

## Overview of the EEPROM in LTC PSM Devices

Nick Vergunst

### INTRODUCTION

Linear Technology® has a large family of devices that provide a great deal of power and configurability for all applications. These parts additionally provide onboard nonvolatile memory (NVM) in the form of EEPROM to store and recall configuration parameters and on some devices provide a fault log and user scratch pad.

These Power System Management (PSM) devices' architectures allow them to power up and load the desired configuration parameters from this NVM autonomously with no I<sup>2</sup>C or firmware interaction required.

A common question customers ask is, "Now that I have settled on a particular configuration, how do I program this configuration into the chip's onboard nonvolatile memory (NVM)."

Some available options are presented in rough order of increasing complexity throughout the document.

**NOTE:** Linear Technology strongly recommends **Option 1** for preproduction prototyping, and **Option 2**, **Option 3** or **Option 4** for higher volume production. **Option 5** is a convenient method to program a small quantity of loose ICs prior to board assembly. **Option 6** requires advanced software development expertise.

### CONVENIENT LIST OF ALL OPTIONS

- Option 1A.** Manual programming with LTpowerPlay™ Built In Programming Utility (PC->NVM)
- Option 1B.** Manual programming with LTpowerPlay in Normal Mode (PC->RAM->NVM)
- Option 1C.** Fully automated programming with LTpowerPlay Command Line Interface
- Option 2A.** Buy pre-programmed parts through Linear Express® for quantities of 1 to 499 units
- Option 2B.** Buy pre-programmed parts from Linear Technology Corporation for production quantities > 500 units
- Option 3A.** Self-program parts using a BPM Programmer
- Option 3B.** Buy pre-programmed parts from Arrow
- Option 4A.** Self-program parts via JTAG on custom board with an IO device on the scan-chain
- Option 4B.** Self-program parts via JTAG on custom board without an IO device on the scan-chain
- Option 5.** Self-program parts using a LTC® PSM Socketed Programming Board
- Option 6.** Author custom firmware and self-program parts

### SO MANY CHOICES! WHICH ONE DO I CHOOSE?

There are many options available for programming LTC PSM products and it is possible to implement multiple strategies on a single board to fully take advantage of the powerful tools at your disposal.

LT, LT, LTC, LTM, Linear Express, Linear Technology and the Linear logo are registered trademarks of Linear Technology Corporation and LTpowerPlay is a trademark of Linear Technology Corporation. All other trademarks are the property of their respective owners.

# Application Note 145

We strongly recommend having an accessible header on your board that allows access to the VCC lines of the PSM devices, I<sup>2</sup>C clock (SCL), I<sup>2</sup>C data (SDA), and a ground. Having this header provides an entry point for LTpowerPlay which not only allows you to program the board easily, but also gives access to a very powerful tool suite to read telemetry, change variables on the fly, and perform debug.

During the debug stages this is extremely valuable because in today's constantly evolving power environments, the final requirements for the power supply and overall system may not be known until well into the development cycle. The ability to tweak or completely change the configura-

tion to fit system requirements at any point in the design process can be invaluable.

For prototypes or low to medium volume production runs, the basic header may be all you need to program the PSM devices by attaching an external USB dongle such as the DC1613A. This would be an example of programming Option 1A, Option 1B, Option 1C and Option 4B.

To reiterate, having this header is resolutely encouraged independent of which programming method you choose! The short-list of reasons includes the ability to program your PSM devices onboard powered via the dongle and the ability to keep the brains of the PSM device alive, independent of the power supply rails on the board, which

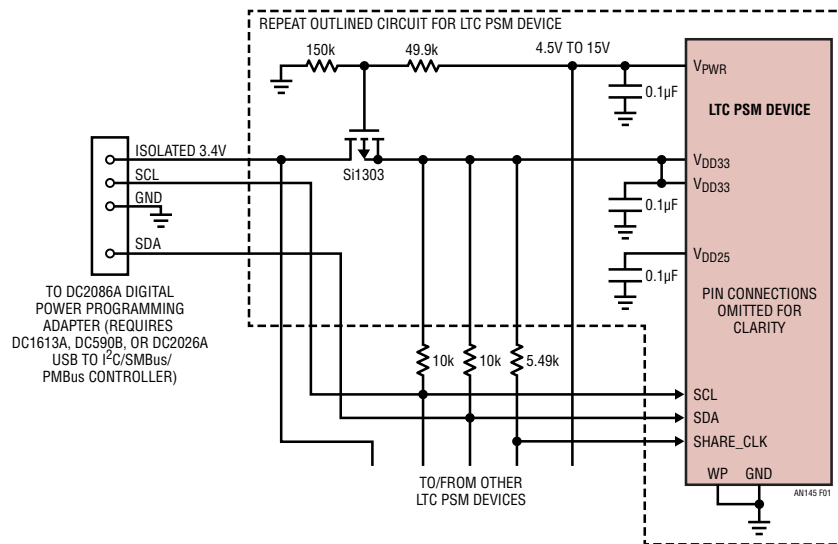


Figure 1. LTC Controller Connections when Powering LTC PSM Devices with PFET to V<sub>PWR</sub>

