LTC2970
Dual Power Supply Monitor Linduino Shield with LTC3604 and LT3581 Regulators

DESCRIPTION

The DC2467 Linduino® shield is a demonstration system for the LTC®2970 dual power supply monitor and margining controller. The LTC2970 provides a rich set of features for monitoring and controlling two power rails, including monitoring supply voltage and current, detecting faults in voltage and current, margining the supply voltage high or low, measuring temperature, and turning the supplies on and off through GPIO pins. It is fully accessible through the built-in I^{2}C bus, which allows it to communicate with an external microcontroller or other I^{2}C bus master.

The DC2467 Linduino shield operates in conjunction with a Linduino One (DC2026) and with a DC1613 controller. The Linduino is a microcontroller that comes with a full library of existing C code examples, as well as the capability to run custom C code to control the LTC2970. The DC1613 enables a PC running LTpowerPlay® to access the LTC2970 on the DC2467. Linduino and LTpowerPlay cooperate to enable code development and debugging in one convenient system.

On the DC2467 board, two DC/DC switching regulators provide +5V and –5V power outputs and demonstrate how the LTC2970 monitors and controls power supplies in a system. The LTC2970 fully controls both regulators. The LTC3604 is a buck regulator with integrated switches, configured to step down the +12V input to +5V at 1A. The LT3581 is a multifunction switching regulator, configured here as an inverting DC/DC converter producing –5V at –1A.

DC2467 Features

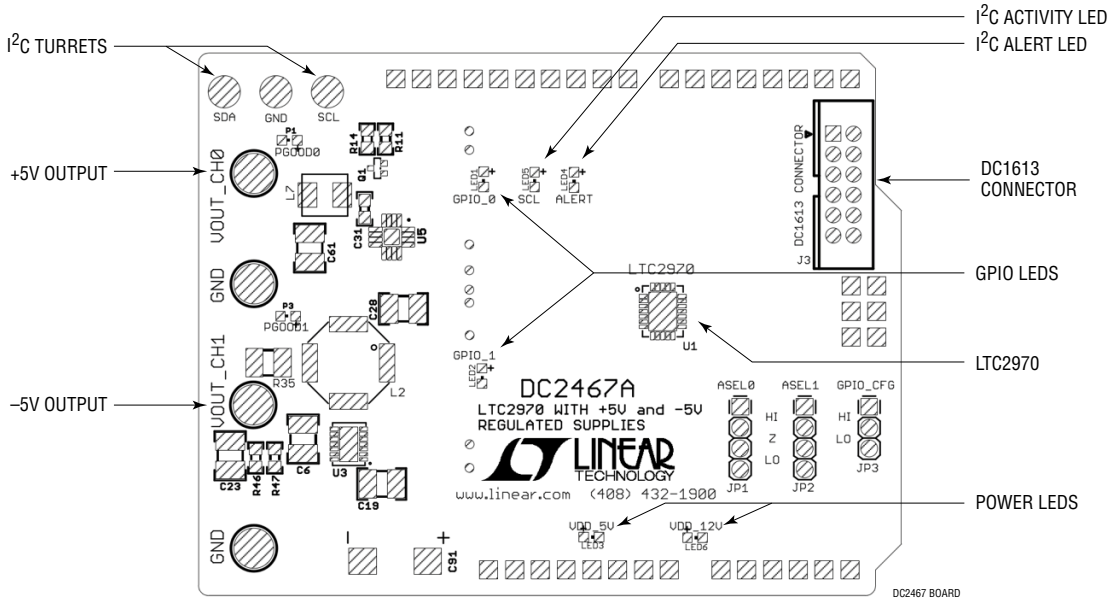
- Demonstrate the Capabilities of the LTC2970
- Two Onboard Power Supplies: +5V and –5V
- Monitor Supply Voltage and Current
- Adjust Supply Voltage Higher or Lower on Command
- Monitor Voltage and Current Faults with Adjustable Thresholds
- Monitor LTC2970 Die Temperature
- Connect to both Linduino and LTpowerPlay
- Demonstrate C Code Available for the LTC2970

DC2467 Hardware Required

- DC2467 Shield
- Linduino One (DC2026)
- USB-to-I^{2}C/SMBus/PMBus Controller (DC1613)
- +12V Power Supply
- 2 USB Cables

Design files for this circuit board are available at http://www.linear.com/demo/DC2467A

DESCRIPTION



PERFORMANCE SUMMARY

Table 1. LTC2970 Performance Summary

PARAMETER	CONDITIONS	VALUE
12VIN Supply Input Voltage Range		8V to 15V
VDD Regulator Voltage Range		4.75V to 5.25V
ADC Total Unadjusted Error	V_{IN} = 3V	±0.5%
ADC Input Range		0V to 6V
ADC Resolution	Resolution = 8.192V/16384	500µV/LSB
ADC Conversion Time		33.3ms
Margining DAC Resolution		8-bits
Temperature Sensor Resolution		0.25°C/LSB
I2C Serial Clock Frequency		10kHz to 400kHz

Table 2. DC2467 Performance Summary

PARAMETER	CONDITIONS	VALUE
VOUT_CH0	Untrimmed	+5V ±50mV
IOUT_CH0 (Max)	T = 25°C	1A
VOUT_CH0 Servo Range		±0.8V
VOUT_CH1	Untrimmed	–5V ±50mV
IOUT_CH1 (Max)	T = 25°C	–1A
VOUT_CH1 Servo Range		±0.75V

LINDUINO QUICK START

Make sure that the Linduino software is properly installed on your PC, then get started quickly developing code for the LTC2970 with these easy steps.

1. Follow the Linduino installation instructions in the DC2026 demo manual.

   Follow the software installation and IDE configuration instructions. These can be found on the Linduino main web page and include instructions for downloading and installing the Arduino software and downloading and installing the LTSketchbook. Both of these steps must be completed successfully before proceeding further (see Note 1 at the end of this document). Begin with this web page:

   http://www.linear.com/solutions/linduino

   This document assumes software running on a Windows PC. LTpowerPlay only runs on Microsoft Windows. Linux and Mac versions of the Arduino software are available. While these usually work well with the Linduino, they may have their own platform-specific details and are not specifically addressed in this document.

   LTSketchbook is the complete code base for the Linear Technology devices supported by the Linduino platform. It includes library code, demonstration sketches and HTML documentation. Download the latest LTSketchbook.zip from:

   http://www.linear.com/docs/43958

   The LTSketchbook contains code for many Linear Technology® ICs and demo boards, so it is important to upload the specific DC2467 sketch to the Linduino before using it. Below, we show how to do this (see Note 5).

2. Plug in the Linduino

   Attach the DC2467 to the DC2026 shield connector. The DC2467 shield will plug into the top of the DC2026, with corresponding pins lined up along both edges. Figure 1 shows the proper arrangement.

   Set the DC2467 jumpers as follows:

JUMPER	SETTING
ASEL0	L0
ASEL1	L0
GPIO_CFG	L0

   Plug in the USB cable to the DC2026 and to the PC. The USB connector powers the DC2026 and the LTC2970. The DC2026 communicates with the PC through the USB connection.

   Plug in the +12V wall adapter power source. This is necessary to supply power to the two onboard DC/DC converters on the DC2467. +12V is not necessary to power the LTC2970 and initiate I²C communication. The voltage should remain between 8V and 15V.

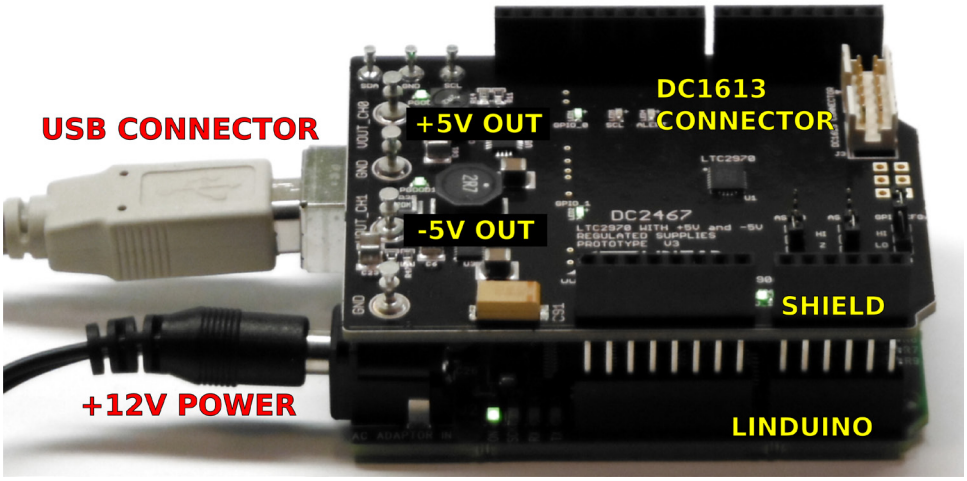


Figure 1. DC2467 Shield Mounted on the DC2026 Linduino

LINDUINO QUICK START

3. Upload the Linduino sketch

   The Linduino has a microcontroller with nonvolatile flash program memory. It stores instructions that execute upon power-up or reset. The contents of the Linduino flash memory should be updated to talk to the LTC2970. The LTSketchbook contains a demonstration sketch for the DC2467.

