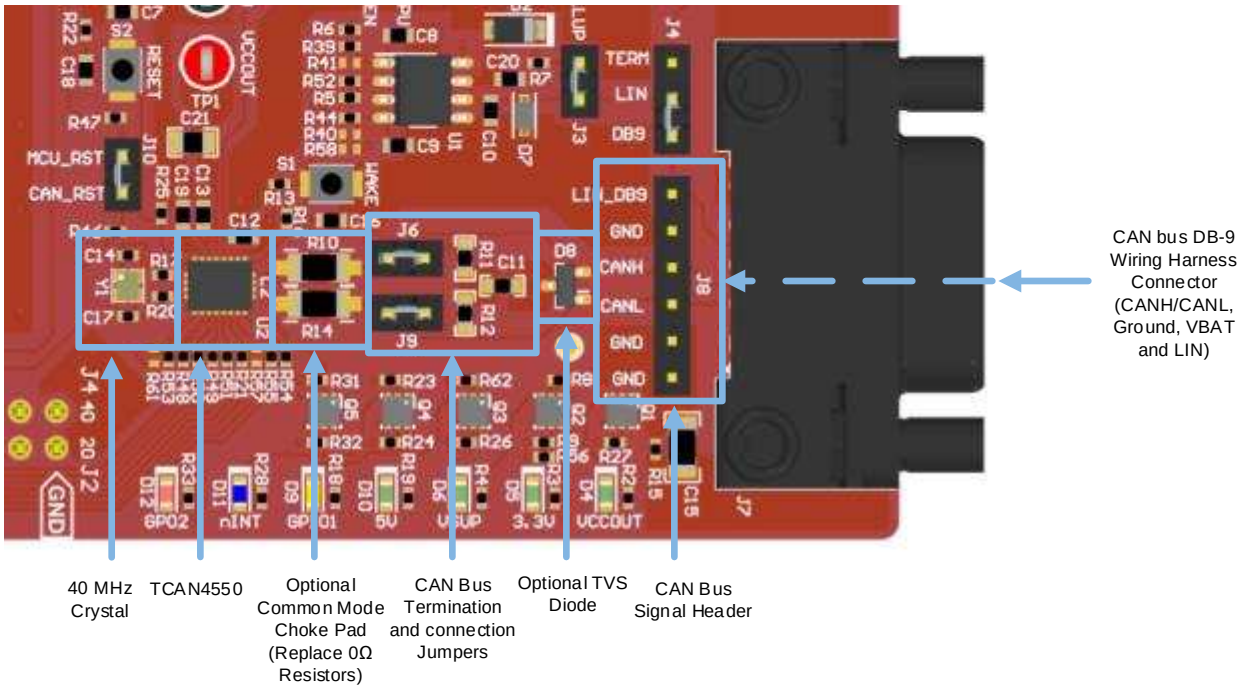
While the TLIN2029-Q1 does not support a wake and inhibit feature, the BoosterPack was designed to support LIN transceivers that do contain those features allowing the LIN transceiver to also control the LDO's enable pin in conjunction with or instead of the TCAN4550-Q1.

NOTE: If the BoosterPack is used only for LIN, the 3.3-V and 5-V LDO may become disabled by the TCAN4550-Q1 entering sleep mode. To prevent this, configure the LDO enable pins to be pulled High through pullup resistors by moving the shunt on J15 to short the LDO_EN and LDO_PU pins together.

2.2 CAN

The following sections describe the features of the CAN bus.

Figure 3. CAN bus components and features



2.2.1 CAN Bus DB-9 Connector (J7)

The CAN bus signals CANH and CANL are accessible through the DB-9 connector J7, which follows the industry-standard pin mapping. This mapping allows for easy integration into an existing system. The external supply voltage V_{BAT}, ground, and connector shield ground connections are also included in the connector J7.

2.2.2 CAN Bus Signal Header (J8)

The CAN bus signals are also available on header J8. This may be more convenient in a lab environment or to attach scope probes to monitor the bus connected the EVM through the DB-9 connector.

2.2.3 CAN TVS Diode (D8)

A TVS diode footprint has been added to the BoosterPack and diodes installed to provide maximum protection when used in environments that may not have proper ESD controls. However, the use of these external diodes are considered optional and the TCAN4550-Q1 contains robust internal ESD protection.

2.2.4 Termination

The CAN standard requires termination of the CAN bus at each end with 120 Ω of resistance between the CANH and CANL pins. A common technique, also used on this BoosterPack, is to implement a split termination using two 60-Ω resistors in series and a 4.7-nF capacitor to ground. The capacitor to ground provides a path for filtering common-mode noise out of the signals that can come from sources such as asymmetry between the CANH and CANL signals. Common-mode noise is a result of unequal lengths between the CANH and CANL wires, asymmetrical bus loading, or other sources. If the BoosterPack is used in a lab environment or as one of the end nodes of the CAN bus, the termination can be enabled by placing shunts on jumpers J6 and J9 to connect the termination resistors to the CANH and CANL lines. However, if the BoosterPack is used on a bus with existing termination resistors, or does not require local termination, removing the shunts on J6 and J9 disconnects the local termination from the CANH and CANL lines.

2.2.5 Common-Mode Choke

Many applications have system-level electromagnetic compatibility (EMC) requirements that may require the use of a common-mode choke on the CAN pins. A typical common-mode choke pad has been added to the BoosterPack for the user to add their desired choke. However, by default, 0-Ω resistors have been placed across the choke pins to connect the CANH and CANL pins of the TCAN4550-Q1 device with the rest of the CAN bus connectors and features of the BoosterPack. To add a choke to the BoosterPack, simply remove resistors R10 and R15 from the BoosterPack and install the choke in their place.

2.2.6 Clocking

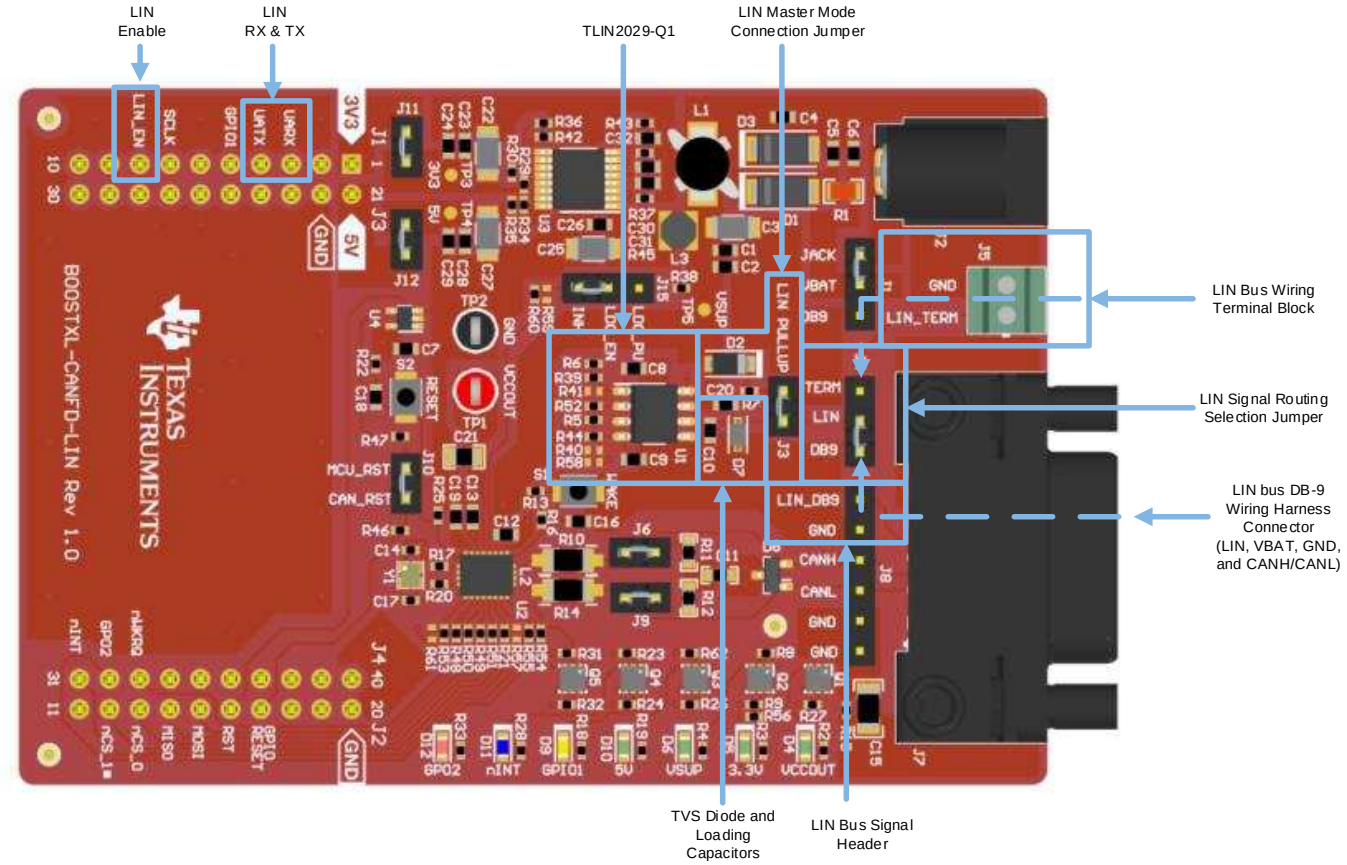
The TCAN4550-Q1 requires either a crystal oscillator or a single-ended clock to run its digital core. A 40-MHz crystal oscillator is installed and connected to the OSC1/CLKIN and OSC2 pins by default and is available whenever the V_{IO} 3.3-V supply is present. CAN FD data rates of 2 Mbps or greater require a clock frequency of 40 MHz to generate a time quanta small enough to allow for accurate data sampling of CAN bits. For slower applications with data rates < 2 Mbps, the crystal could be replaced with a 20-MHz version if desired, but the 40-MHz crystal is still recommended.