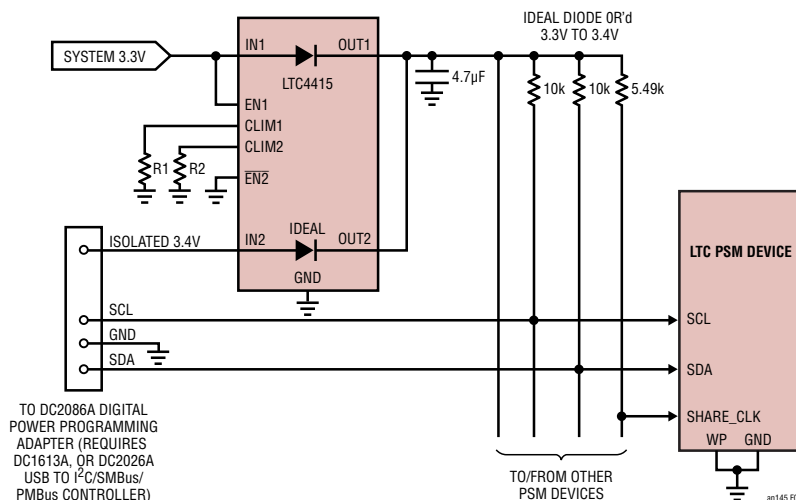


Figure 2. LTC Controller Connections when Powering LTC PSM Devices from either System Power or Dongle Power

is extremely useful in any debugging situation. When used in conjunction with LTpowerPlay software, the device then provides a powerful way to debug an entire power system. Failures are quickly diagnosed using telemetry, fault status registers and the fault logs. The final system configuration can be quickly developed and stored to the PSM products' respective nonvolatile memories all using this one interface.

Please see Figure 1 and Figure 2 for examples on how to design your board to make full use of these strikingly important features and allow for your LTC PSM products to be powered from the DC1613A or DC2086A.

If you do not want to program these devices on your board, then there are multiple options available as well. If you would like to order default parts, program using a socket and then solder onto your board, you should choose Option 3A or Option 5. If you want to program more than one device at a time, then you can select a multi-site programmer from BPM for Option 3A.

If you would rather order pre-programmed parts then you should choose Option 2 or Option 3B. If you select either of these options, your parts will come delivered to you with the configuration file you select already programmed into the device. **Be warned** that if you go this route and fail to heed the previous recommendation of putting a basic header on your board, LTpowerPlay will not be able to reprogram your parts if your requirements change later or if you want to do any sort of debug.

If you have another smart device on your board, such as an FPGA, ASIC, MCU, etc., then you can elect to author custom firmware using Option 6. If your smart device is JTAG capable and the I<sup>2</sup>C lines are tied to a boundary scan node then you can use Option 4A to program the LTC PSM device at the same time as programming the rest of your JTAG devices. This is good for saving time during assembly for medium to high volume production runs.

No matter which option you choose, it is highly recommended that you archive the \*.zip file that LTpowerPlay creates from your programming files for later verification of first article devices.

## OPTIONS INVOLVING LTpowerPlay GUI

Most of the options presented here involve use of Linear Technology powerful LTpowerPlay GUI (available from <http://www.linear.com/LTpowerPlay>). In general, you will use the LTpowerPlay GUI and your target hardware's demo board (such as the DC1540B for the LTC2977) to build a project file for one or more IC's on your board. The demonstration system allows you to verify that the configuration parameters provide the desired power supply management behavior prior to your own board bring-up. Once you have saved your configuration parameters into a project file (.proj file), you are ready to implement one of the following methods to transfer this configuration to the nonvolatile memory (NVM) of the IC's on your board.

### Option 1: Use LTpowerPlay to Program LTC PSM Devices on the Customer Board

The LTpowerPlay GUI is designed to communicate not only with Linear Technology designed demo boards, but also with any number of LTC devices on a custom board.

**NOTE:** Linear Technology strongly recommends Option 1 for pre-production prototyping as it requires **no knowledge of I<sup>2</sup>C protocols and no firmware development**. This option also allows you to change the configuration on your board without de-soldering parts.

### Option 1A: Manual Programming with LTpowerPlay Built In Programming Utility

LTpowerPlay has a built-in Programming Utility that easily programs and verifies the configuration file. Connect the LT USB-PMBus controller (DC1613A) to your PC via USB and to the target board. Also verify that the write protect pin is low and software write protect is disabled. See Figure 1 and Figure 2 for a more detailed schematic of the process.

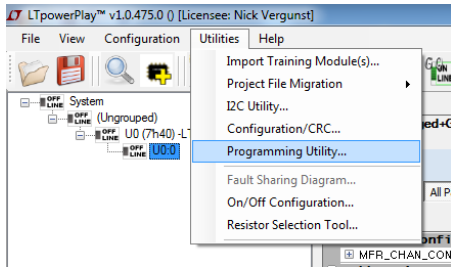
In the LTpowerPlay GUI:

- Make sure you save your current project file if you haven't done so already. The following steps will close the currently opened project file and discard any

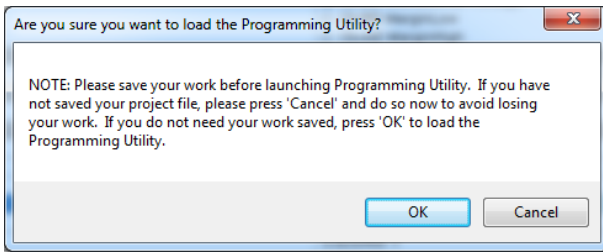
# Application Note 145

unsaved changes in order to open the project file for programming.