   Begin by opening the Arduino software. Select the DC2467 sketch in the Arduino File menu (Figure 2):

   File→Sketchbook→Part_Number→2000→2900→ 2970→ DC2467

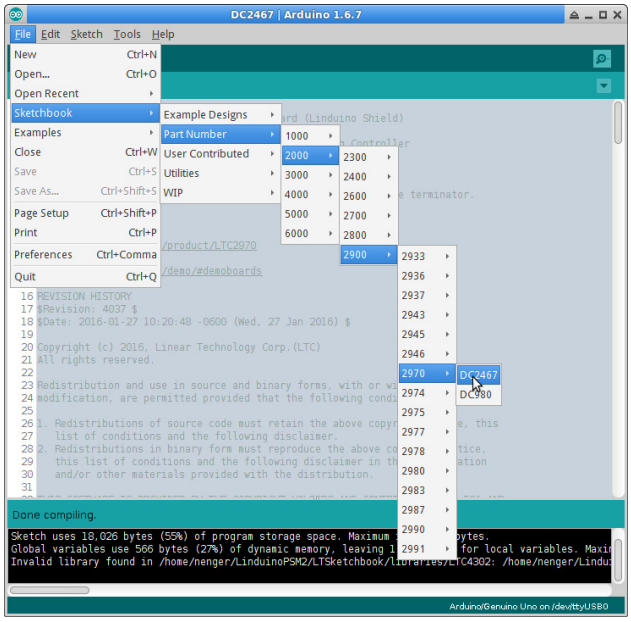


Figure 2. Open the DC2467 Sketch from the Sketchbook Menu

   Ensure that the Arduino software is using the correct COM port (see Note 3 at the end of this document). Microsoft Windows communicates with the Linduino through a serial port (COM port). You must tell the Arduino software which COM port to use.

   Tools→Port→COMxx

   Note that every time you plug in the Linduino USB connector your PC may select a different COM port for it. Often the COM port number will be 5 or higher.

   Press the arrow button at the top of the window to compile and upload the sketch (Figure 3).

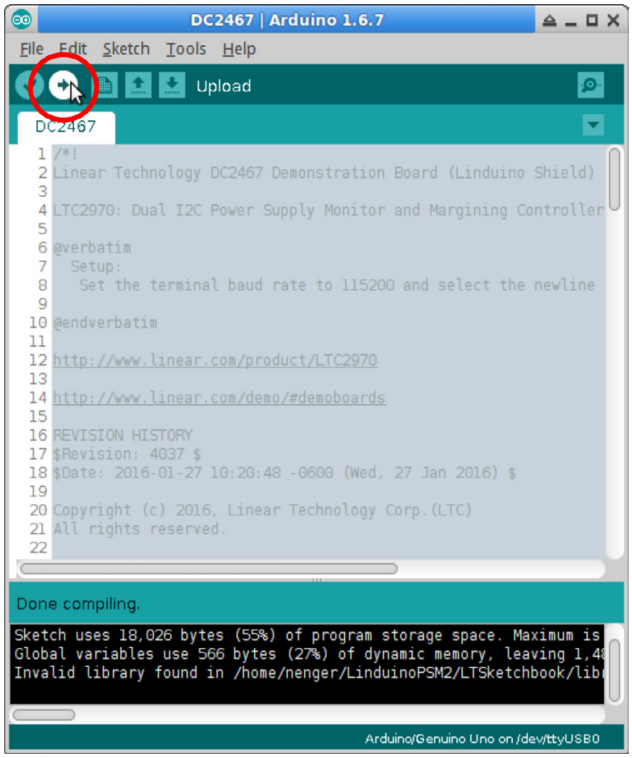


Figure 3. Press the Compile and Upload Button

LINDUINO QUICK START

4. Execute Linduino menu commands

   Now the Linduino is running the desired code and ready to command the LTC2970. The primary user interface is the serial port, which sends and receives simple ASCII characters to the PC and can act on simple menu-driven inputs and return basic responses (Figure 5).

   Open the serial terminal by pressing the "Serial Monitor" button at the top of the window (Figure 4). The serial monitor opens in a new window.

Figure 4. Press the Serial Monitor Button

When the Arduino software is configured correctly, the Linduino prints a menu in the serial monitor window (Figure 5). Select one of the menu actions by typing its number in the command line at the top of the window and pressing ENTER.

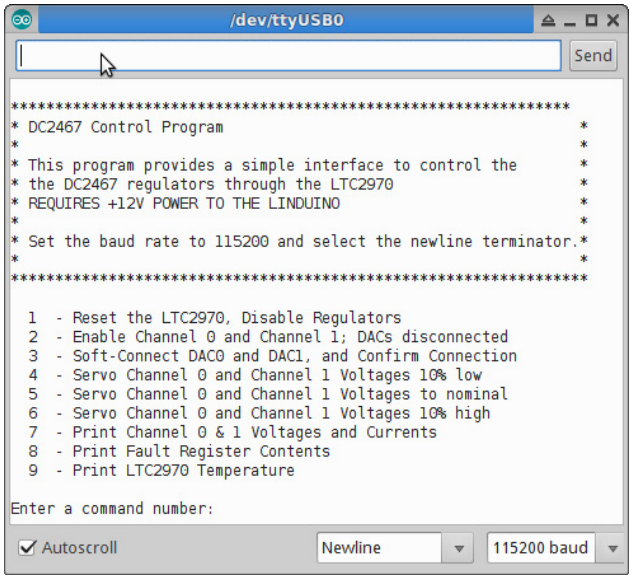


Figure 5. Serial Monitor Showing the Menu from the Linduino and the DC2467

The DC2467 Linduino sketch includes options for exercising many of the features of the LTC2970. Each option invokes a C-code procedure. When you select a menu option the Linduino executes the code, sending a series of commands over the I^2C bus, then it stops, freeing the bus for other traffic. This is important because it allows other I^2C bus masters to communicate with the LTC2970.

LTpowerPlay GUI SOFTWARE

LTpowerPlay is a powerful Windows-based graphical user interface (GUI) software tool that supports many Linear Technology products, including the LTC2970 dual power supply monitor and margining controller. LTpowerPlay communicates with the LTC2970 on the DC2467 through the DC1613 dongle connected to the 12-pin header on the DC2467 shield board.

LTpowerPlay gives access to all of the LTC2970 configuration and status registers. It allows the user to change configuration register settings and read back status and telemetry in real time. It can store and retrieve settings in project files on the PC. It can also produce executable lines of C code for the Linduino, encapsulating register read and write operations.

The LTpowerPlay software includes an automatic update feature to periodically check for updates and stay current with the latest device drivers and documentation. Download LTpowerPlay here:

http://www.linear.com/ltpowerplay

Access technical support documents for Linear Technology products from within the tool by clicking on the Help→View Online Help menu.