Since a 40-MHz single-ended clock source is not commonly available on most LaunchPads, there is no provision on the BoosterPack to support a single-ended clock and the TCAN4550-Q1 must rely on the crystal oscillator for its clock source.

2.3 LIN

The following sections describe the components and features of the LIN bus.

Figure 4. LIN bus components and features



2.3.1 LIN Bus DB-9 Connector (J7)

The LIN bus signal is accessible through an unused pin (pin 8) of the DB-9 connector J7, which follows the industry-standard CAN pin mapping. This mapping allows for easy integration into an existing system. The external supply voltage V_{BAT}, ground, and connector shield ground connections are also included in the connector J7. The LIN bus signal has also been included on header J8 along with the other CAN signals allowing an additional connection or monitoring point.

To route the LIN bus signal to the DB-9 connector and header J8, place a shunt between the pins labeled LIN and DB9 on jumper J4.

CAUTION

Pin 8 of the DB-9 connector pinout is considered reserved as an upgrade path for future use and could be used for different purposes by other hardware. If the BoosterPack is to be connected to other types of hardware, the compatibility of pin 8 as well as all the other pins, should be checked for compatibility to avoid the possibility of damage.

2.3.2 LIN Terminal Block (J5)

The LIN bus signal is accessible on the 2-pin terminal block J5. This is a convenient method for connecting the BoosterPack to other hardware without the need for specific connectors and only requires basic stripped twisted-pair wire. To route the LIN bus signal to the terminal block J5, place a shunt between the the pins labeled LIN and TERM on jumper J4.

2.3.3 LIN Master/Slave Mode

The BoosterPack kit provides users with the ability to evaluate TI's TLINx02x-Q1 family of the single-channel, LIN transceivers. The BoosterPack allows both master and slave mode applications to be evaluated through the use of a single jumper (J3) that connects or disconnects the external 1-kΩ pullup resistor and series diode required in master mode from the LIN bus. To configure the BoosterPack for master mode, place a shunt on J3 to connect the external 1-kΩ resistor and series diode to the LIN bus per the LIN specification. To configure the board for slave mode, remove the shunt on J3 to disconnect the external 1-kΩ resistor and series diode from the LIN bus per the LIN specification.

2.3.4 LIN IO pins

The received data (RXD) pin is an open-drain output that requires an external pull-up resistor (R6) to the MCU voltage rail. The BoosterPack uses the 3.3 V rail for the digital IO levels and includes a 10-kΩ resistor to 3.3 V. The enable (EN) control pin has an external 10-kΩ pull-down resistor (R5) to keep the transceiver in Sleep Mode until this pin is driven to 3.3 V by the MCU to place it into Normal Mode. Non-populated capacitor pads (C8 and C9) are also available on the TXD and RXD pins to accommodate a variety of different tests that may require a capacitive load.

The BoosterPack also supports versions of the TLINx02x-Q1 device family with inhibit and wake features which can be evaluated with a simple hardware assembly modification.

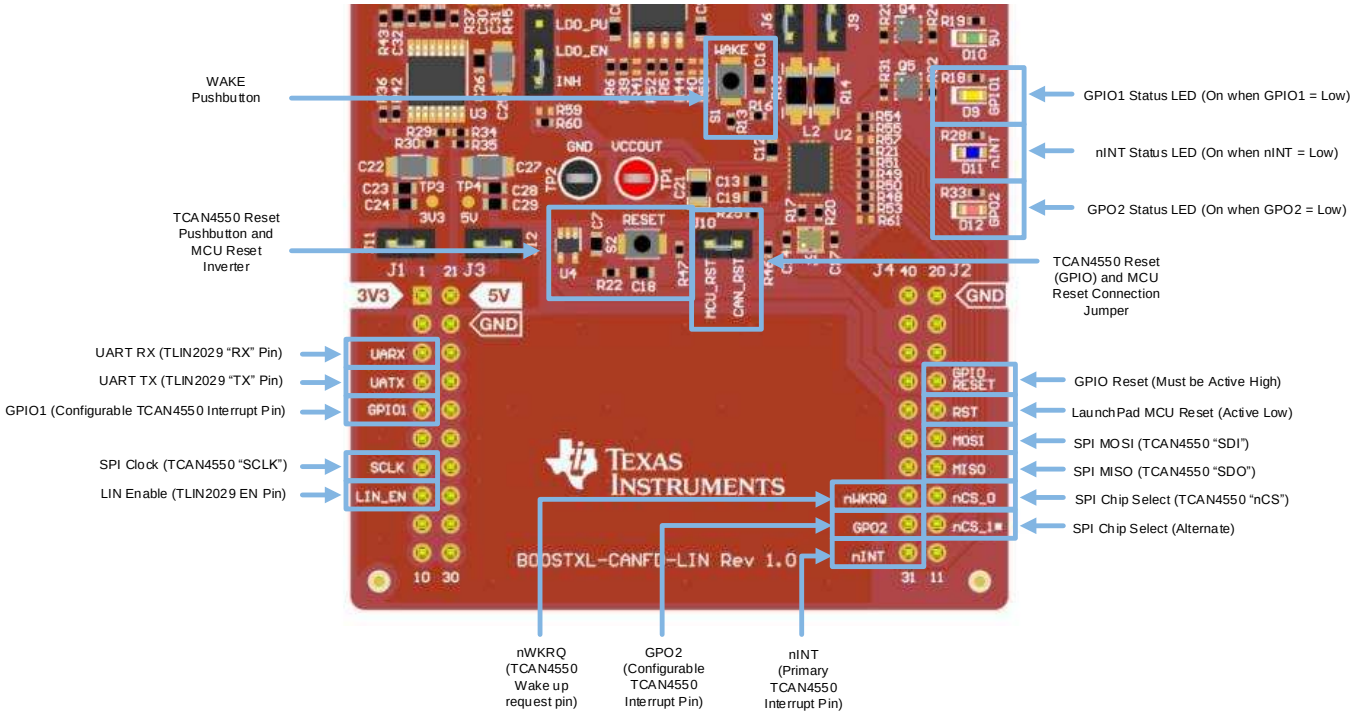
2.3.5 LIN TVS Diode (D7)

A SOD323 TVS diode footprint has been added to the BoosterPack and diodes installed to provide maximum protection when used in environments that may not have proper ESD controls. However, the use of these external diodes are considered optional and the TLIN2029-Q1 contains robust internal ESD protection.

2.4 MCU interface (SPI/GPIO)

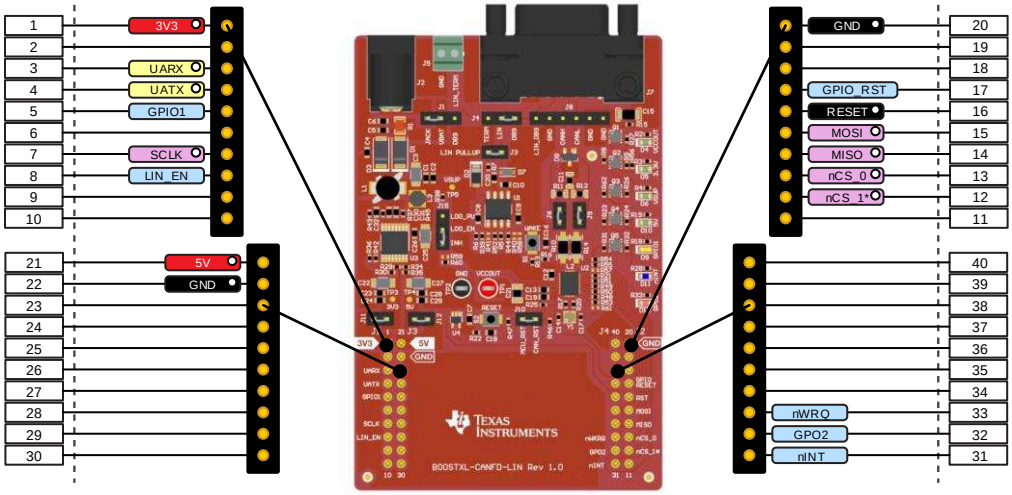
The following sections describe the features of the MCU/LaunchPAD interface

Figure 5. MCU interface components and features



2.4.1 BoosterPack Pinout

Figure 6. Pinout



The BoosterPack kit adheres to the 40-pin LaunchPad and BoosterPack pinout standard. This standard was created to aid compatibility between LaunchPad development kit and BoosterPack module tools across the TI ecosystem.

The 40-pin standard is compatible with the 20-pin standard that is used by other LaunchPad kits like the MSP-EXP430G2 LaunchPad development kit. This allows for 40-pin BoosterPack modules to be used with 20-pin LaunchPad kits with some limited functionality.

The BOOSTXL-CANFD-LIN supports BoosterPack module stacking with its male and female BoosterPack headers. Many BoosterPack modules can be stacked onto your LaunchPad for additional functionality.

More information about compatibility can also be found at http://www.ti.com/launchpad.

2.4.2 TCAN4550-Q1 SPI

The SPI communication uses a standard SPI interface. Physically the digital interface pins are nCS (Chip Select Not), SDI (Slave Data In), SDO (Slave Data Out) and SCLK (SPI Clock). Each SPI transaction is a 32 bit word containing a command byte followed by two address bytes and length bytes. The data shifted out on the SDO pin for the transaction always starts with the Global Status Register (byte). This register provides the high level status information about the device status. The two data bytes which are the 'response' to the command byte are shifted out next. Data bytes shifted out during a write command is content of the registers prior to the new data being written and updating the registers. Data bytes shifted out during a read command are the current content of the registers and the registers is not updated.

The SPI input data on SDI is sampled on the low to high edge of the SCLK. The SPI output data on SDO is changed on the high to low edge of the SCLK.

2.4.2.1 Chip Select Not (nCS)

This input pin is used to select the device for a SPI transaction. The pin is active low, so while nCS is high the SDO pin of the device is high impedance allowing a SPI bus to be shared with other devices. When nCS is low the SDO driver is activated and communication may be started. The nCS pin must be held low for the duration of the SPI transaction. A special feature on this device allows the SDO pin to immediately show the Global Fault Flag on a falling edge of nCS.