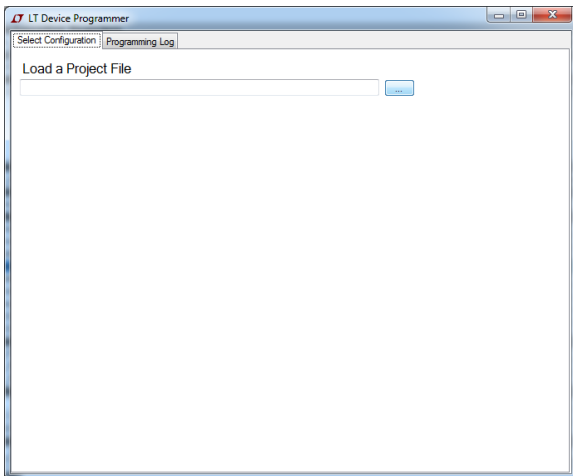
- Select Utilities->Programming Utility



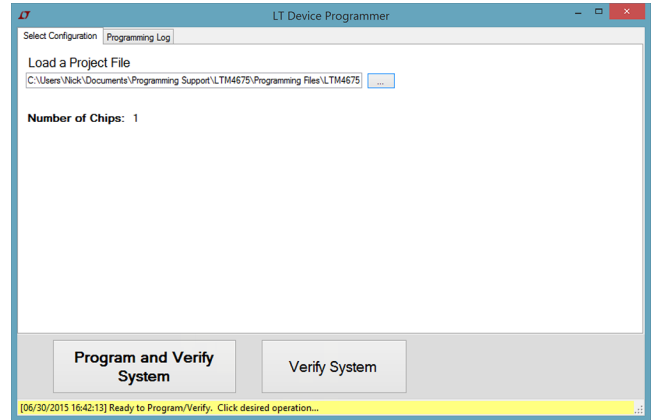
- LTpowerPlay will then ask you to confirm that you have followed the first step. If you have not saved your currently opened project and wish to, cancel this prompt and restart the process when ready



- The "LT Device Programmer" window will then pop-up. Make sure you are on the "Select Configuration" tab. Then click the "..." button to browse for a project file you wish to use as the source file to program devices



- Browse for your \*.proj file and when you select Open the LT Device Programmer window will change into this:



- You now have 2 additional large buttons near the bottom of the window.
  - The Program and Verify button will first program the device's NVM to the correct state and then verify its contents.
  - The Verify button will NOT program the device's NVM, but will attempt to verify the NVM contents with the project file's content. See Appendix B for more information.
- For this exercise we will Program and Verify. The status message on the bottom will change to yellow and say "Programming" then "Verifying" and eventually when successful it will display "Successfully Programmed and Verified Device"



You can now attach the USB dongle to another board to be programmed, press the Program and Verify button and the new board will be programmed like the first one.




It is highly recommended that you save the project file that you are using to program the devices to a separate version controlled repository, so that later verification of programmed devices can be performed if ever required.

## Option 1B: Manual Programming with LTpowerPlay in Normal Mode (PC->RAM->NVM)

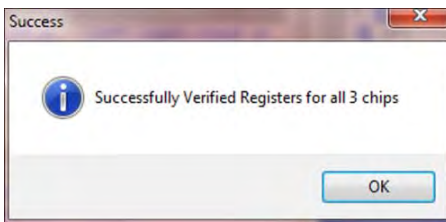
The process is quite simple:

Connect the LT® USB-SMBus controller (DC1613A) to your PC via USB and to the target board. See Figure 1 and Figure 2 for a more detailed schematic of the process.

In the LTpowerPlay GUI:

- Load your project (.proj) file using the “Open” button in the toolbar 
- Click “Go Online” in the toolbar 
- Click the “Write All” button in the toolbar 

The GUI, will program and verify each operational register according to the project file. If this process is successful, you will see a message similar to this:



- Click the “Store User All” button in the toolbar 

All the chips on your board now have the desired configuration stored into their NVM.

## Option 1C: Fully Automated Programming with LTpowerPlay Command Line Interface

The other way to program parts directly via the GUI is to pass the shell command line parameters. It will then automatically connect, program, and verify your device without any further human interaction. This can be very useful in a low-volume production environment where all the programming is fully automated.

From a Microsoft Windows environment, open a command shell window. This is most easily done in XP/Vista by going to Start->Run then typing in “cmd.exe” and in Windows7/Windows8 by typing “cmd” into the Start menu.

Alternatively you can browse to your Windows/System32 folder where you will find cmd.exe and can open it like any other program.

**NOTE:** You can use other programming languages to further automate the automatic programming via the GUI. The simplest method would be to make a Windows Batch File (\*.bat) which is a simple collection of command line commands. Looping, error checking, and other features can be implemented this way and requires nothing more than a text editor.

For something that is easier to implement and maintain, many programming languages today such as C#, C++, C, VB, Delphi, and many more allow their programs to call windows command line options. This allows you to incorporate the GUI’s automatic programming into any system with little effort.

If you have not added LTpowerPlay’s installation folder to your PATH environment variable list, then you must either navigate to the installation directory or type in the full directory name for the commands. It is easiest to navigate to the directory in which you installed LTpowerPlay.