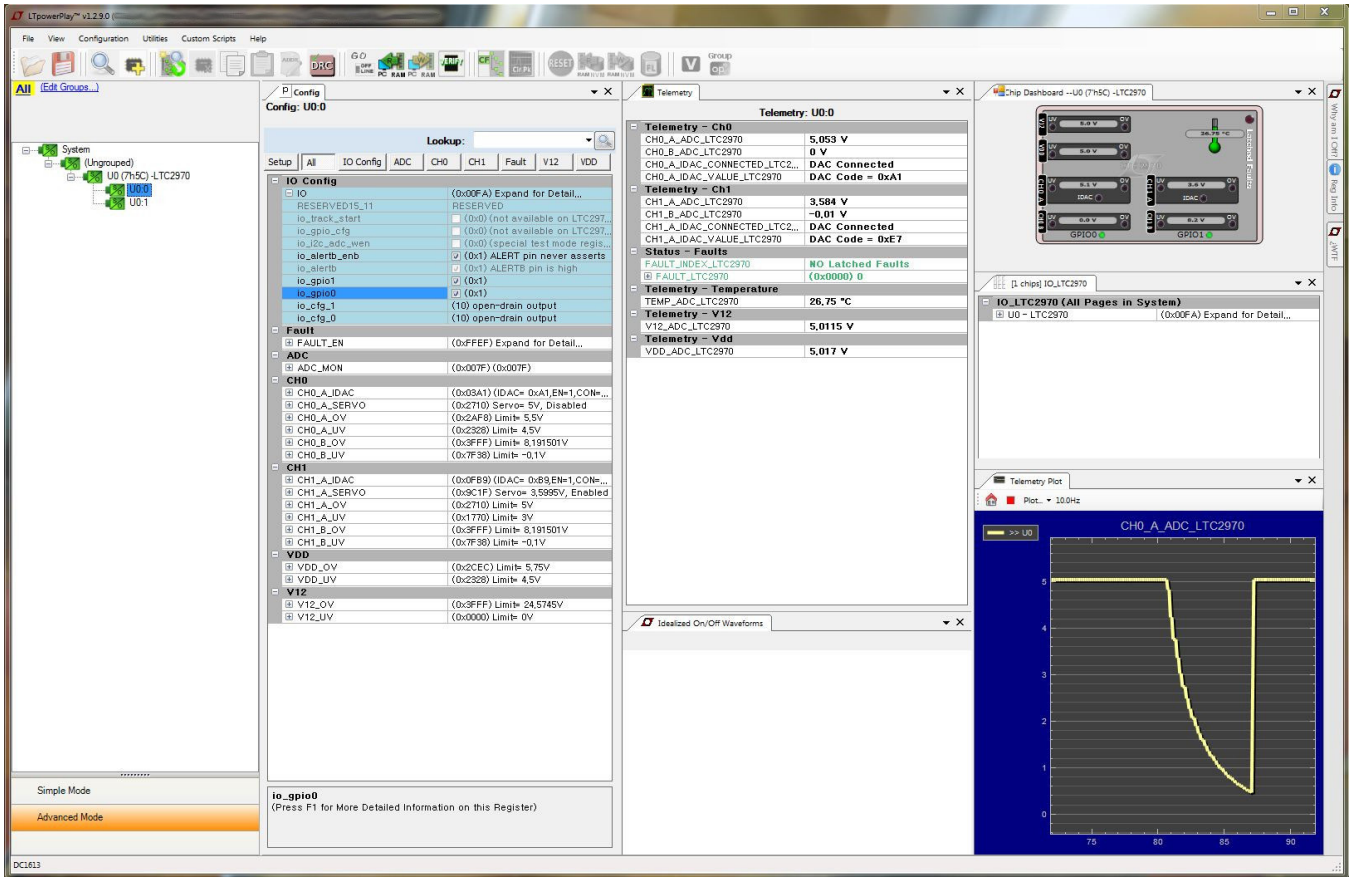


Figure 6. LTpowerPlay GUI Window

USING LTpowerPlay WITH DC2467

Figure 7. DC2467 with DC1613 Dongle Attached

The procedure for using LTpowerPlay with the DC2467 and Linduino is simple:

1. Plug in the Linduino USB, the DC1613 dongle and +12V power (Figure 7) (Note 6).
2. Load the DC2467 Linduino sketch (Figure 2 and Figure 3).
3. Start LTpowerPlay on your PC.

   Because there are multiple USB-to-serial devices connected (the Linduino and the DC1613), select the "LTC DC1613" in the list of devices.

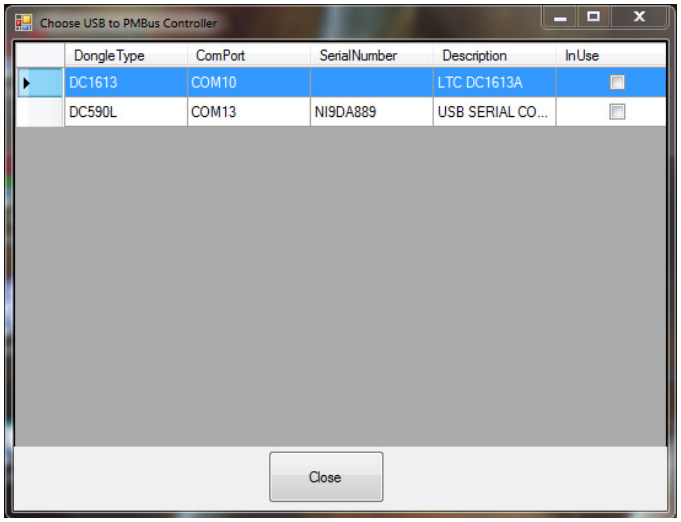
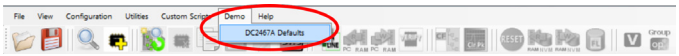


Figure 8. LTpowerPlay Device List

4. Load the DC2467 project into LTpowerPlay

   The demo menu is a 1-click option for loading good demo board defaults. You can also load your own file through the File menu.



5. Use LTpowerPlay to read and write registers in the LTC2970, view telemetry and save state to a project file.

   The DC1613 dongle becomes I^2C bus master while LTpowerPlay is sending and receiving commands, so it has the potential to conflict with the Linduino, which also becomes bus master when it is talking to the LTC2970. It is important that only one device talks on the I^2C bus at any given time, so we take precautions to only allow one bus master. The primary way to do this is to force LTpowerPlay offline when the Linduino is communicating and to refrain from selecting any Linduino menu commands while LTpowerPlay is online.

6. Go offline by pressing the button in LTpowerPlay:

7. Type a Linduino menu command number to execute the code (Figure 5).
8. Go online by pressing the button in LTpowerPlay:

   Notice that while LTpowerPlay is online, the green SCL LED on the DC2467 demo board flickers continually. This indicates that I^2C bus traffic is flowing from the DC1613 dongle to the LTC2970.

LTpowerPlay has many features that make it helpful, not only for visualizing what the LTC2970 is doing, but for understanding what the whole system is doing. One such feature is the scaling parameter list which allows some parameters to be linearly scaled and shifted to more meaningful values. This is especially useful in cases where the register voltage reading must be transformed in order to arrive at the actual measured quantity. This is the case

USING LTpowerPlay WITH DC2467

with current, which is measured as the voltage across a 0.02Ω resistor and negative regulator voltage, which is level-shifted and scaled by a gain factor.

Figure 9 shows the Setup tab in LTpowerPlay. The tool allows linear scaling and offset for CH0_A, CH0_B, CH1_A and CH1_B. Since the DC2467 uses CH0_B to measure current, we scale its value by 1/0.02 = 50. Since CH1_A measures a level-shifted negative voltage, we scale and offset the register reading to display the voltage at the turret on the demo board. These adjustments apply to the servo voltages and OV and UV limits as well.