There are two pins in the LaunchPad and BoosterPack pinout standard that could be used for the SPI Chip Select defined as SPI_CS. Both of these pins are supported on the BoosterPack with the header pin 13 connected to the TCAN4550-Q1 nCS pin by default. If there is a conflict with this pin the other supported chip select pin can be used by removing the 0-Ω resistor R21 and installing it on R57 instead. These pins are labeled on the BoosterPack nCS_0 and nCS_1 next to pins 12 and 13 of the LaunchPad board-to-board headers.

NOTE: The Chip Select signal must transition back to a high following the end of the data transaction and cannot be held low indefinitely as is sometimes common practice when only a single device is on the SPI bus. There are 2 primary reasons for this:

1.) The Global Status Register (byte) is always shifted out on the SDO pin for every SPI transaction starting with the first clock cycle following the chip select high-to-low transition.

2.) The device counts the number of bits received on the SDI pin which must be a multiple of 32 bits between the chip select transition to low at the beginning of the transaction and then back to high at the completion of the transaction. If the number of bits is not a multiple of 32 bits, the last word of the transfer is ignored and the SPIERR flag is set.

2.4.2.2 SPI Clock Input (SCLK)

This input pin is used to input the clock for the SPI to synchronize the input and output serial data bit streams. The SPI Data Input is sampled on the rising edge of SCLK and the SPI Data Output is changed on the falling edge of the SCLK.

Pin 7 of the LanchPad board-to-board header is used for the SPI SCLK as is defined in the LaunchPad and BoosterPack pinout standard as SPI_CLK.

2.4.2.3 SPI Slave Data Input (SDI)

This input pin is used to shift data into the device. Once the SPI is enabled by a low on nCS the SDI samples the input shifted data on each rising edge of the SCLK. The data is shifted into a 32 bit shift register. If the command code was a write, the new data is written into the addressed register only after exactly 32 bits have been shifted in by SCLK and the nCS has a rising edge to deselect the device. If there are not exactly a multiple of 32 bits shifted in to the device during one SPI transaction (nCS low) the last word of the transfer is ignored, the SPIERR flag is set.

NOTE: Due to needing multiples of 32 bits on each SPI transaction the device should be wired for parallel operation of the SPI as a bus with control to the device via nCS and not as a daisy chain of shift registers.

Pin 15 of the LanchPad board-to-board header is used for the SPI SDI as is defined in the LaunchPad and BoosterPack pinout standard as SPI_MOSI.

2.4.2.4 SPI Slave Data Output (SDO)

This pin is high impedance until the SPI output is enabled via nCS. Once the SPI is enabled by a low on nCS, the SDO is immediately driven high or low showing the Global Fault Flag status which is also the first bit (bit 32) to be shifted out if the SPI is clocked. Once SCLK begins, on the first low to high edge of the clock the SDO retains the Global Fault Flag which is bit 31 of the shift. On the first falling edge of SCLK, the shifting out of the data continues with each falling edge on SCLK until all 32 bits have been shifted out the shift register.

Pin 14 of the LanchPad board-to-board header is used for the SPI SDO as is defined in the LaunchPad and BoosterPack pinout standard as SPI_MISO.

2.4.3 TCAN4550-Q1 Interrupt (nINT)

The nINT is a dedicated open drain global interrupt output pin. This pin needs an external pull-up resistor to V_{IO} to function properly. All interrupt requests are reflected by this pin when pulled low.

In test mode this pin is used as an EN pin input for testing the CAN transceiver and referenced as EN_INT. When this pin is high the device is in normal mode and when low it is in standby mode. This is accomplished by writing 0 to register 16'h0800[0].

NOTE: This pin is an active low and is the logical OR of all faults in registers 16'h0820 and 16'h0824 that are not masked.

Pin 31 of the LaunchPad connection header is connected to the nINT pin. Furthermore, since nINT is an interrupt status pin, an LED has also been added to this pin as a visual indicator to the user on the status of this pin and buffered through a transistor to prevent excessive loading on the device pin. However, since the nINT is a negative logic indicator, the signal is first inverted through an additional transistor such that the LED is illuminated when the nINT pin is low indicating a device interrupt has occurred.

2.4.4 TCAN4550-Q1 General Purpose Output (GPO2)

The GPO2 pin is an open drain configurable output function pin that provides selected interrupts. This pin needs an external pull-up resistor to V_{IO} to function properly. The output function can be changed by using register 16'h0800[23:22] and can be configured as a watchdog output reset pin.

In test mode this pin becomes the RXD_INT_PHY transceiver output or TXD_INT_CAN CAN controller output pin.

Pin 32 of the LaunchPad connection header is connected to the GPO2 pin. Furthermore, since GPO2 can be used as an interrupt status pin, an LED has also been added to this pin as a visual indicator to the user on the status of this pin and buffered through a transistor to prevent excessive loading on the device pin. However, since the GPO2 is a negative logic indicator when used for interrupts, the signal is first inverted through an additional transistor such that the LED is illuminated when the GPO2 pin is low indicating a device interrupt has occurred.

2.4.5 TCAN4550-Q1 General Purpose Input/Output (GPIO1)

This pin defaults out as the M_CAN_INT 1 (active low) interrupt. The functionality of the pin can be changed to a configurable output function pin by setting register 16'h0800[15:14] = 00. The GPO function is further configured by using register 16'h0800[11:10]. To configure the pin to support a watchdog input timer reset pin use SPI register 16'h0800[15:14] = 10.

When in test mode the GPIO1 pin is used to provide the input signal for the transceiver (TXD_INT_PHY) or the input to the M_CAN core (RXD_INT_CAN). This is accomplished by first putting the device into test mode using register 16'h0800[21] = 1 and then selecting which part of the device is to be tested by setting register 16'0800[0].

Pin 5 of the LaunchPad connection header is connected to the GPIO1 pin. Furthermore, since GPIO1 can be used as an interrupt status pin, an LED has also been added to this pin as a visual indicator to the user on the status of this pin and buffered through a transistor to prevent excessive loading on the device pin. However, since the GPIO1 is a negative logic indicator when used for interrupts, the signal is first inverted through an additional transistor such that the LED is illuminated when the GPIO1 pin is low indicating a device interrupt has occurred.

2.4.6 TCAN4550-Q1 Wake Request (nWKRQ)

The nKWRQ pin is a dedicated wake up request pin from a bus wake (WUP) request, local wake (LWU) request and power on (PWRON). The nWKRQ pin is defaulted to a wake enable based upon a wake event. In this configuration the output is pulled low and latched to serve as an enable for a regulator that does not use the INH pin to control voltage level. The nWKRQ pin can be configured by setting 16'h0800[8] = 1 as an interrupt pin that pulls the output low but once the wake interrupt flag is cleared releases the output back to a high. This pin defaults to an internal 3.6 V rail that is active during sleep mode. In this configuration, if a wake event takes place, the nWKRQ pin switches from high to low. This output can be configured to be powered from the V_{IO} rail through SPI programming, 16'h0800[19]. When powered off the V_{IO} pin the device does not insert an interrupt until the V_{IO} rail is stable. When configured for V_{IO} this pin is an open drain output and requires an external pull-up resistor V_{IO} rail. This configuration bit is saved for all modes of operation and is not reset in sleep mode. As some external regulators or power management chips may need a digital logic pin for a wake up request this can be used.

NOTE: This pin is active low and is the logical OR of CANINT, LWU and WKERR register 16'h0820 that are not masked.

If a pull-up resistor is placed on this pin it must be configured for power from the V_{IO} rail

2.4.7 TCAN4550-Q1 Reset (RST)

The TCAN4550-Q1 RST pin is a device reset pin. It has a weak internal pull down resistor for normal operation. If communication has stopped with the TCAN4550-Q1 the RST pin can be pulsed high and then back low for greater than t_{PULSE_WIDTH} to perform a power on reset to the device. This resets the device to the default settings and puts the device into standby mode. If the device was in normal or standby mode the INH and nWKRQ pins remain active (on) and do not toggle. If the device is in sleep mode and reset is toggled the device enters standby mode and at that time INH and nWKRQ turns on.

After a reset has taken place a wait time of ≥ 700 µS should be used before reading or writing to the TCAN4550-Q1.

2.4.8 TLIN2029-Q1 Transmit (TXD)

TXD is the interface to the MCU's LIN protocol controller or SCI and UART that is used to control the state of the LIN output. When TXD is low the LIN output is dominant (near ground). When TXD is high the LIN output is recessive (near $V_{BATTERY}$). The TXD input structure is compatible with microcontrollers with 3.3-V and 5-V I/O. TXD has an internal pull-down resistor. The LIN bus is protected from being stuck dominant through a system failure driving TXD low through the dominant state time-out timer.

2.4.9 TLIN2029-Q1 Receive (RXD)

RXD is the interface to the MCU's LIN protocol controller or SCI and UART, which reports the state of the LIN bus voltage. LIN recessive (near $V_{BATTERY}$) is represented by a high level on the RXD and LIN dominant (near ground) is represented by a low level on the RXD pin. The RXD output structure is an open-drain output stage. This allows the device to be used with 3.3 V and 5 V I/O microcontrollers. If the microcontoller's RXD pin does not have an integrated pullup, an external pullup resistor to the microcontroller I/O supply voltage is required. In standby mode the RXD pin is driven low to indicate a wake up request from the LIN bus.

2.4.10 TLIN2029-Q1 Enable (EN)

EN controls the operational modes of the device. When EN is high the device is in normal operating mode allowing a transmission path from TXD to LIN and from LIN to RXD. When EN is low the device is put into sleep mode and there are no transmission paths available. The device can enter normal mode only after wake up. EN has an internal pull-down resistor to ensure the device remains in low power mode even if EN floats.