Type in “cd ” (note the space) and then the absolute directory path where LTpowerPlay is installed. The default on a x64 based system is “C:\Program Files (x86)\Linear Technology\LTpowerPlay” so you would type:

```
cd "C:\Program Files (x86)\Linear Technology\LTpowerPlay"
```

You can also copy the directory string from a Windows Explorer address bar and paste into the command shell to avoid typing and making a mistake in some versions of Windows. If you have an x86 based machine, then you only have one Program Files folder so the path would be:

```
cd "C:\Program Files\Linear Technology\LTpowerPlay"
```

Now you should see a command prompt in that directory into which you can type. To program and verify a device with the project file stored in “C:\myProgrammingProject.proj” via this command line, you would type:

```
LTpowerPlay.Shell.exe -Program -Verify -ProjectFile="C:\myProgrammingProject.proj"
```

This will automatically launch the GUI shell, program, then verify your device, and return a success message or an error depending on what happens. Note that you must have the dongle connected just as when doing manual programming with the GUI. It is highly recommended that

# Application Note 145

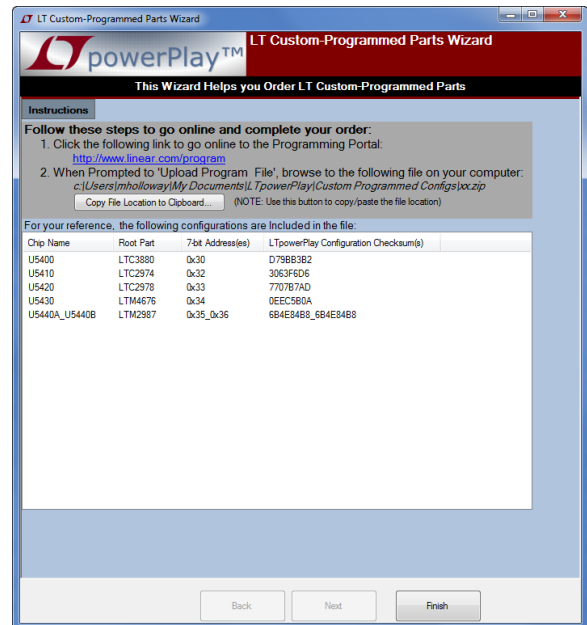
you save the project file that you are using to program the devices to a separate version controlled repository so that later verification of programmed devices can be performed if ever required.

## Option 2A: Use Linear Express to Order Pre-Programmed Parts from Linear Technology Corporation (Sample Quantities < 500 units)

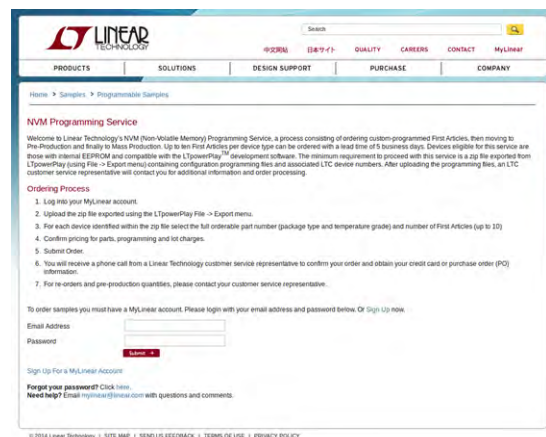


Linear Express First Article and Pre-Production Programming, LTExpress, is a fast and reliable way of getting pre-programmed parts directly from Linear Technology Corporation. The tight integration with LTpowerPlay makes it easy. After you package your project using the one-click operation in the GUI, you upload the file to the linear.com website to complete the process and your programmed devices will arrive within a couple days.

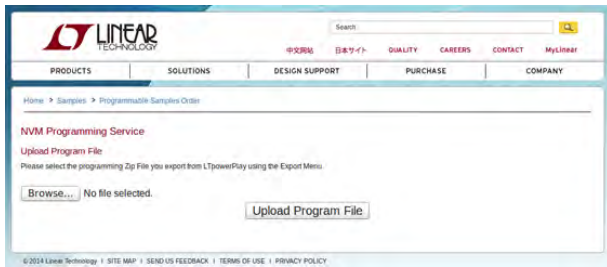
- Open your project file that contains the full system of your board. You can later de-select individual items if you choose to not have those devices programmed.
- Select File -> Order Custom LT-Programmed Parts from the top menu in LTpowerPlay
- The LT Pre-Programmed Parts Programming Wizard menu pops up. Click Next to continue
- The wizard will then automatically package all your devices into a single archive file in your Documents folder (My Documents on legacy OS's) under "My LTpowerPlay Files\LT Programming Orders". The \*.zip archive will automatically be named as what the project file is called with the date and time appended to the end. The window being displayed in LTpowerPlay will display the full path to the archive it created and the full name of the archive. It also includes a table with each of the devices included in the archive with the part number, address of the device, and the checksum for that particular configuration.



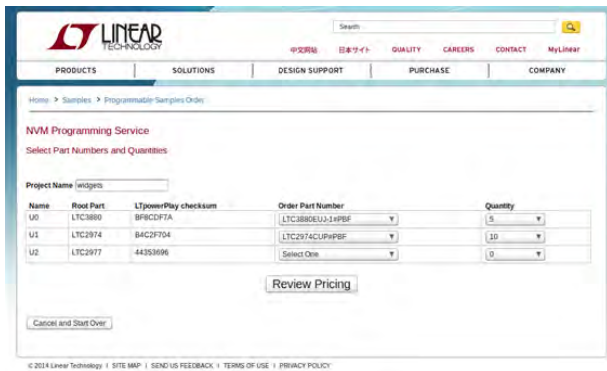
- There is a link on that window that will take you to <http://www.linear.com/program>, the online programming portal. If the link does not work, open your web browser and navigate to <http://www.linear.com/program>. On that page there is a tutorial video to walk you through these steps as well.