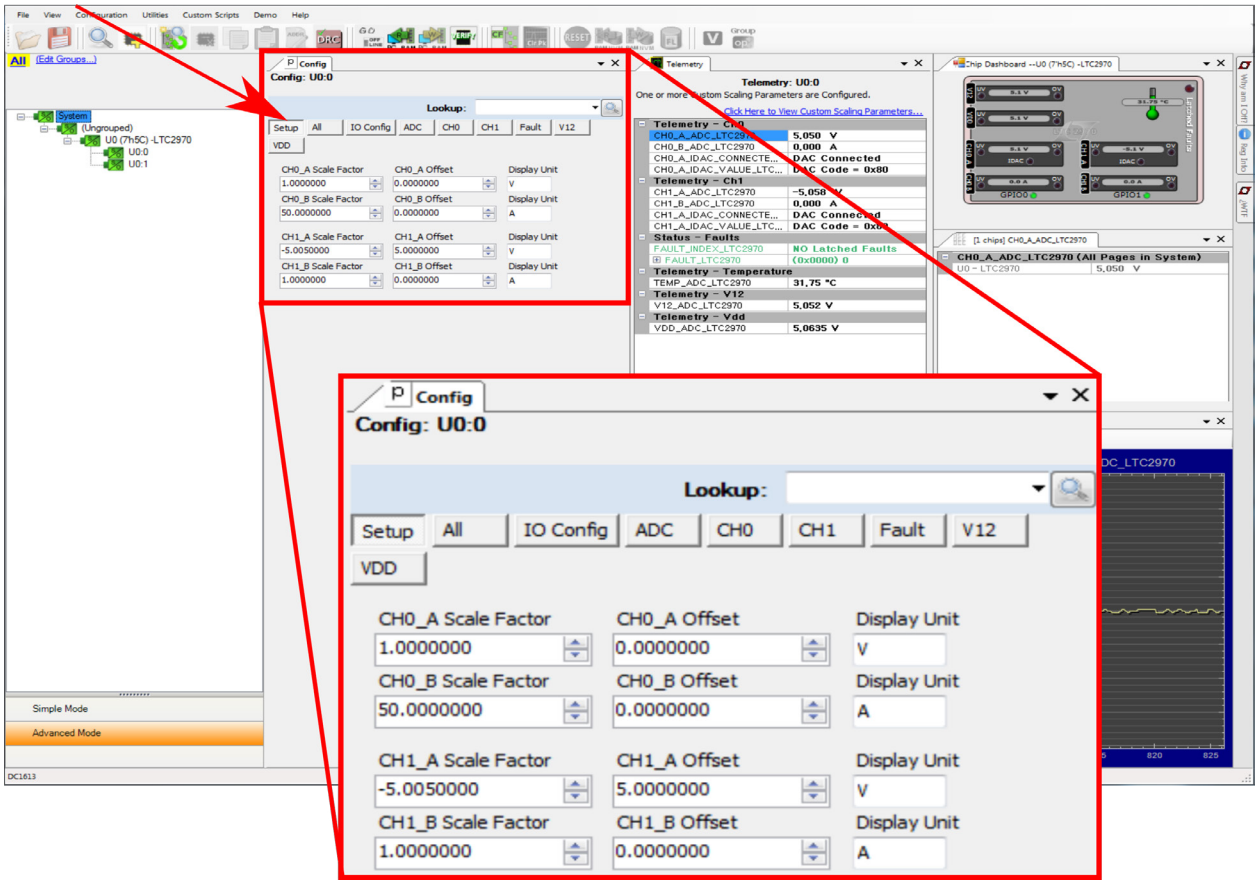


Figure 9. LTpowerPlay Setup Tab

WRITING AND DEBUGGING CODE

The LTSketchbook contains a large set of code for talking to many different Linear Technology ICs. These examples are an excellent starting place for code development. Many of the subtleties of using each particular part have been coded in the examples.

Each example Linduino sketch is located in its own directory in the LTSketchbook file tree. Each sketch is associated with the chip and demo board by the directory structure. Find code for the LTC2970 and the DC2467 here:

…/LTSketchbook/Part Number/2000/2900/2970/DC2467/DC2467.ino

The DC2467.ino sketch calls a set of functions for controlling the LTC2970, such as configuring all of the registers, enabling and disabling the GPIOs, soft connecting the servo DACs to the regulators, margining the regulators high and low, and reading telemetry and fault registers. These functions are contained in a library, located here:

…/LTSketchbook/libraries/LTC2970/LTC2970.cpp

This makes it easy to program the LTC2970 because most of the functions that a user will need are coded and debugged. A typical user will use the Linduino code library directly and only need to update how it accesses the I^2C bus in their system.

Writing Your Own

The Arduino software requires each sketch to reside in a directory of the same name as the sketch file. A preexisting template where you can insert your own code is here:

…/LTSketchbook/Part_Number/2000/2900/2970/DC2467_TEMPLATE/ DC2467_TEMPLATE.ino

This is a framework for code experimentation. There are comments in the template indicating useful places to insert your own commands. Use your favorite code editor or the Arduino software to edit the file. Below is a stripped down example of a Linduino sketch, showing the structure of the code (see the full sketch file for important details):

```
static uint8_t ltc2970_i2c_address = 0x5B;
static LT_SMBusNoPec *smbus = new LT_SMBusNoPec();
uint16_t some_var;
void setup()
{
 // CODE IN THIS SECTION RUNS ONCE
 //   AT THE BEGINNING OF EXECUTION.
 // PLACE INITIALIZATION HERE
  Serial.begin(115200);
}
void loop()
{
  // CODE IN THIS SECTION RUNS IN A LOOP FOREVER
  // USUALLY THIS IS A MENU-DRIVEN LIST OF OPTIONS
  //   COMMUNICATED OVER THE SERIAL PORT
  // THE USER CAN PLACE CODE INTO THE "SWITCH"
  //   STATEMENT BELOW
  uint8_t user_command;
  user_command = read_int();
  switch (user_command){
   case 1 :
    Serial.print(F("*INITIALIZE THE LTC2970*\n"));
    smbus→writeWord(ltc2970_i2c_address,
         LTC2970_FAULT_EN, 0x0DEF);
    some_var = smbus→readWord(ltc2970_i2c_address,
         LTC2970_IO);
   break;
   case 2 :
    // INSERT YOUR CODE HERE...
   break;
  }
}
```

WRITING AND DEBUGGING CODE

Notice several features of the sketch above. First, there is no "main" function. A sketch is not a complete C program. It contains an initialization routine, called setup(), and a "loop forever" routine, called loop(). These functions do not accept input parameters and do not provide return values. Communicating between the setup() and loop() routines must be done through global variables. When the Arduino program on the PC compiles the sketch it adds all of the necessary infrastructure to make a complete program.

The sketch accepts a user input through the read_int() function and the switch statement operates on the value returned when the user, using the Arduino serial monitor, types a key and hits ENTER. The switch statement can have any number of cases to handle all possible user inputs. This code is written so that the loop() completes once for every user input.

The sketch sends text back to the serial monitor through the Serial.print() function. The "Serial" object is a C++ construct that communicates over the PC's COM port. Serial.print is one of many methods of the Serial class.

The sketch sends I^2C commands over the smbus object, created as an instance of the LT_SMBusNoPec class. The writeWord() method sends a word over the bus to the LTC2970. The readWord() method reads a word from the bus. Because the LTC2970 is an SMBus part, we take advantage of the LT_SMBus library classes to simplify communication with the part.

These are just a few of the many possibilities. The user is encouraged to play with the DC2467_TEMPLATE.ino sketch to explore and learn. Notice that the sketch file contains more details that were omitted here for clarity. Use the code in the libraries to find examples of complex routines that control the various Linear Technology ICs.

Compile and upload your code to the Linduino with the arrow button shown in Figure 3.

You may write your own C code to execute the desired behaviors, or use LTpowerPlay, as outlined below.

LTpowerPlay and Code

LTpowerPlay has a feature that generates executable C code from register operations. This feature produces lines of code that correspond to individual register reads and writes performed in the tool. Each line of code can be pasted directly into the Linduino C code and executed.

To access this feature, click the "Reg Info" tab on the right-hand side of the LTpowerPlay GUI (Figure 10). At the top of the Reg Info window is a "Code Sequence Clipboard" area that contains a running list of actions in C code format. Each time the user updates a register using the F12 function key or reads register contents using the F11 function key, LTpowerPlay inserts a line of code in the Code Sequence Clipboard. Simply select a register in the Config list, then press the F11 or F12 function key. Cut and paste these lines of code directly into a Linduino sketch, such as DC2467_TEMPLATE.ino, to perform the same register reads and writes.

For example, click on the CH0_A_IDAC register in the Config section of the LTpowerPlay window. Change a setting, such as the value of the IdacDataValue. Press F12 on the keyboard to send this updated register value to the LTC2970. The resulting line of code in the Reg Info tab will look similar to the code in Figure 10. Similarly, pressing F11 on the keyboard will read from the register and produce a line of C code that reads the register and returns a result.

Note that the generated C code is executable, but it is only the first step in the process. There are several things to watch out for. A register read must return a value and the example code returns to a variable called "some_var". Because this is an example, the name is as good as any other. You can use it as is, or rename it in your code. Another subtlety is that the register names and I^2C addresses are defined in the LTC2970.h file by #define statements, which LTpowerPlay does not use (see Note 4).