3 Firmware

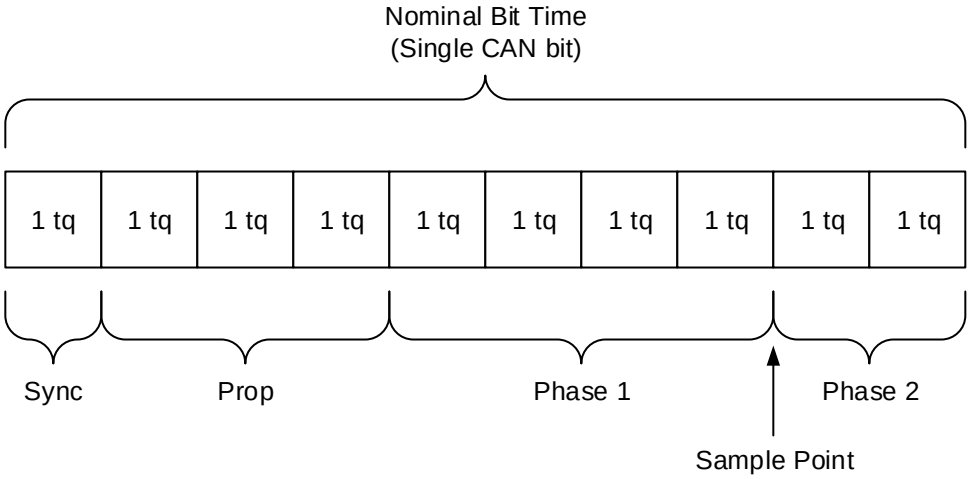
The TCAN4550-Q1 CAN FD controller with integrated transceiver SBC supports classic CAN and CAN FD communication through a SPI interface and incorporates the common Bosch_M_CAN controller. The provided firmware examples helps users configure their MCU for CAN or CAN FD communication without a deep understanding of the CAN/CAN FD protocol. A basic overview of the firmware features is included in this document, but a more detailed explanation can be found in the TCAN45xx software user's guide.

3.1 CAN / CAN FD Controller Configuration

The CAN controller is a state machine that manages protocol-specific details leaving the microcontroller to focus on managing the data being transmitted and received. These details must be setup during initialization of the TCAN4550-Q1 following every power cycle and reset event. Several of these settings are critical to proper communication and varies in value based on factors such as data rate, crystal oscillator or clock frequency, and physical bus characteristics that may lead to adjustments in the proper sample point within the bit period to avoid noise and overshoot ringing that needs to settle out before sampling the bit and avoid errors.

3.1.1 Bit Timing Setup

Figure 7. Nominal bit time



There are 4 sections to a nominal bit time in the CAN protocol: the synchronization segment, the propagation delay time segment, and the phase buffer segments 1 and 2. The sync_seg is used to synchronize the nodes in the network. The prop_seg is used to compensate for any physical delay times in the network and must be long enough to compensate for the delays between the sender and receiver and back to the receiver again in order to allow for successful arbitration. The phase_seg1 and phase_seg2 are used to compensate for signal edge phase errors and is adjusted based upon the sync_seg to adjust the bit's sample point which occurs between the phase_seg1 and phase_seg2.

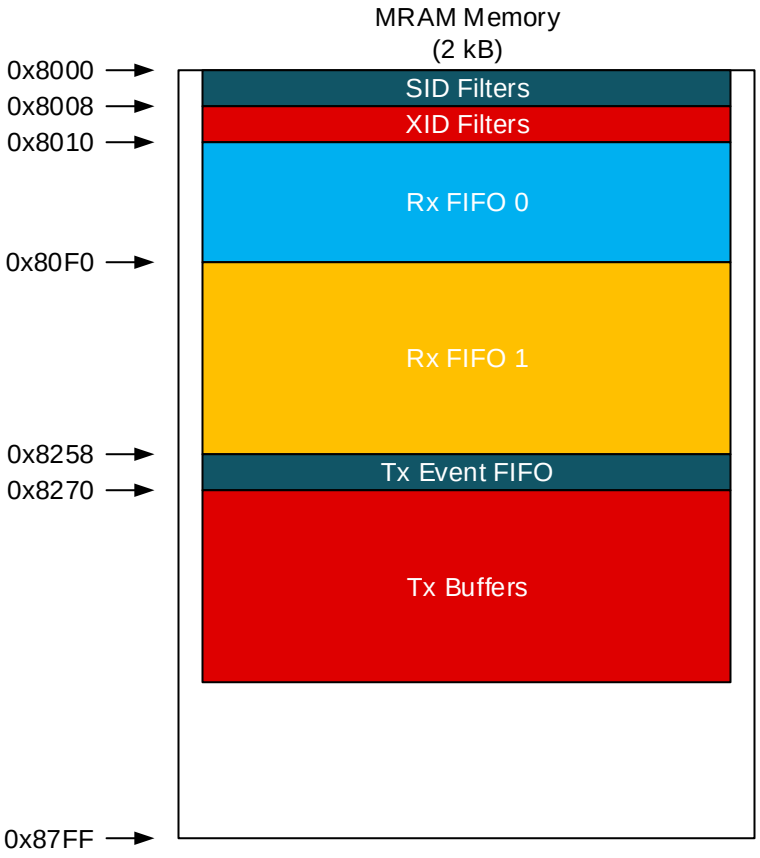
Each CAN bit is over-sampled by the controller and the interval between these samples is called a time quanta (tq). The four segments of the bit are defined in units of tq and these values must be programmed into the controller. The crystal or clock input frequency determines the minimum tq (1/frequency). The BoosterPack uses a 40-MHz crystal as its clock source and can have a prescaler applied to adjust the sample clock frequency to achieve the desired tq for the data rate. For example, if the 1:4 prescaler was applied to the 40-MHz input clock, the sample clock would be 10 Mhz. If we wanted the data rate to be 1-Mbps, this would yield 10 tq per bit. The four segments of the bit could then be defined with 1 tq for the sync_seg, 3 tq for the prop_seg, 4 tq for prop_seg1, and 2 tq for prop_seg2. This would place the sample point at 80% of the bit period allowing enough time for the signal to propagate through the bus.

There must be an integer multiple of tq per bit period and therefore these settings must be adjusted for the desired data rate that is used on the bus as well as any accommodations needed for the propagation delay of the bus used. CAN applications only use one data rate for the arbitration and data portions of a CAN packet. CAN FD applications need to calculate and program the settings for both the arbitration data rate and the faster data payload data rate.

NOTE: No segment can be equal to 0 tq, yet this is a valid value to be input to the TCAN4550-Q1 registers for the CAN bit timing. Therefore the TCAN4550-Q1 interprets the values input to the registers as 1 greater than the input value. For example, if the raw value for the timing parameter is 4, the register value should be set with a value of 3.

3.1.2 Message RAM Setup

Figure 8. Visual representation of MRAM allocation



The TCAN4550-Q1 contains a 2-kB block of memory used for sending and receiving CAN messages called the Message RAM (MRAM). This memory must be allocated by the system designer to define non-overlapping memory sections that holds the various message related elements. The TCAN4550-Q1 does not perform any check on the MRAM layout to ensure a valid configuration free of overlapping sections. It is critical for the system designer to verify the MRAM configuration is correct in order to avoid unanticipated behavior.

The MRAM can be configured into 7 sections, all of which are optional and without any order or sequencing restrictions.

There are two sections that can be used to define the 11-bit and 29-bit ID Filter elements. These are used to filter any incoming CAN messages as relevant, or non-relevant, for this node.

There are two sections that can be used as Receive FIFOs (RX FIFO 0 and RX FIFO 1). These store the incoming CAN messages but are not the only method of handling received messages.

RX Buffers can also be setup as another section that holds specific CAN messages, such as those containing specific message identifiers. These are not FIFOs and if new data is received into the buffer before the old data is read, the previous data is lost. RX Buffers and RX FIFOs can be setup and used at the same time if desired.

A Transmit Event FIFO section can be created to store message transmit event messages. These elements are generated by the TCAN4550-Q1 when transmitting a message are are for the microcontroller to read and know the status of a sent message.

TX Buffers are the final section that can be setup that holds the messages to be transmitted. When a message is to be sent, the data must first be loaded into the buffer before the transmission request can be made to the TCAN4550-Q1 by the microcontroller to start the transmission. The controller then tries to send the messages on the bus at the first available opportunity given the arbitration rules and message identifier priority level as compared with the other messages being transmitted from other nodes at that moment.

3.2 Sending and Receiving CAN Messages

In order to transmit a message with the TCAN4550-Q1, the following should be complete:

1. Ensure that the TCAN4550-Q1 is in standby mode (register 0x0800[7:6] = 0'b01). This forces M_CAN into INIT mode.
2. Set the M_CAN CCR register to allow for configuration. Set CCE and INIT bits if not already set. NOTE: The CSR bit reads back a 1 when in standby mode, but the user MUST write a 0 to this bit when doing a read-modify-write; otherwise, CAN communication fails.
3. If CAN FD and Bit Rate Switching (BRS) support is desired, it must be globally enabled via the FDF and BRS bits in the CCR register during configuration. See the device datasheet for more information about this register.
4. Any desired device features should be configured such as the watchdog timer, etc.
5. CAN timing information must be set.
6. The MRAM sections should be configured and initialized with any data.
7. Put the TCAN4550-Q1 device into "normal" mode (register 0x0800[7:6] = 0'b10) to turn on the transceiver and enable the CAN core for transmission.

Once these steps are complete, the microcontroller is able to transmit a message by writing to the TX Buffer and then requesting a message be sent by writing to the TXBAR register.

3.2.1 Sending a message

To send a message, it must first be written into the TX Buffer. A more detailed example of this process can be found in the section 4.3.1 of the TCAN45xx Software User's Guide.

1. Check the TX FIFO/Queue Status register (TXFQS: 0x10C4) bits [5:0] to make sure the free level is greater than 0 (meaning that at least 1 buffer is available) and that TFQF bit is set to 0.
2. Read TXFQS.TFQPI to get which index the message should be loaded into.
3. Calculate the memory offset to determine the start address in the MRAM memory.

4. Write the CAN message to the MRAM memory.
5. Send the transmission request to the TCAN4550-Q1.

Once the above process is completed, the TCAN4550-Q1 starts the transmission process and an entry is added to the event FIFO that can be read by the microcontroller.