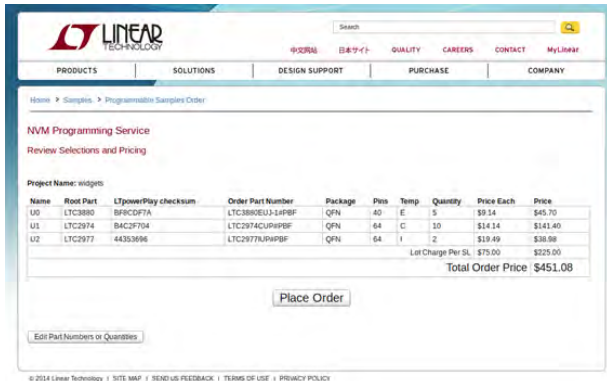
- Log into your MyLinear account
- Upload the \*.zip file exported from LTpowerPlay using the "File" -> "Order LT Custom-Programmed Parts..." menu item. If you click the "Copy File Location to Clipboard" button in the Parts Wizard, you can then paste the exact location into the portal upload.



- For each device identified within the zip file select the full orderable part number (package type and temperature grade) and number of First Articles (up to 10)



- Confirm pricing for parts, programming and lot charges



- Place Order
- You will then be contacted by a Linear Express customer service representative to confirm your order and obtain your credit card or purchase order (PO) information

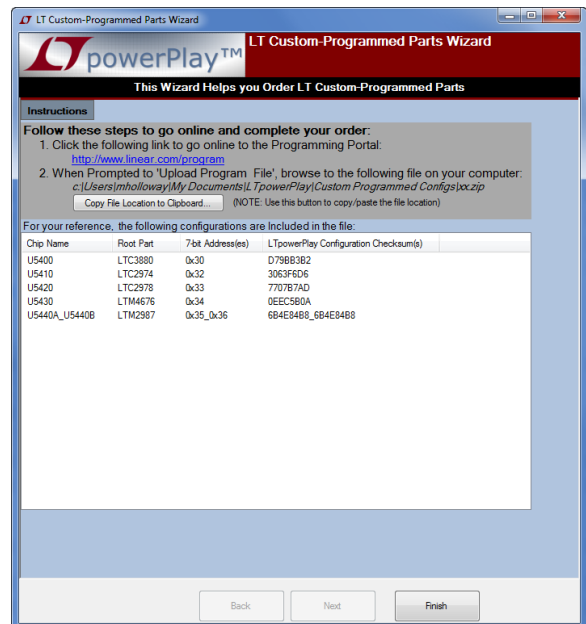
For re-orders and pre-production quantities, please contact your customer service representative.

It is highly recommended that you save this archive to a separate version controlled repository so that later verification of programmed devices can be performed if ever required.

## Option 2B: Use LTpowerPlay to Order Pre-Programmed Parts From Linear Technology Corporation (Production Quantities)

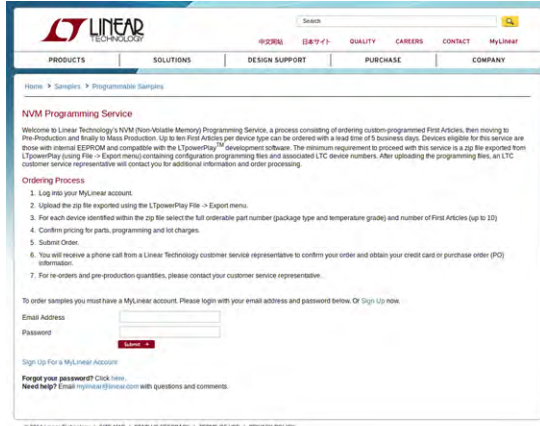
Option 2B, which is appropriate for higher volume production, is to order pre-programmed parts from Linear Technology. Contact Linear Technology for minimum order quantities. Once you have used LTpowerPlay to save the project file (.proj) that you desire. Follow these simple steps:

- Open your project file that contains the full system of your board. You can later de-select individual items if you choose to not have those devices programmed.
- Select File -> Order Custom LT-Programmed Parts from the top menu in LTpowerPlay
- The LT Pre-Programmed Parts Programming Wizard menu pops up. Click Next to continue
- The wizard will then automatically package all your devices into a single archive file in your Documents folder (My Documents on legacy OS's) under "My LTpowerPlay Files\LT Programming Orders". The \*.zip archive will automatically be named as what the project file is called with the date and time appended to the end. The window being displayed in LTpowerPlay will display the full path to the archive it created and the full name of the archive. It also includes a table with each of the devices included in the archive with the part number, address of the device, and the checksum for that particular configuration.