WRITING AND DEBUGGING CODE

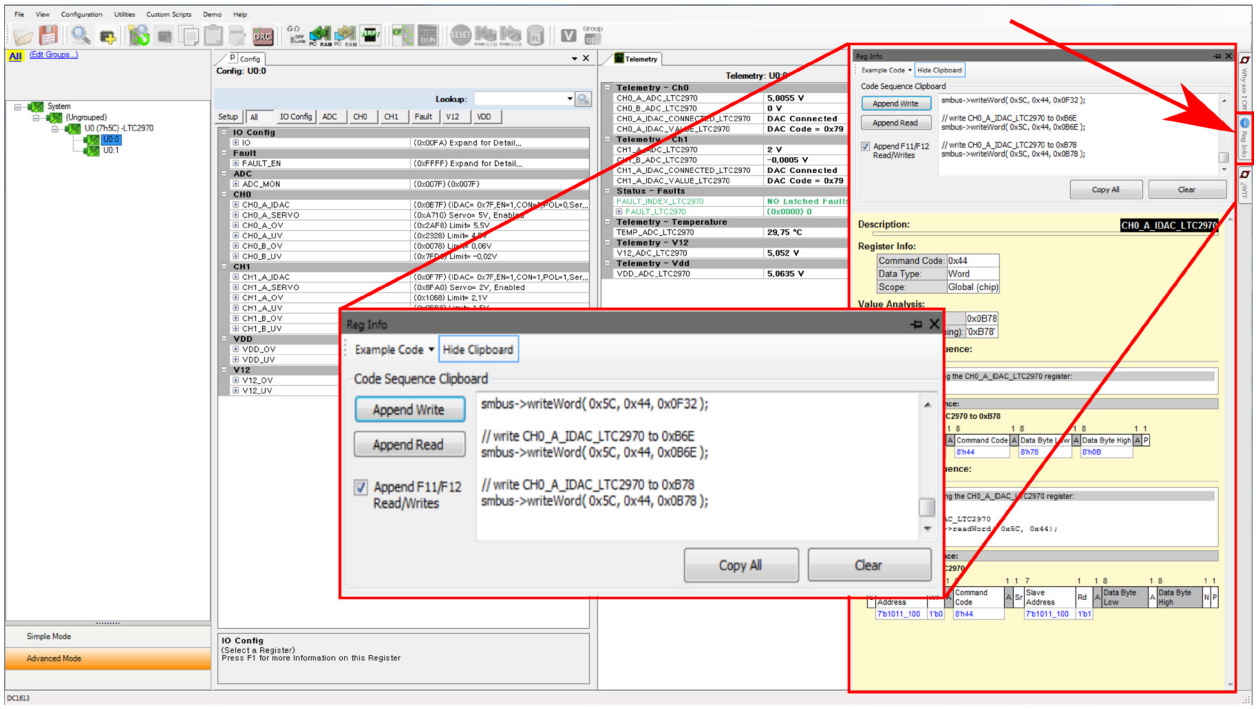


Figure 10. LTpowerPlay Reg Info Tab and Code Generation

Debugging

A very common method of debugging firmware code involves running to a breakpoint then using a debugger to examine the contents of registers and using equipment to measure conditions in the circuit before allowing the code to advance. In the Linduino environment, this task is made simple with the addition of the LTpowerPlay GUI. It is easy to use LTpowerPlay to observe the effects of running Linduino code. It is also easy to make changes in the LTpowerPlay GUI then paste the generated code into the Linduino environment.

The general procedure for debugging is similar to the technique given above in the section: Using LTpowerPlay with DC2467. The key is to make sure that the Linduino and LTpowerPlay take turns and do not conflict in their use of the I^2C bus. Use the “GO ON LINE” and “GO OFF LINE” buttons in LTpowerPlay.

A prime example of using LTpowerPlay in debugging is in decoding and deciphering fault conditions. Faults can come in different varieties, may be transient and intermittent, and are presented as bits in registers, which must be decoded and deciphered. Most firmware code does not decode the fault state and print helpful debug messages to the serial monitor for human convenience. LTpowerPlay does, however, and it contains a powerful set of helpful tools for visualizing the fault registers and fault logs. If a fault condition results during code execution, LTpowerPlay helps to determine what happened.

Another example is using the VERIFY button to compare the register state of the part to the register “expected values” from a project file.

This is useful to determine what registers have been changed by running code. Since LTpowerPlay maintains a representation of the register states, it can compare this representation against what it finds in the hardware registers and present a compact report of any discrepancies that it finds.

THE DC2467 BOARD

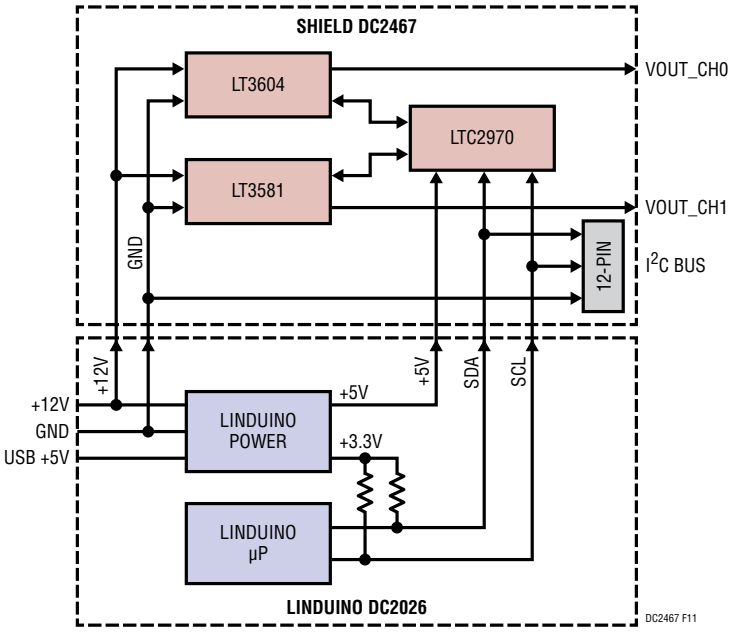


Figure 11. DC2467 Block Diagram

LTC2970 FUNCTIONS

Turning On and Off

The LTC2970 has two programmable GPIO pins that can function in several modes. On the DC2467 these two pins are configured as enables for the two onboard regulators. Setting a GPIO output high enables the attached regulator. It also illuminates the associated GPIO LED on the board, indicating that the channel is active.

Monitoring Voltage

The LTC2970 has two differential 14-bit ADC inputs for monitoring voltage at the output of the regulators. These inputs can measure inputs from 0V to 6V, so are ideally suited for the +5V supply. For the negative supply we use resistors to level-shift the voltage up above ground. The Linduino code has scale and offset parameters to compensate.

Margining High and Low

The LTC2970 has two 8-bit DAC outputs that create offset currents to move the regulator outputs up and down.

The LTC2970 can affect a change at the regulator output using one of two methods: open-loop DAC forcing, or closed-loop servo. The open-loop method simply sets the DAC voltage to a code and forces the regulator output without measuring its voltage. The closed-loop method measures the regulator output voltage and moves the DAC to the code that produces the desired voltage. The open-loop method is faster, but error prone. The closed-loop method is slow, but can be as accurate as the ADC.

The Linduino code contains a menu item for soft-connecting the DAC (to minimize regulator voltage disturbance) and for servoing the voltages (with a closed loop). Because

LTC2970 FUNCTIONS

the negative supply has ADC gain and offset terms, the Linduino code uses an adjusted voltage target to make the servo loop achieve the correct output voltage (similar to the scale and offset parameters in LTpowerPlay—Figure 9).

Monitoring Current

The LTC2970 has two differential 14-bit ADC inputs for monitoring current by measuring voltage across a current sense resistor. These are the "CHn_B" ADC inputs, and have the same specifications as the ADC inputs for voltage monitoring, so they can be used over a wide range.

For the +5V output rail, the channel 0 (CH0) ADC measures voltage across a 0.02Ω sense resistor with 500µV precision, so the smallest current that it can resolve is 25mA.

For the –5V rail, the ADC cannot directly measure current, since it cannot touch voltage below ground. Instead there is a current sense amplifier (LTC6105) that translates the current flowing in the sense resistor into a voltage above ground at a factor of 1V/A.

Reading Faults

The LTC2970 has a complete set of fault monitoring capabilities for detecting limit excursions of any of its ADC readings. In the DC2467 this amounts to limits on output voltages and load currents. In addition, the LTC2970 can throw faults for excursions on the +12V power supply input and the VDD regulator voltage, as well as if the DAC reaches code 0 or code 255, the limits of its control range.

The Linduino code can read the fault registers and report if faults are present. The DC2467 sketch initializes the LTC2970 to have reasonable fault limits for the DC2467 board. In addition, the DC2467 has an ALERT LED that can be programmed to illuminate when the LTC2970 detects a fault.

POWER OUTPUTS

The DC2467 has two switching regulators, one producing +5V and one producing –5V. Both are monitored and controlled by the LTC2970, as shown in Figure 12 and Figure 13.

The DC2467 requires power for the LTC2970 and power for the DC/DC regulators. LTC2970 derives power from the 5V supply in the USB. The Linduino and LTC2970 will begin to communicate as soon as a USB connection is established. The LTC3604 and LT3581 regulators require a separate +12V supply. This must be plugged in to the +12V DC jack on the Linduino before the regulators can operate. If +12V is not present, the LTC2970 will read 0V and 0A at its inputs.