The example firmware sends a CAN message every time the S1 switch is pressed on the MSP-EXP430FR6989 LaunchPad. The contents of this message are always the same and can be modified by changing the code to customize it for your application. Pressing a push-button switch was an easy event to use for an example of how to send a CAN message. Simply replace this code with your own code that causes the TCAN4550-Q1 to send a CAN message as desired in order to customize it for your application.

3.2.2 Receiving a message

The process to read a received message may differ slightly based upon how the TCAN4550-Q1 was configured. A more detailed example of this process can be found in section 4.3.2 of the TCAN45xx Software User's Guide. A basic process is as follows:

1. Determine where the new message is (RX FIFO 0, RX FIFO 1, or RX Buffer).
2. Based upon the buffer location of the new message, determine the buffer index and then the start address to read from MRAM.
3. Read the MRAM to retrieve the message.
4. Acknowledge the new message is read to release the FIFO element and make it available for a new message.

The process varies slightly depending on if the new message is in a RX Buffer or a RX FIFO; since they are fundamentally different. The FIFOs require the microcontroller to read a FIFO status register which tells the microcontroller how many new messages are in the FIFO, and what index to start reading at. The buffer requires the microcontroller to read the New Message Register, which tells the microcontroller which buffers have unread messages in them. At the end of each read, the microcontroller must let the TCAN4550-Q1 know that the message has been received in order to release the element for reuse.

In the example firmware, the message contents can be seen through the use of the debugger by setting a breakpoint on the line of code following the line where the message is read from the TCAN4550-Q1. The following line of code is the line that reads message from the TCAN4550-Q1.

numBytes = TCAN4x5x_MCAN_ReadNextFIFO(RXFIFO0, &MsgHeader, dataPayload); // This reads the next element in the RX FIFO 0

Place the breakpoint on the next line of code:

if (MsgHeader.ID == 0x0AA) // Example of how you can do an action based off a received address

From the Debugger's Variables window, you can view the message contents by expanding the MsgHeader variable by clicking on the arrow. Once you are comfortable with receiving this message data, you can simply add your own code to this if statement and customize it for your application.

3.3 Performance Optimization

There are several issues that can have a negative impact on the CAN throughput performance. The microcontroller firmware should be designed to minimize these effects in order to maximize performance.

The most common issue that hurts CAN throughput is poor SPI throughput. Excessive idle time between the chip select going low and the start of the data, or the time between the end of the data and the chip select going high, and the idle time between the MCU's SPI words can all contribute to a lot of idle time and increase the amount of time required between CAN messages being transmitted on the bus. Structuring the code to minimize these idle times ensures that maximum possible CAN throughput is possible.

Reducing the amount of SPI overhead alos ensures the maximum CAN throughput can be achieved. Every SPI transaction requires a single word (4 byte) header. Doing a single word data transfer is inefficient because 50% of the SPI bits are register data. Using Burst Reads and Writes can reduce unnecessary header data and reduce the total number of SPI bytes transmitted. When possible, structure the SPI data into packets of up to 256 words with the starting address of where the data bytes should be written. Not only does this reduce excess headers, but it also reduces the idle time associated with the chip select cycling between SPI transactions.

It is common to receive multiple CAN messages in the time it takes a microcontroller to read a message and act upon it before retrieving new incoming messages. If multiple messages have been received, reading these messages into the microcontroller in a bulk group reduces the amount of SPI overhead and reduce excess time incurred by retrieving these messages individually. Furthermore, it helps ensure that the RX FIFOs and Buffers do not fill up and are always available to receive new incoming messages.

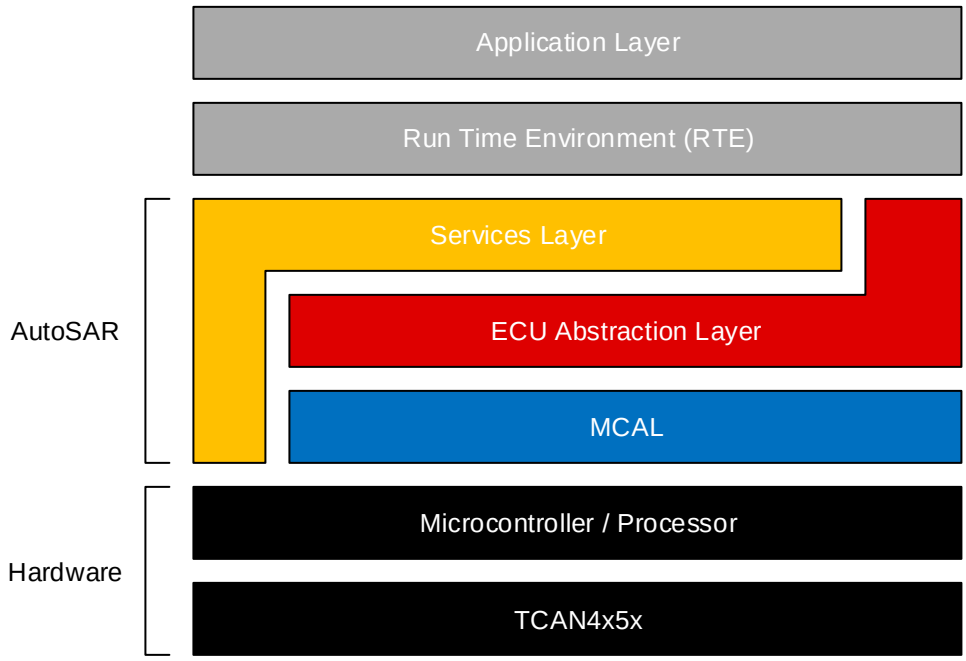
3.4 Microcontroller Abstraction

The BoosterPack was designed to support operation with any microcontroller launchpad with a SPI bus. Creating abstraction layers in the code simplifies the effort of porting the code to a different microcontrollers and allowing maximum amount of reuse at a system level.

AutoSAR (Automotive Open Systems Architecture) is a partnership of automotive-interested parties that have worked towards developing a specification for automotive systems. It provides specifications that describe the software modules which communicate with hardware and builds a common methodology of application development. The advantage of such specifications is that drivers for different hardware modules can be written in a way that are easily adaptable for many different processors or microcontrollers, since they are not built for a specific piece of hardware, but rather rely on lower-layer function calls which are defined by AutoSAR.

There are three main types of AutoSAR blocks that play into the system. The Microcontroller Abstraction Layer (MCAL) provides standardized function calls to peripherals of a microcontroller such as the IOs, bus ports such as SPI, and memory. The Electronic Control Unit (ECU) Abstraction Layer provides a common set of functions that an ECU would use such as CAN communication or GPIOs for sensing buttons or controlling lights. The MCAL and ECU modules are designed to be processor/vendor specific since they deal with the specific code required to perform a function. The last main layer is the Services Layer which provides background services to the application such as network services and bus communication services.

Figure 9. AutoSAR Abstraction Layers

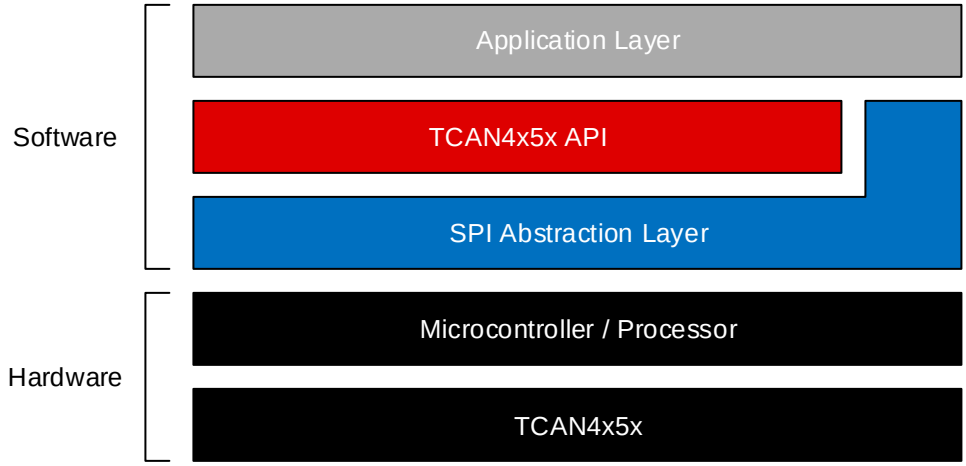


Texas Instruments provides the necessary ECU abstraction layer driver source code necessary to integrate the TCAN4550-Q1 into an AutoSAR environment. Contact TI to get the source code.

The demo firmware provided is a lower-level API for controlling various TCAN4550-Q1 functions and sending and receiving CAN messages. It does not have as significant of an overhead as AutoSAR, but it also has less abstraction. For non-AutoSAR applications this code library provides all of the building blocks to use as a starting point for any system development. This API abstracts TCAN4550-Q1 function calls to call upon a SPI Abstraction Layer function, allowing the user to quickly change code to a different microcontroller by only changing the code which controls the SPI peripheral. There are three main layers in this abstraction:

1. The SPI Abstraction Layer is responsible for handling the processor-specific SPI function calls to perform register reads and writes to the TCAN4550-Q1
2. The TCAN4550-Q1 API Layer provides a set of functions for performing TCAN4550-Q1 actions such as sending or receiving a CAN message.
3. The Application Layer is the end-user's code which calls upon the API to communicate with the TCAN4550-Q1 easily without much overhead

Figure 10. Microcontroller Abstraction



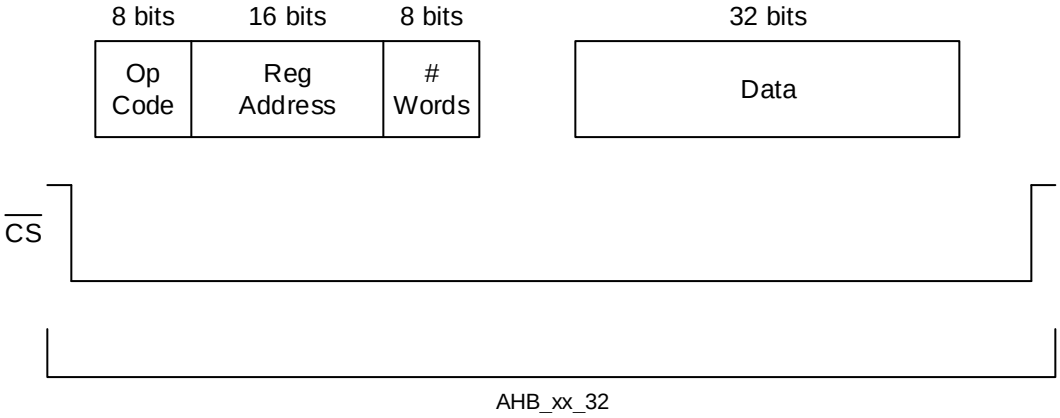
This Microcontroller Abstraction source code is developed for a MSP430FR6989 and is functional with the MSP-EXP430FR6989 LaunchPad. The code can be used with other microcontrollers and processors by updating the appropriate SPI drivers.