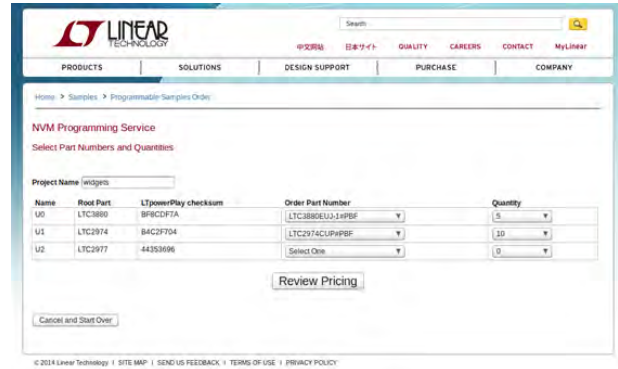


# Application Note 145

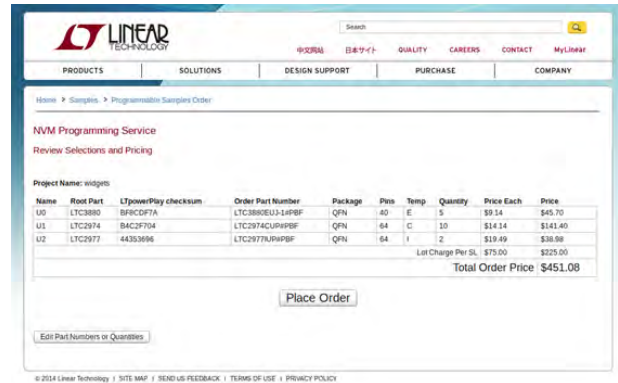
- There is a link on that window that will take you to <http://www.linear.com/program>, the online programming portal. If the link does not work, open your web browser and navigate to <http://www.linear.com/program>. On that page there is a tutorial video to walk you through these steps as well.



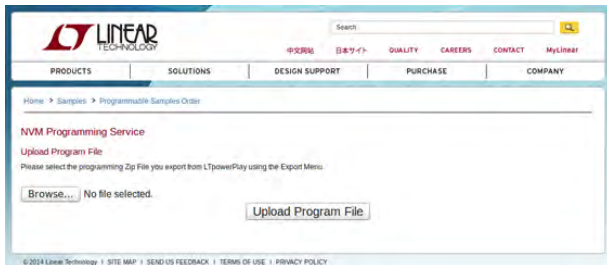
- For each device identified within the zip file select the full orderable part number (package type and temperature grade) and number of First Articles (up to 499)



- Confirm pricing for parts, programming and lot charges. If you agree, click the “Place Order” button.



- Log into your MyLinear account
- Upload the \*.zip file exported from LTpowerPlay using the “File” -> “Order LT Custom-Programmed Parts...” menu item



It is highly recommended that you save this archive to a separate version controlled repository so that later verification of programmed devices can be performed if ever required.

Simply send the .zip file to your Linear Technology Field Representative and you are ready to order pre-programmed parts<sup>1, 2</sup>.

**NOTE 1:** If you are ordering more than one unique configuration, you need to clearly specify to Linear Technology the quantity required for each unique EEPROM CRC ID.

See the instructions file included in the \*.zip file (“OrderingInstructions.txt”) which lists each I<sup>2</sup>C address (or U number) in your GUI project file and maps it with its unique EEPROM CRC ID.


**NOTE 2:** If you choose not to purchase pre-programmed devices from LTC, terms and conditions will apply. Please check with your LTC Field Representative regarding device warranty if a Contact Manufacturer or a Third Party Programming House supports programming your device.



## Option 3A: Using LTpowerPlay and a BPM Micro Programmer to Create Custom Programmed Parts

Once you have the project file (.proj) that you desire, follow these simple steps to order pre-programmed parts from a contract manufacturer or a third party programming house that uses BPM Micro programmers:

In the LTpowerPlay GUI:

- Load your project (.proj) file using the “Open” button in the toolbar 
- Under the menu, select “File, Export, To Programming File, Package Project for BPM Micro Programmer...” (See Figure 3)
- Browse for the name of a .zip file to store.

The GUI will save a .zip file in the location you specified above. The .zip file contains a sub-folder for each chip in your project file. The \*.zip file is created for your convenience to contain the individual .oem hex files required for each IC in one file. It is highly recommended that you save this archive to a separate version controlled repository so that later verification of programmed devices can be performed if ever required.

Inside each sub-folder of the .zip file is a separate OEM Programming Hex (.oem) file for the corresponding IC on your board.

- Send this .oem hex file to your contract manufacturer
- Your contract manufacturer will use this .oem file to program a number of IC’s with this desired configuration.

- Repeat this step for the individual .oem file included in the .zip file for each of the remaining chips in your project.

**Note:** The contract manufacturer requires two pieces of hardware to program your device:

- A BPM Micro programmer
- A socket module for your device

Consult <http://www.bpmicro.com> for a list of supported devices and for further details.

**NOTE:** BPM Micro programmers use the BPWin software to load data patterns and program devices. Customers and contract manufacturers commonly use a computed checksum to uniquely identify a configuration. LTpowerPlay pre-computes a checksum for you that BPWin can also use to compute and identify a configuration. More detail is included in the .zip file under the *ProgrammingInstructions.txt* file.

Open the .zip file generated above, and look at the text file *ProgrammingInstructions.txt* in the main folder. This text file includes detailed ordering instructions for each of the chips in your project file. The checksum can be used to communicate with your contract manufacturer is called the “Data Pattern CRC-32”, and this checksum is computed and listed individually for each chip in your project file in the *ProgrammingInstructions.txt* file as shown in Figure 4.