The DC2467 has several LEDs to indicate status. These are labeled as follows:

Table 3. DC2467 LED Definitions

LABEL	FUNCTION
PGOOD0	Power good indicator for VOUT_CH0
PGOOD1	Power good indicator for VOUT_CH1
GPIO_0	Active high GPIO_0 state (enable CH0)
GPIO_1	Active high GPIO_1 state (enable CH1)
ALERT	Active low ALERTB state indicator
VDD_5V	Power good: 5V Power from Linduino
VDD_12V	Power good: 12V Power from Linduino
SCL	I^2C bus activity indication

POWER OUTPUTS

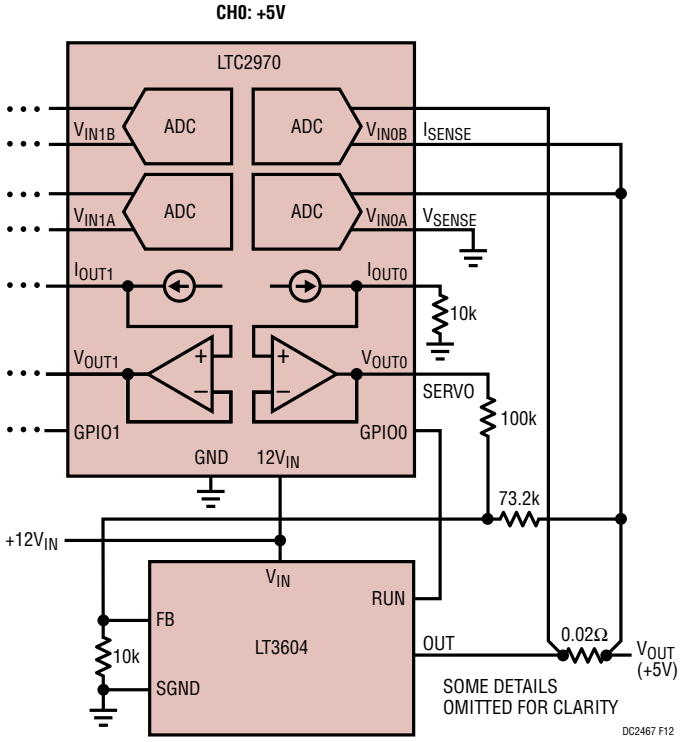


Figure 12. LTC2970 Interface to +5V Rail

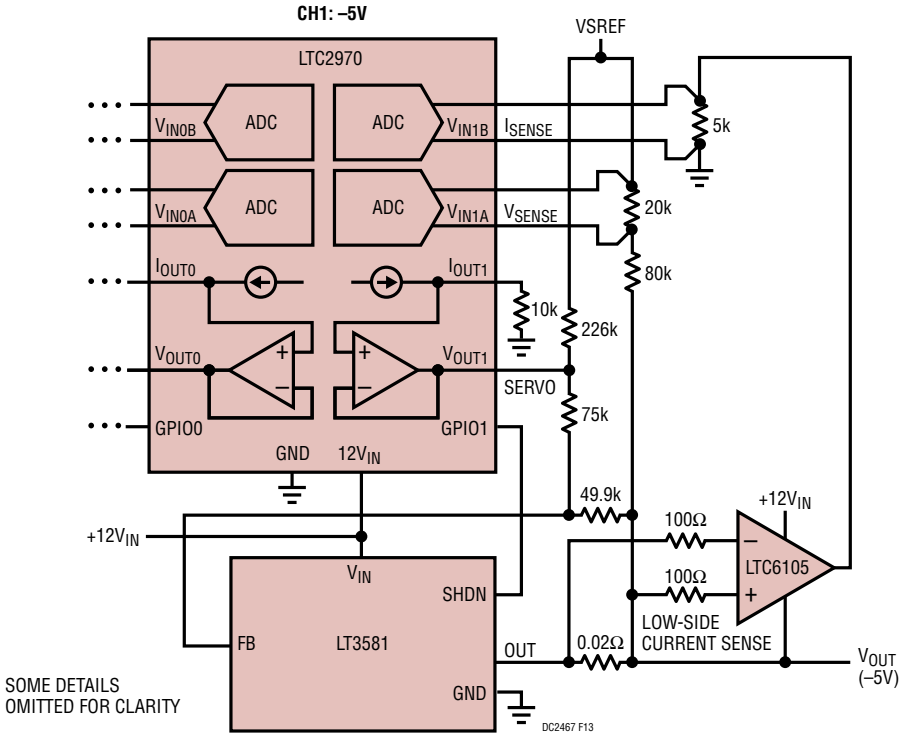


Figure 13. LTC2970 Interface to –5V Rail

NOTES AND TROUBLESHOOTING

Note 1: The Linduino documentation may recommend an older version of the Arduino software. It is not necessary to use this old version. In most cases the newer Arduino software has improvements and enhancements that make it superior. The code in the LTSketchbook usually compiles cleanly with the newest software. If, however, you encounter trouble compiling code in the LTSketchbook, you may need to fall back to an older version of the Arduino software.

Note 2: A brand new Linduino, when it comes from the factory, contains a special sketch, called the DC590B. This sketch makes the Linduino perform the functions of the DC590 dongle, translating from USB to I^2C and enabling the use of either LTpowerPlay or QuikEval™ with some demo boards. It is possible to make LTpowerPlay talk to the DC2467 through this DC590 emulator, but it is not recommended. We leave it to the user to get LTpowerPlay talking through this interface. It will function, but for various technical reasons it is not optimal.

Note 3: COM port selection for the Arduino software is something of an art. In MS Windows, the simplest way to determine for certain which COM port is associated with your Linduino is to open the Device Manager and look for the "USB SERIAL CONTROLLER" (not the "LTC DC1613A"). Clicking "Properties" should bring up a window containing information about the COM port that it is using (Figure 14). Of course, you can always use trial and error until you find the correct COM port in the list, which may be a shortcut.

Note 4: Linduino uses C++ code architecture, with pointers to class objects defining the SMBus interface. Another way to code the I^2C bus transactions is with standard C function notation (no objects or classes). Select this style with the "Example Code" button at the top of the "Reg Info" tab in LTpowerPlay. Make sure to use the C++ style when running Linduino code. The difference looks like this:

```
// Linduino C++ notation:
some_var = smbus->readWord( 0x5C, 0x50);
// standard C notation:
some_var = smbus_read_word( 0x5C, 0x50);
```

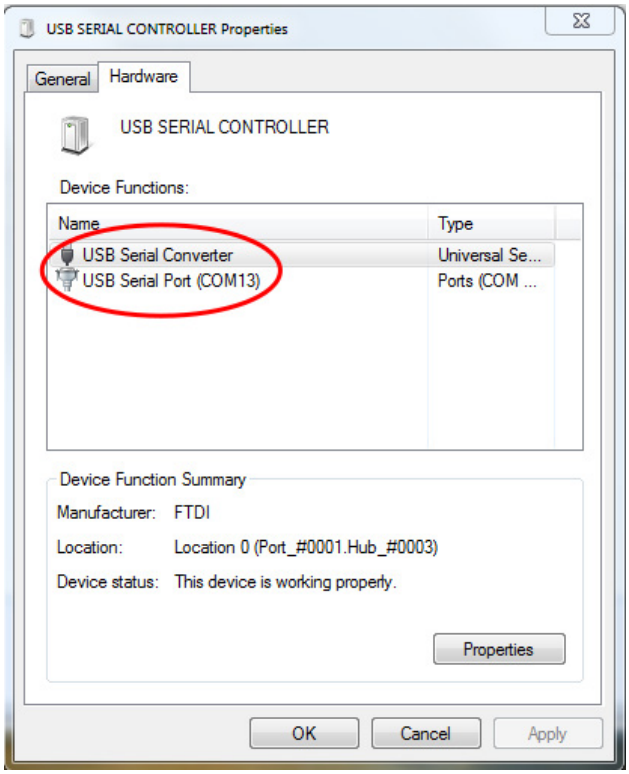


Figure 14. MS Windows Device Properties Window for Linduino

Note 5: The Arduino software looks for the LTSketchbook at start-up time and constructs its menu and its pointers according to the files that it finds. If you add or remove files from the directory tree while Arduino is running, the tool will not show these changes until you restart. This applies to the LTSketchbook itself. If you pull down File→Preferences and change the "Sketchbook location" field, you must restart Arduino before it will find the new files.

Note 6: Linduino has a *black* 14-pin header. This will be covered and hard to see when the DC2467 shield is attached. Do not attempt to plug in the 12-pin ribbon cable from the DC1613 dongle to the Linduino 14-pin header. People have done this before and it never ends well. Instead, look for the *white* 12-pin header on the DC1613 shield and plug in the dongle there. Follow Figure 7.

NOTES AND TROUBLESHOOTING

Note 7: Linduino has a limited amount of memory, both RAM and nonvolatile, in which to store programming and variables. It is possible to write a program that is too large to fit into nonvolatile memory, or takes too much RAM during execution. This is rare, but it can cause unpredictable behavior. The Arduino compiler attempts to calculate these limits and warn you when the program is too big. Pay attention to the compiler log messages in the bottom of the Arduino window during compilation and uploading. They will look something like below:

```
Sketch uses 18,088 bytes (56%) of program
storage space. Maximum is 32,256 bytes.

Global variables use 566 bytes (27%) of
dynamic memory, leaving 1,482 bytes for
local variables.
Maximum is 2,048 bytes.
```

Look for percentages less that 90% to be safe. The Linduino may behave badly at less than 100%. If your compiled code exceeds the memory limits of the Linduino, you may need to split your sketch into multiple smaller sketches to fit into memory.