All of the code in the API software relies on the SPI Abstraction Layer. There are 8 functions that reside in this layer. The first 2 listed below are the fixed-length single-word read and write functions. The next 6 are multi-word read and write functions

1. uint32_t AHB_READ_32 (uin16_t address) : Single-register 32-bit word read.
2. void AHB_WRITE_32 (uint16_t address, uint8_t words) : Single-register 32-bit word write.
3. void AHB_READ_BURST_START (uint16_t address, uint8_t words) : Send the SPI header for a multi-word read, providing the starting register address and how many words are read.
4. uint32_t AHB_READ_BURST_READ ( ) : Returns a 32-bit word of data that is read, without toggling the CS pin.
5. void AHB_READ_BURST_END ( ) : At the end of a multi-register read, this function ends a SPI transaction by pulling the CS pin high.
6. void AHB_WRITE_BURST_START (uint16_t address, uint8_t words) : Send the SPI header for a multi-word write, providing the starting register address and how many words are written.
7. void AHB_WRITE_BURST_WRITE (uint32_t data) : Writes a 32-bit word of data, without toggling the CS pin.
8. void AHB_WRITE_BURST_END ( ) : At the end of a multi-register write, this function ends a SPI transaction by pulling the CS pin high.

For the multi-register read and write functions, there are 3 individual function calls to perform the read or write. The start of the SPI transaction transmits the correct command code, register address, and number of words that is transmitted but does not start any data transfer. The READ/WRITE functions do the actual data transfer to make it easier to put into a loop to handle each word of reading and writing. The END function pulls the CS pin high to signal to the TCAN4550-Q1 that the SPI transaction is complete.

Figure 11. 32-bit SPI read or write example



In Figure 11, a single 32-bit (1 word) SPI read or write example is shown. The first word contains the SPI header, which tells the TCAN4550-Q1 what action to perform (read or write), what register address to start at, and how many words of data to read/write.

Figure 12. SPI packet breakdown

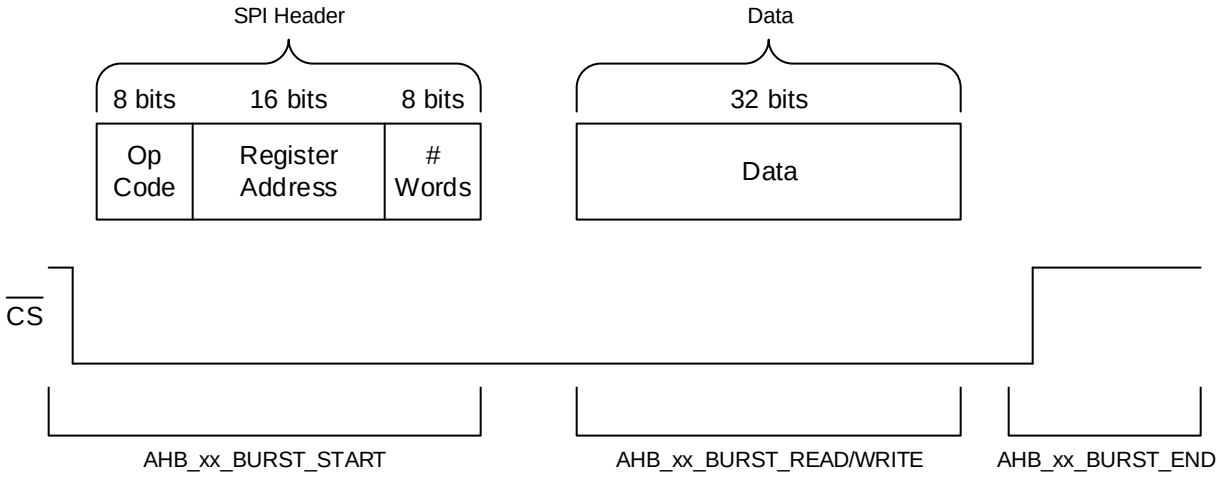


Figure 12 shows how a single word read or write can be broken up into 3 functions. The START function pulls CS low and sends the SPI header. The READ/WRITE function is responsible for reading or writing a single word at a time, for however many words were stated in the SPI header. The END function is responsible for pulling the CS high, to signal the end of the transfer.

Figure 13. Multi-word SPI packet example

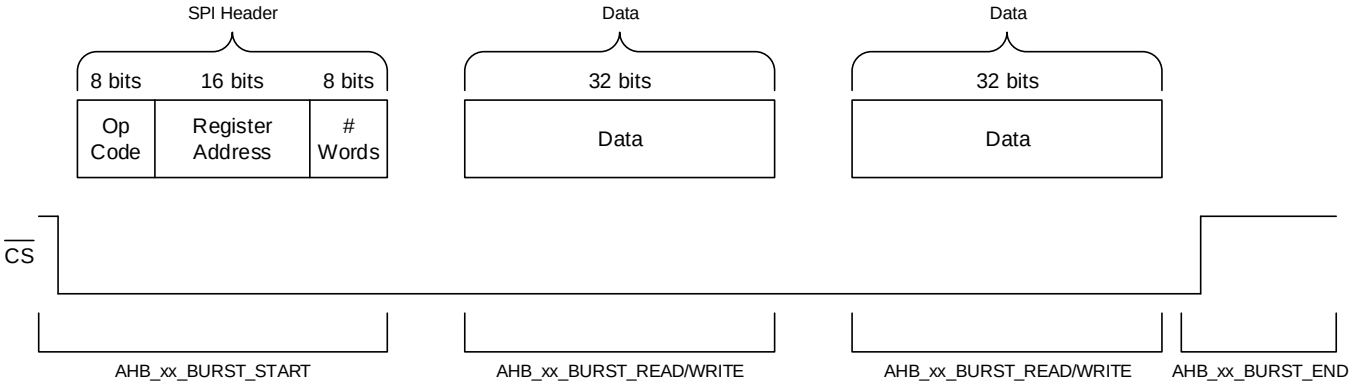


Figure 13 shows a 2 word SPI transfer example, and how the READ/WRITE function is called twice to do each individual word transfer. This is helpful for cutting down on the SPI overhead when transferring a CAN message to or from the TCAN4550-Q1, and minimize the maximum SPI frequency required.

4 Board Layout

Figure 14 and Figure 15 show the top and bottom of the BoosterPack.