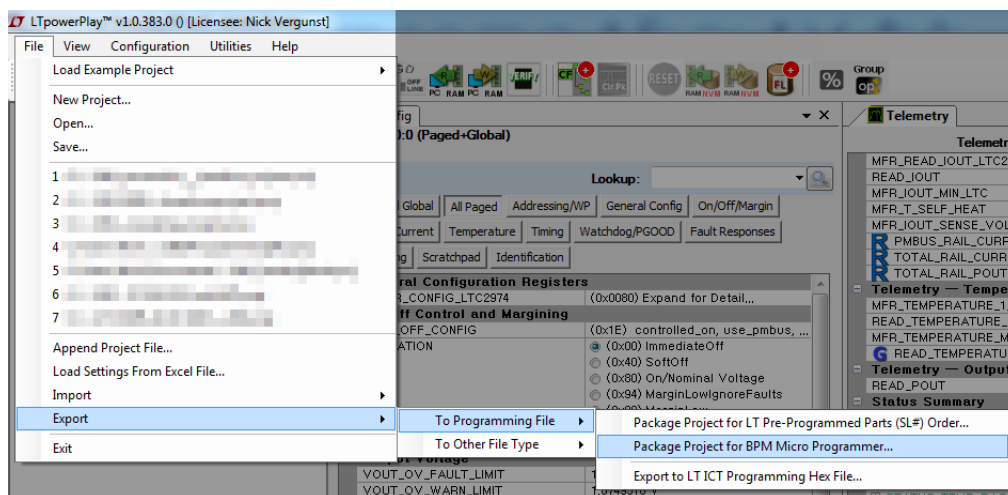


Figure 3. File, Export, To Programming File, Package Project for BPM Micro Programmer

an145f

# Application Note 145

```
ProgrammingInstructions.txt - Notepad
File Edit Format View Help
This file describes the contents of the archive .zip file "exampleprogrammingfile.zip". This .zip file contains
1 unique LT OEM Programming Hex (.hex) files, which can be used
individually with BPM Micro programming hardware to program that specific configuration.
For instructions on which files to send to the BPM Micro programmer
see the instructions below in the "Programming Instructions" section.
-----
Internal Tracking Information:
-----
The following identifying information is here for the purpose of internal tracking.
Tracking Information:
Order Package Archive File: 'fsdfs.zip'
Original Archive Location: 'c:\Users\Nick\Desktop\fsdfs.zip'
Date Created: '05/05/2014 12:41:19'
GUI Tracking Information
GUI Version: 1.0.383.0
Company: 'Linear Technology Corporation'
User Name: Nick Vergunst
User Email: nvergunst@linear.com
Computer Name: NTKworklaptop
Hardware-ID: 616A-6E7D-797E-388E-0039
Model Number: LTC2974
Silicon Rev:
Number of Chips in this Project Archive: 1
Number of unique configurations: 1
Project Structure:
Chip U0 (0x5C)
LTPowerPlay computed Configuration Checksum = 0x9420ccc2
User Config Comment: 'N/A'
-----
Page Names From the Project File:
-----
U0:0 - 'U0:0'
U0:1 - 'U0:1'
U0:2 - 'U0:2'
U0:3 - 'U0:3'
-----
Programming Instructions:
-----
This project contains 1 unique configurations.
The table below shows the individual .oem file
that corresponds to each chip in your project file:
To program an individual configuration, simply locate
the specific .oem file for the configuration you want to program
in this .zip archive and send this file to the programming house with the following instructions:
NOTE: At this time only BPM Micro programming hardware is supported.
Notes for BPMicro hardware/BPWin Software:
-----
If you are ordering pre-programmed parts from a programming
house, send these instructions to the programming house...
To enable simple serialization, enter Serial Number
Buffer Address '0x0c' and set size to 2 bytes under
Tools serialization in the BPWin software.
Please send the programming house the 'Data Pattern CRC-32'
value shown below for the configuration you want to order.
Instructions for programming house: In the BPWin software,
select the 'CRC-32' as the Checksum method, and
Data Pattern for the checksum type. The BPWin computed checksum
should match the checksum provided ("Data Pattern CRC-32")
-----
Configuration Table:
-----
chip LT configuration checksum Hex File checksums OEM (.oem) File
U0 (0x5C) 0x9420ccc2 Data Pattern CRC-32: 0x8166a339 <ZipFile>\(0x5C)
Hex File MD5: 0x08952234E0A3749FC28C0776FB842607
```

Figure 4. Programming Instructions Embedded in .zip File

The highlighted instructions in Figure 4, excerpted from *ProgrammingInstructions.txt*, explain how to:

- Configure the BPWin software to compute the “Data Pattern CRC-32” checksum
- Optionally configure the BPWin software for simple serialization of devices (serial number programming)

## OPTION 3B: USING ARROW PROGRAMMING SERVICES

Arrow utilizes the BPM programming hardware and software for their programming services<sup>3</sup>. If you choose Arrow as your programming service, please follow these steps to order your pre-programmed parts.

**NOTE 3:** If you have further questions about Arrow’s pre-programmed parts service, please contact Arrow Global Programming Services at 1-(775)-334-1000 or email: [p4FirstArt@arrow.com](mailto:p4FirstArt@arrow.com)

First follow the instructions in Option 3A to generate an individual \*.oem hex file for each unique configuration you are going to order from Arrow. As mentioned in Option 3A, this process will generate a \*.zip file with each individual \*.oem hex file and a *ProgrammingInstructions.txt* file you will need to complete your orders. Use the files within the \*.zip archive as indicated below for each unique configuration. It is highly recommended that you save this archive to a separate version controlled repository so that later verification of programmed devices can be performed if ever required.