RESOURCES

LTpowerPlay

http://www.ltpowerplay.com/download/
(contains its own documentation)

Arduino

http://www.arduino.cc/en/Guide/Windows

Linduino

A full description of the Linduino is located here: http://www.linear.com/solutions/5334

Using Linduino with Power System Management: http://cds.linear.com/docs/en/application-note/an153f.pdf

http://www.linear.com/solutions/5676

I²C bus

http://www.i2c-bus.org/i2c-bus/

Beagle (DC2294)

http://www.totalphase.com/products/beagle-i2cspi/

LTC2970

http://www.linear.com/product/LTC2970

LTC3604

http://www.linear.com/product/LTC3604

LT3581

http://www.linear.com/product/LTC3581

DC980

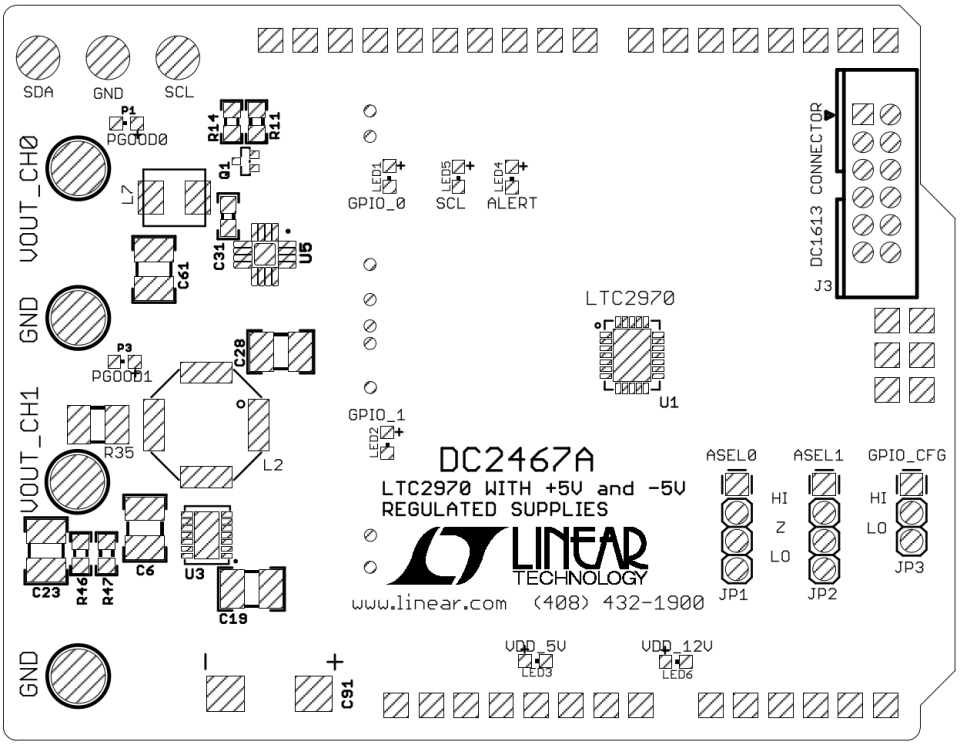
http://www.linear.com/solutions/3829

DC2294

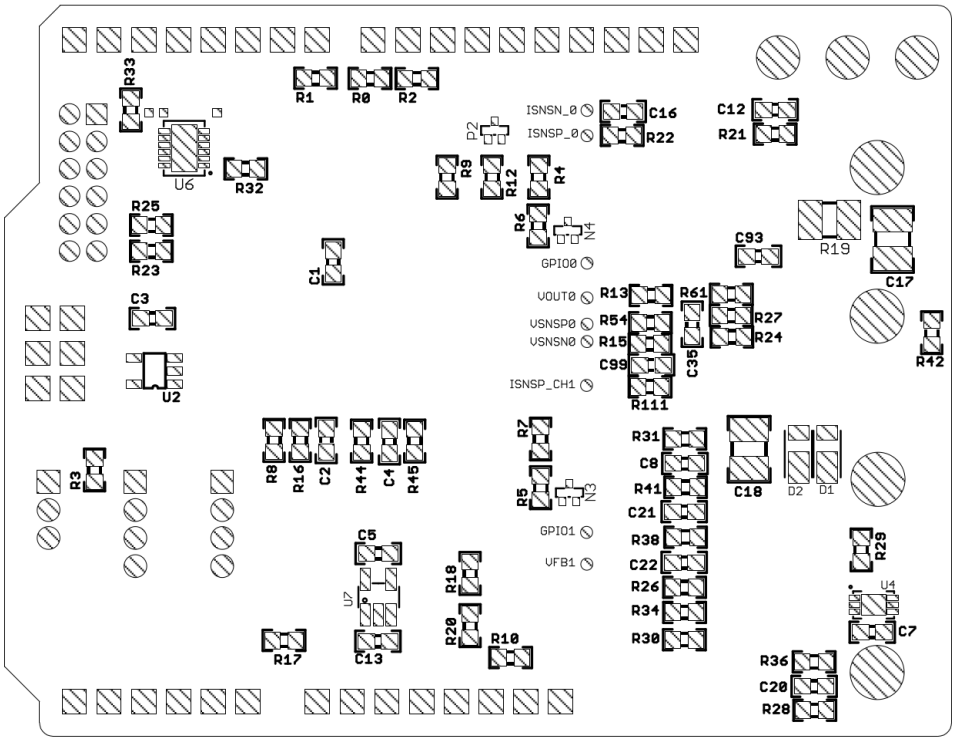
http://www.linear.com/solutions/5718

DC2467 BOARD DETAILS

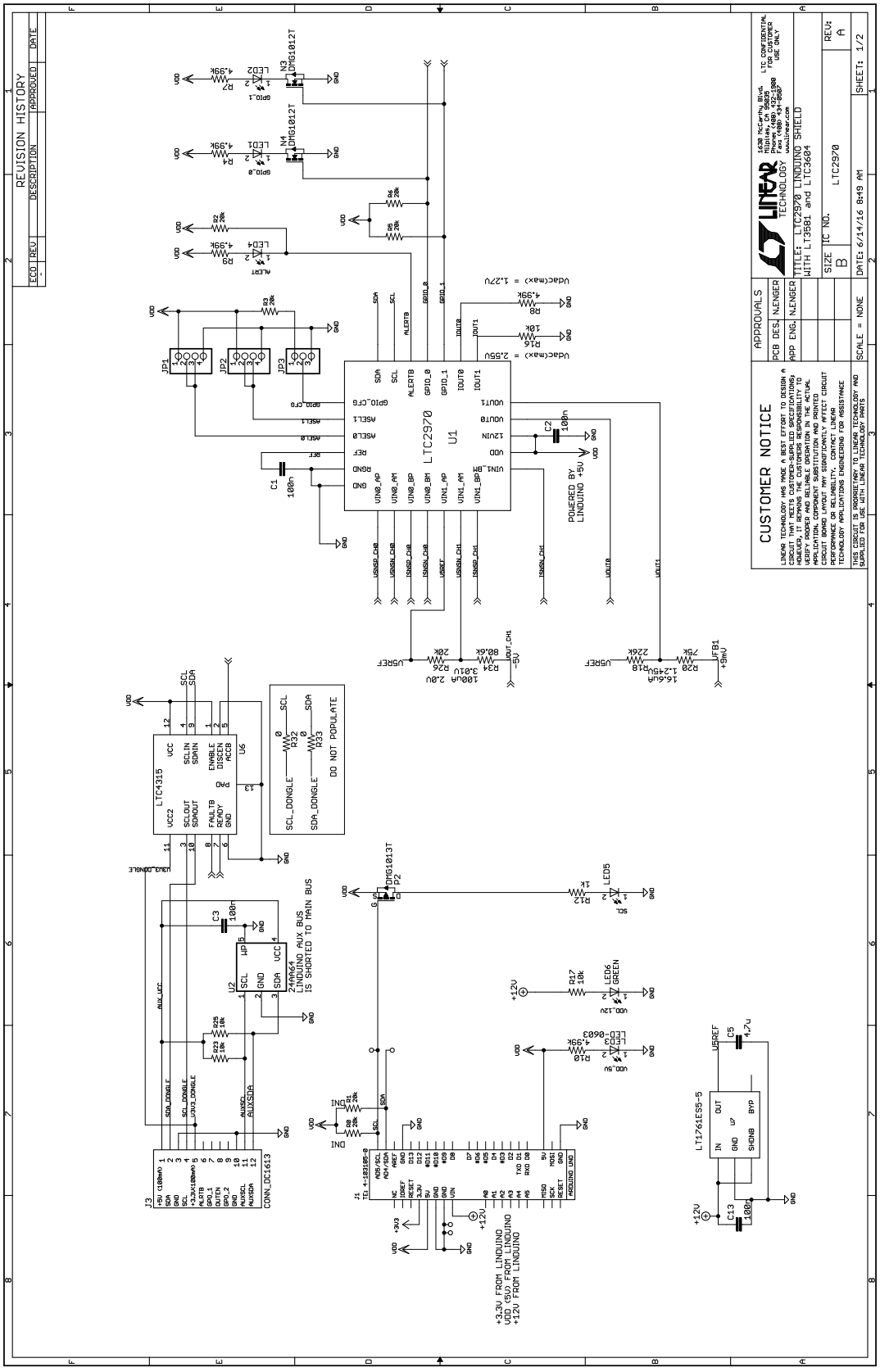
TOP



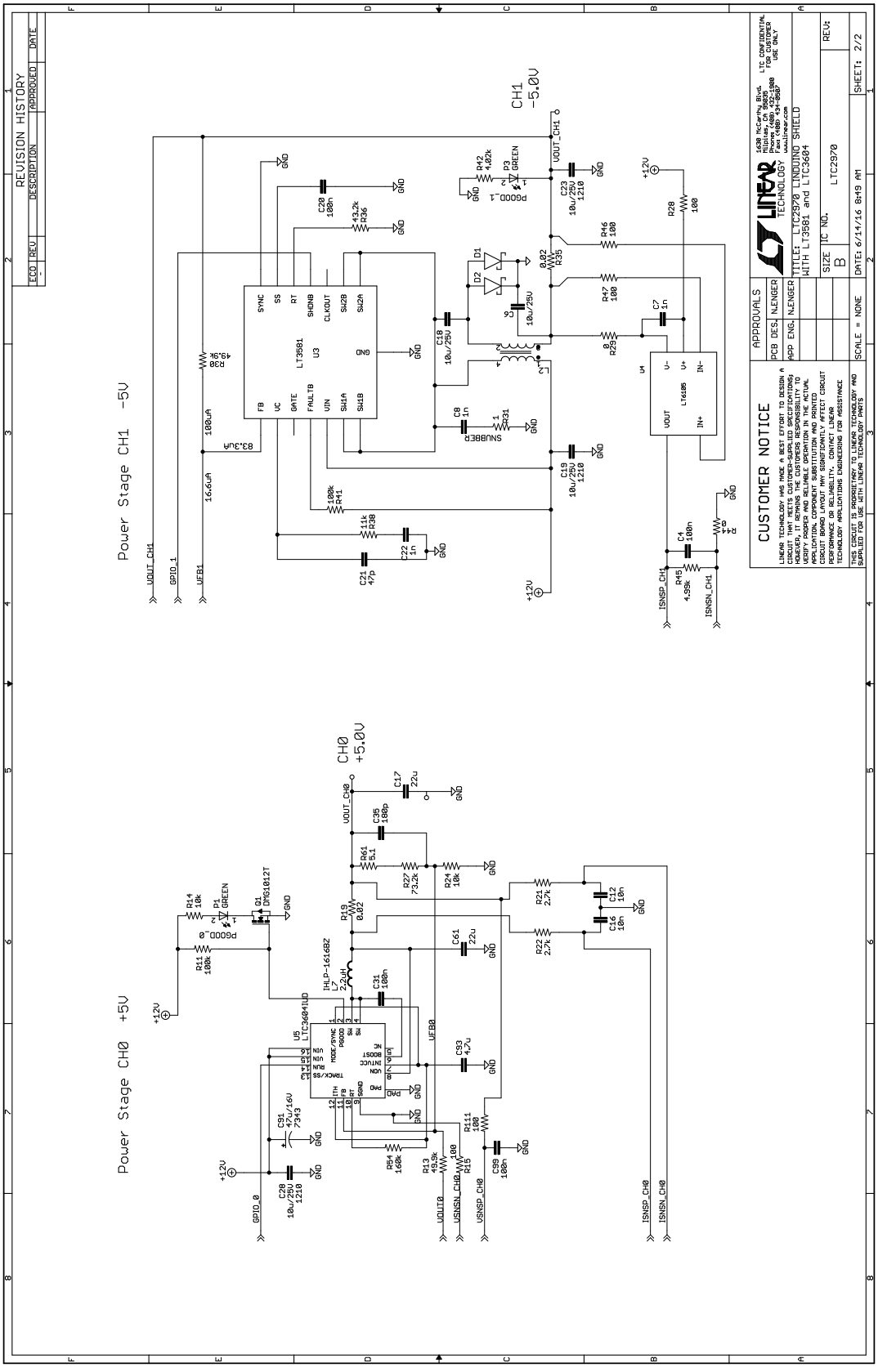
BOTTOM



SCHEMATIC DIAGRAM



SCHEMATIC DIAGRAM



DEMONSTRATION BOARD IMPORTANT NOTICE

Linear Technology Corporation (LTC) provides the enclosed product(s) under the following **AS IS** conditions:

This demonstration board (DEMO BOARD) kit being sold or provided by Linear Technology is intended for use for **ENGINEERING DEVELOPMENT OR EVALUATION PURPOSES ONLY** and is not provided by LTC for commercial use. As such, the DEMO BOARD herein may not be complete in terms of required design-, marketing-, and/or manufacturing-related protective considerations, including but not limited to product safety measures typically found in finished commercial goods. As a prototype, this product does not fall within the scope of the European Union directive on electromagnetic compatibility and therefore may or may not meet the technical requirements of the directive, or other regulations.

If this evaluation kit does not meet the specifications recited in the DEMO BOARD manual the kit may be returned within 30 days from the date of delivery for a full refund. THE FOREGOING WARRANTY IS THE EXCLUSIVE WARRANTY MADE BY THE SELLER TO BUYER AND IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. EXCEPT TO THE EXTENT OF THIS INDEMNITY, NEITHER PARTY SHALL BE LIABLE TO THE OTHER FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES.