Figure 14. BoosterPack Top

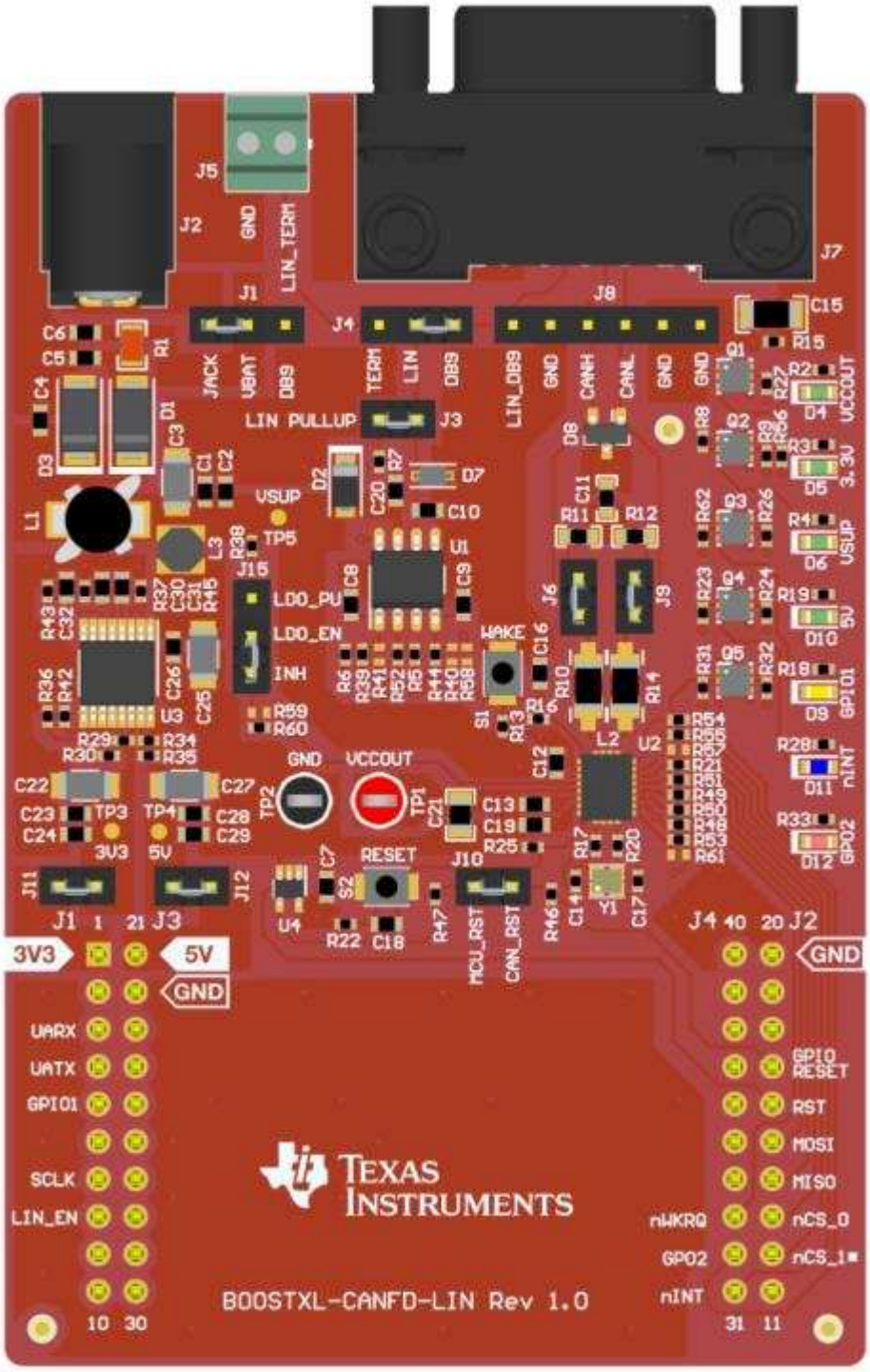
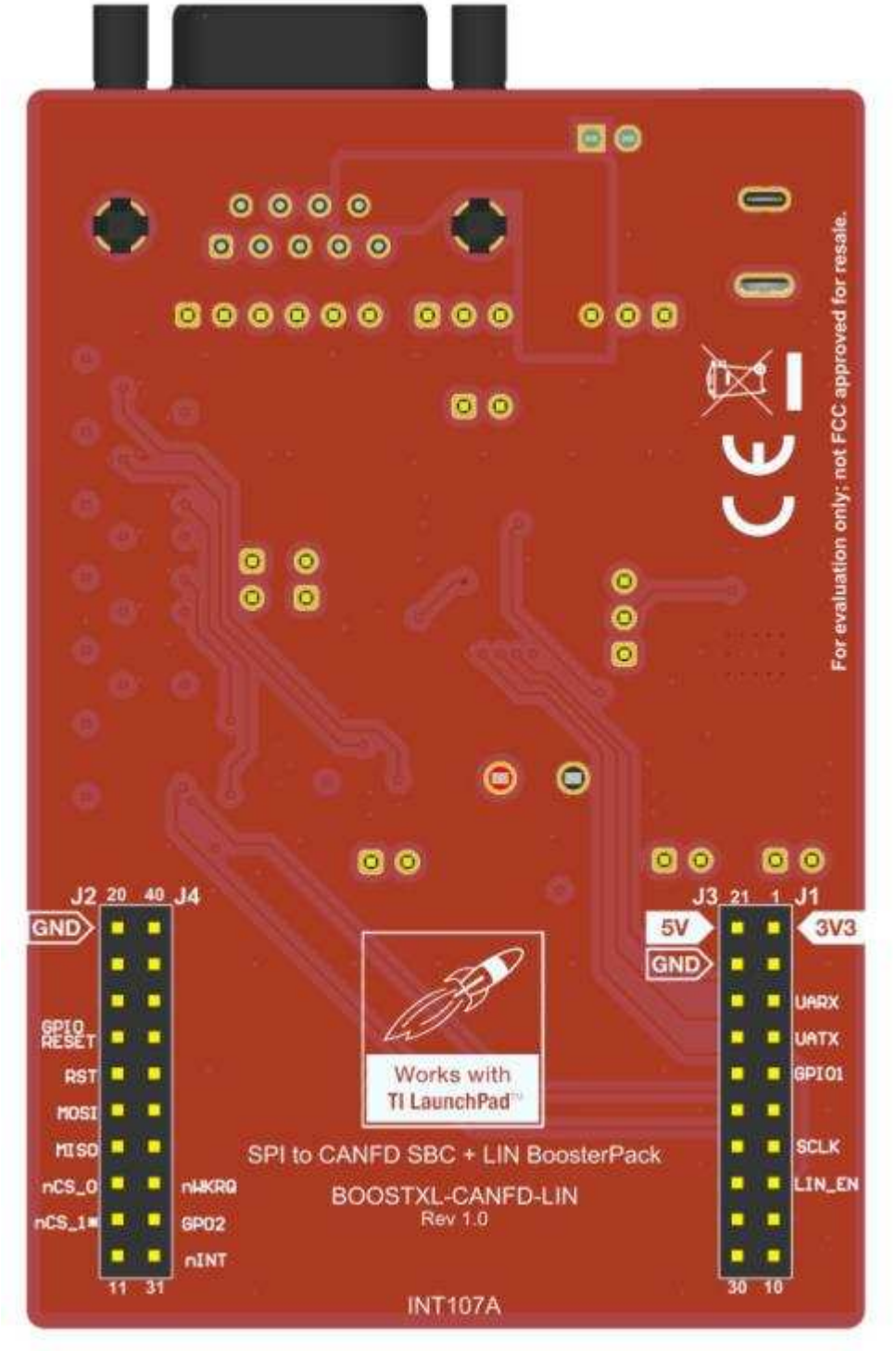


Figure 15. BoosterPack Bottom

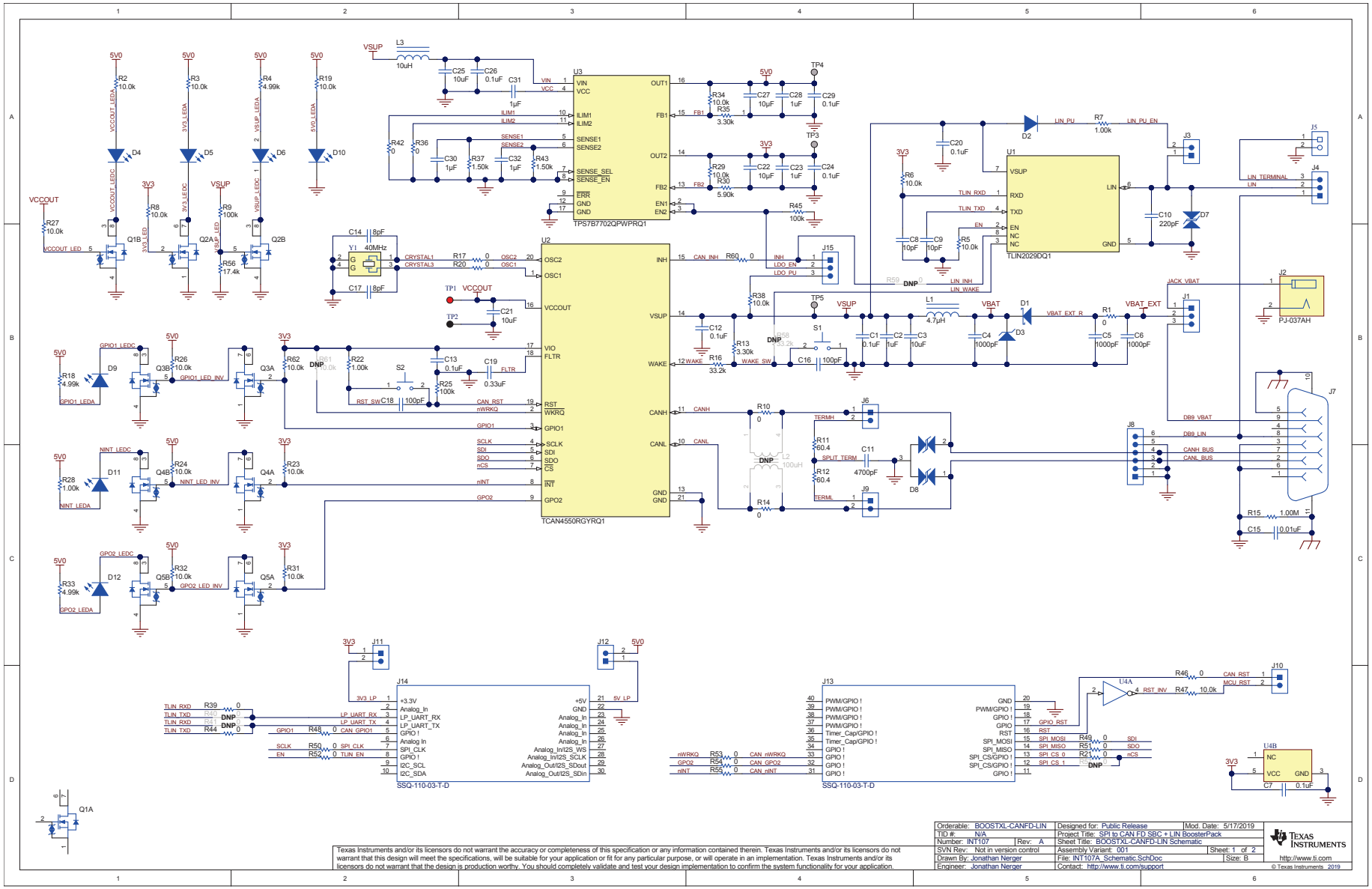


5 Schematic and Bill of Materials

5.1 Schematic

Figure 16 is a schematic diagram of the BoosterPack.

Figure 16. BoosterPack Schematic



5.2 Bill of Materials

Table 1 lists the bill of materials (BOM).

Table 1. Bill of Materials

Designator	Qty	Value	Description	Package Reference	Part Number	Manufacturer
!PCB1	1		Printed Circuit Board		INT107	Any
C1, C7, C12, C13, C20, C24, C26, C29	8	0.1uF	CAP, CERM, 0.1 uF, 50 V, +/- 10%, X7R, AEC-Q200 Grade 1, 0603	0603	CGA3E2X7R1H104K080AA	TDK
C2, C23, C28	3	1uF	CAP, CERM, 1 uF, 50 V, +/- 10%, X7R, 0603	0603	UMK107AB7105KA-T	Taiyo Yuden
C3, C25	2	10uF	CAP, CERM, 10 uF, 50 V, +/- 10%, X7R, 1206	1206	CL31B106KBHNNNE	Samsung
C4, C5, C6	3	1000pF	CAP, CERM, 1000 pF, 100 V, +/- 10%, X7R, AEC-Q200 Grade 1, 0603	0603	CGA3E2X7R2A102K080AA	TDK
C8, C9	2	10pF	CAP, CERM, 10 pF, 50 V, +/- 5%, C0G/NP0, 0603	0603	CGA3E2NP01H100D080AA	TDK
C10	1	220pF	CAP, CERM, 220 pF, 50 V, +/- 1%, C0G/NP0, 0603	0603	06035A221FAT2A	AVX
C11	1	4700pF	CAP, CERM, 4700 pF, 100 V, +/- 10%, X7R, 0603	0603	06031C472KAT2A	AVX
C14, C17	2	8pF	CAP, CERM, 8 pF, 50 V,+/- 6.25%, C0G/NP0, 0402	0402	CC0402DRNPO9BN8R0	Yageo
C15	1	0.01uF	CAP, CERM, 0.01 uF, 100 V, +/- 5%, X7R, 1206	1206	12061C103JAT2A	AVX
C16, C18	2	100pF	CAP, CERM, 100 pF, 50 V, +/- 5%, C0G/NP0, 0603	0603	06035A101JAT2A	AVX
C19	1	0.33uF	CAP, CERM, 0.33 uF, 16 V, +/- 10%, X7R, 0603	0603	C0603C334K4RACTU	Kemet
C21	1	10uF	CAP, CERM, 10 uF, 16 V, +/- 10%, X7R, 0805	0805	CL21B106KOQNNNE	Samsung Electro-Mechanics
C22, C27	2	10uF	CAP, CERM, 10 µF, 16 V,+/- 10%, X7R, 1206	1206	C1206C106K4RACTU	Kemet
C30, C31, C32	3	1uF	CAP, CERM, 1 µF, 16 V,+/- 10%, X7R, AEC-Q200 Grade 1, 0603	0603	GCM188R71C105KA64J	MuRata
D1	1	40V	Diode, Schottky, 40 V, 1 A, SMA	SMA	MBRA140T3G	ON Semiconductor
D2	1	100V	Diode, Switching, 100 V, 0.15 A, SOD-123	SOD-123	1N4148W-TP	Micro Commercial Components
D3	1	30V	Diode, Zener, 30 V, 3 W, SMA	SMA	3SMAJ5936B-TP	Micro Commercial Components
D4, D5, D6, D10	4	Green	LED, Green, SMD	LED_0603	150060GS75000	Wurth Elektronik
D7	1	15V	Diode, TVS, Bi, 15 V, SOD-323	SOD-323	PESD1LIN,115	NXP Semiconductor
D8	1	24V	Diode, TVS, Bi, 24 V, 70 Vc, AEC-Q101, SOT-23	SOT-23	PESD1CAN,215	NXP Semiconductor
D9	1	Yellow	LED, Yellow, SMD	LED_0603	150060YS75000	Wurth Elektronik
D11	1	Blue	LED, Blue, SMD	LED_0603	150060BS75000	Wurth Elektronik
D12	1	Red	LED, Red, SMD	LED_0603	150060RS75000	Wurth Elektronik
J1, J4, J15	3		Header, 100mil, 3x1, Gold, TH	3x1 Header	TSW-103-07-G-S	Samtec

Table 1. Bill of Materials (continued)

Designator	Qty	Value	Description	Package Reference	Part Number	Manufacturer
J2	1		Power Jack, 2mm, R/A, TH	Power Jack, R/A, TH	PJ-037AH	CUI Inc.
J3, J6, J9, J10, J11, J12	6		Header, 100mil, 2x1, Gold, TH	2x1 Header	TSW-102-07-G-S	Samtec
J5	1		Terminal Block, 2.54mm, 2x1, Brass, TH	Terminal Block, 2.54mm, 2-pole, Brass, TH	OSTVN02A150	On-Shore Technology
J7	1		D-Sub, 2.74mm, 9 Position, R/A, TH	D-Sub, 2.74mm, 9 Position, R/A, TH	5747840-5	TE Connectivity
J8	1		Header, 100mil, 6x1, Gold, TH	6x1 Header	TSW-106-07-G-S	Samtec
J13, J14	2		Receptacle, 2.54mm, 10x2, Tin, TH	10x2 Receptacle	SSQ-110-03-T-D	Samtec
L1	1	4.7uH	Inductor, Wirewound, Ferrite, 4.7 µH, 1.65 A, 0.08 ohm, AEC-Q200 Grade 0, SMD	6x6mm	B82462A4472M000	TDK
L3	1	10uH	Inductor, Shielded Drum Core, Ferrite, 10 uH, 1 A, 0.17 ohm, SMD	Inductor, 2.8x2.8x2.8mm	744025100	Wurth Elektronik
Q1, Q2, Q3, Q4, Q5	5	20V	MOSFET, 2-CH, N-CH, 20 V, 6.7 A, DQK0006B (WSON-6)	DQK0006B	CSD85301Q2	Texas Instruments
R1	1	0	RES, 0, 1%, 0.5 W, 0805	0805	5106	Keystone
R2, R3, R5, R6, R8, R19, R23, R24, R26, R27, R29, R31, R32, R34, R38, R47, R62	17	10.0k	RES, 10.0 k, 1%, 0.063 W, 0402	0402	RC0402FR-0710KL	Yageo America
R4, R18, R33	3	4.99k	RES, 4.99 k, 1%, 0.063 W, 0402	0402	RC0402FR-074K99L	Yageo America
R7	1	1.00k	RES, 1.00 k, 1%, 0.063 W, AEC-Q200 Grade 0, 0402	0402	CRCW04021K00FKED	Vishay-Dale
R9, R25, R45	3	100k	RES, 100 k, 1%, 0.063 W, 0402	0402	RC1005F104CS	Samsung Electro-Mechanics
R10, R14	2	0	RES, 0, 5%, 0.25 W, AEC-Q200 Grade 0, 1206	1206	ERJ-8GEY0R00V	Panasonic
R11, R12	2	60.4	RES, 60.4, 1%, 0.1 W, 0603	0603	RC0603FR-0760R4L	Yageo
R13, R35	2	3.30k	RES, 3.30 k, 1%, 0.1 W, AEC-Q200 Grade 0, 0402	0402	ERJ-2RKF3301X	Panasonic
R15	1	1.00Meg	RES, 1.00 M, 1%, 0.1 W, 0402	0402	ERJ-2RKF1004X	Panasonic
R16	1	33.2k	RES, 33.2 k, 1%, 0.063 W, AEC-Q200 Grade 0, 0402	0402	CRCW040233K2FKED	Vishay-Dale
R17, R20, R21, R36, R39, R42, R44, R46, R48, R49, R50, R51, R52, R53, R54, R55, R60	17	0	RES, 0, 5%, 0.063 W, 0402	0402	RC0402JR-070RL	Yageo America
R22, R28	2	1.00k	RES, 1.00 k, 1%, 0.0625 W, 0402	0402	RC0402FR-071KL	Yageo America
R30	1	5.90k	RES, 5.90 k, 1%, 0.063 W, AEC-Q200 Grade 0, 0402	0402	CRCW04025K90FKED	Vishay-Dale
R37, R43	2	1.50k	RES, 1.50 k, 1%, 0.063 W, AEC-Q200 Grade 0, 0402	0402	CRCW04021K50FKED	Vishay-Dale
R56	1	17.4k	RES, 17.4 k, 1%, 0.063 W, AEC-Q200 Grade 0, 0402	0402	CRCW040217K4FKED	Vishay-Dale

Table 1. Bill of Materials (continued)

Designator	Qty	Value	Description	Package Reference	Part Number	Manufacturer
S1, S2	2		SWITCH TACTILE SPST-NO 0.05A 12V	3x1.6x2.5mm	B3U-1000P	Omron Electronic Components
SH-J1, SH-J2, SH-J3, SH-J4, SH-J5, SH-J6, SH-J7, SH-J8, SH-J9	9		Shunt, 2.54mm, Gold, Black	Shunt, 2.54mm, Black	60900213421	Wurth Elektronik
TP1	1		Test Point, Multipurpose, Red, TH	Red Multipurpose Testpoint	5010	Keystone
TP2	1		Test Point, Multipurpose, Black, TH	Black Multipurpose Testpoint	5011	Keystone
U1	1		Fault Protected Local Interconnect Network (LIN) Transceiver with Dominant State Timeout, D0008A (SOIC-8)	D0008A	TLIN2029DQ1	Texas Instruments
U2	1		CAN-FD Controller with Integrated Transceiver, RGY0020A (VQFN-20)	RGY0020A	TCAN4550RGYRQ1	Texas Instruments
U3	1		Dual-Channel Antenna LDO With Current Sense, PWP0016J (TSSOP-16)	PWP0016J	TPS7B7702QPWPRQ1	Texas Instruments
U4	1		Low Power, 1.8/2.5/3.3-V Input, 3.3-V CMOS Output, Single Inverter Gate, DCK0005A, LARGE T&R	DCK0005A	SN74AUP1T04DCKR	Texas Instruments
Y1	1		Crystal, 40 MHz, 10 ppm, 8 pF, AEC-Q200 Grade 0, SMD	2.0x0.45x1.6mm	NX2016SA-40M-STD-CZS-3	NDK
FID1, FID2, FID3	0		Fiducial mark. There is nothing to buy or mount.	N/A	N/A	N/A
L2	0	100uH	Inductor, Ferrite, 100 uH, 0.15 A, 2 ohm, SMD	SMD, 4-Leads, Body 4.7 x 3.7 mm	ACT45B-101-2P-TL003	TDK
R40, R41, R57, R59	0	0	RES, 0, 5%, 0.063 W, 0402	0402	RC0402JR-070RL	Yageo America
R58	0	33.2k	RES, 33.2 k, 1%, 0.063 W, AEC-Q200 Grade 0, 0402	0402	CRCW040233K2FKED	Vishay-Dale
R61	0	10.0k	RES, 10.0 k, 1%, 0.063 W, 0402	0402	RC0402FR-0710KL	Yageo America

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265