The user assumes all responsibility and liability for proper and safe handling of the goods. Further, the user releases LTC from all claims arising from the handling or use of the goods. Due to the open construction of the product, it is the user's responsibility to take any and all appropriate precautions with regard to electrostatic discharge. Also be aware that the products herein may not be regulatory compliant or agency certified (FCC, UL, CE, etc.).

No License is granted under any patent right or other intellectual property whatsoever. **LTC assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or any other intellectual property rights of any kind.**

LTC currently services a variety of customers for products around the world, and therefore this transaction **is not exclusive**.

Please read the DEMO BOARD manual prior to handling the product. Persons handling this product must have electronics training and observe good laboratory practice standards. **Common sense is encouraged**.

This notice contains important safety information about temperatures and voltages. For further safety concerns, please contact a LTC application engineer.

Mailing Address:

Linear Technology
1630 McCarthy Blvd.
Milpitas, CA 95035

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Power Management IC Development Tools ***category:***

Click to view products by Analog Devices ***manufacturer:***

Other Similar products are found below :

EVAL-ADM1168LQEBZ EVB-EP5348UI MIC23451-AAAYFL EV MIC5281YMME EV DA9063-EVAL ADP122-3.3-EVALZ ADP130-0.8-EVALZ ADP130-1.2-EVALZ ADP130-1.5-EVALZ ADP130-1.8-EVALZ ADP1714-3.3-EVALZ ADP1716-2.5-EVALZ ADP1740-1.5-EVALZ ADP1752-1.5-EVALZ ADP1828LC-EVALZ ADP1870-0.3-EVALZ ADP1871-0.6-EVALZ ADP1873-0.6-EVALZ ADP1874-0.3-EVALZ ADP1882-1.0-EVALZ ADP199CB-EVALZ ADP2102-1.25-EVALZ ADP2102-1.875EVALZ ADP2102-1.8-EVALZ ADP2102-2-EVALZ ADP2102-3-EVALZ ADP2102-4-EVALZ ADP2106-1.8-EVALZ ADP2147CB-110EVALZ AS3606-DB BQ24010EVM BQ24075TEVM BQ24155EVM BQ24157EVM-697 BQ24160EVM-742 BQ24296MEVM-655 BQ25010EVM BQ3055EVM NCV891330PD50GEVB ISLUSBI2CKIT1Z LM2744EVAL LM2854EVAL LM3658SD-AEV/NOPB LM3658SDEV/NOPB LM3691TL-1.8EV/NOPB LM4510SDEV/NOPB LM5033SD-EVAL LP38512TS-1.8EV EVAL-ADM1186-1MBZ EVAL-ADM1186-2MBZ