Fill out Arrow’s pre-programming worksheet for your First Article order (<http://www.arrow.com/globalprogramming>)

* Customer Part Number	XYZ-12345
* Master / Program Name	XYZ-12345
Master / Program Name Revision	
* Device Type	Other
Checksum	DCC98102
Checksum Calculation	32-crc
Checksum Unknown (Program with Arrow Checksum)	<input type="checkbox"/>

On this web form, be sure to fill in the following:

- Enter a unique **customer part number** that corresponds to this configuration.
- Select Device Type: **OTHER**
- Enter the unique “Data Pattern CRC-32” value corresponding to the \*.oem hex file for this configuration (consult *ProgrammingInstructions.txt* in the \*.zip archive which lists the Data Pattern CRC for each \*.oem hex file)
- Select Checksum Calculation Method: **32-CRC**
- Fill in the remaining information on the form (contact Arrow Global Programming Services at 1-(775)-334-1000 or email at [p4FirstArt@arrow.com](mailto:p4FirstArt@arrow.com) if you have questions)

Once you have filled in the required information, click the **SUBMIT** button on the web form.

You will then be sent a confirmation email for this order. Use the link provided in the email to attach/upload the corresponding \*.oem hex file for the requested configurations.

## OPTION 4: JTAG PROGRAMMING

Many of the complex system boards of today have at least one JTAG programmable part. It is not uncommon for these JTAG devices to be daisy chained into a “scan-chain” to allow for programming and testing multiple devices with a single entry point.

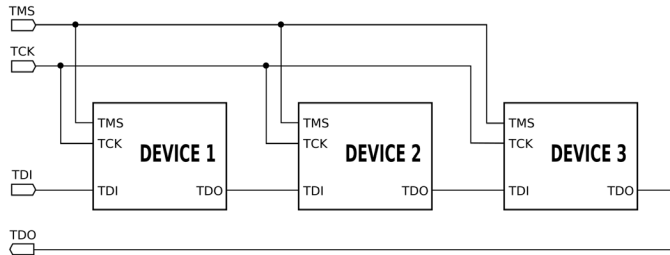


Figure 5. Example of JTAG Scan-Chain

We have worked closely with JTAG Technologies and Asset Intertech to achieve a method of programming LTC PSM Products via the standard JTAG programming interface or at a minimum the same JTAG programming hardware.

### Option 4A: JTAG Programming Using A Device in the Scan-Chain with I/O Bits Connected to the PSM Products' PMBus

The first, most convenient, and easiest method is to connect the PMBus to a device on the scan-chain with boundary scan features available on the pins. Most CPLD's, ASIC's, FPGA's, and MCU's with JTAG programmability have this ability.

Once the PMBus is connected to boundary scan capable pins in the JTAG scan-chain, you can force and measure logical high and low signals on these pins by shifting the data through the entire scan-chain. This allows for a simplistic bit-banging method of interfacing to the bus. Because the signals are bit-banged, speed is an issue. For every clock transition, two entire scan-chain loops

must be made. PMBus has a minimum speed of 10KHz before timeouts occur and the data is invalid. Therefore the minimum speed of clocking your entire scan-chain must be at least twice as great as 10KHz multiplied by the number of JTAG nodes in the scan-chain. This is generally not a problem because scan-chains operate in the tens of MHz range. However, if you have many devices, large devices, or a combination, the scan-chain may become so large that the minimum PMBus speed cannot be met.

In this scenario, you can split your JTAG chains to reduce the number of nodes, increase your scan-chain speed, or try Option 4B.

### Option 4B: JTAG Programming Using an External Device Connected to the PSM Products' PMBus

In the cases where you want to program via JTAG but either a) no JTAG devices are present on the system board, b) the PMBus pins are not connected to boundary scan capable hardware in the scan-chain, and/or c) the overall scan-chain speed is too slow to guarantee PMBus operation above 10KHz, you may still be able to program via JTAG and an external device.

The solution from Asset Intertech involves using their JTAG programmer with one of their generic JTAG to GPIO boards. This then acts as a JTAG node that can be connected directly to the PMBus on your board. The generic JTAG to GPIO board acts as a single device on the scan-chain with boundary scan capabilities where the same bit banging methodology is utilized.

This may be a more cost effective method to program your boards using a single JTAG programmer instead of multiple programmers for different device manufacturers.

# Application Note 145

## OPTION 5: USE LTpowerPlay TO PROGRAM LOOSE IC'S WITH A SOCKETED PROGRAMMING BOARD

The LTpowerPlay GUI can also program individual loose IC's using the DC#### Socketed Programming Board specific to your device. For example, the LTC2978 uses the DC1508. This may be an appropriate option if you are unable to physically connect the LTpowerPlay GUI to your target board (à la Option 1). If this is the case, you can program individual loose IC's before soldering them down on your board.

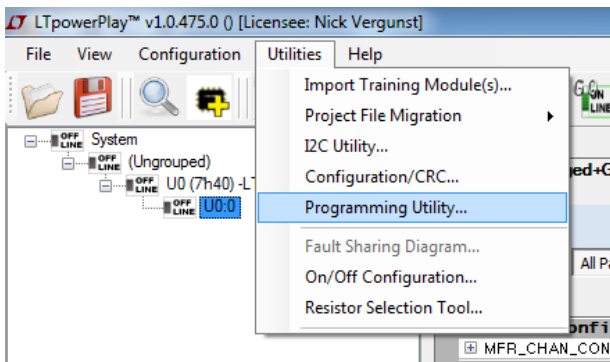
**NOTE: Option 1** is strongly preferred over this option as it allows you to change your configuration at will without de-soldering parts.

This option is identical to Option 1A: Manual Programming with LTpowerPlay Built In Programming Utility, except you will use the corresponding Socketed Programming Board instead of the end-use board.

Connect the LT USB-PMBus controller (DC1613A) to your PC via USB and to the target board using the 12-pin header from the dongle.

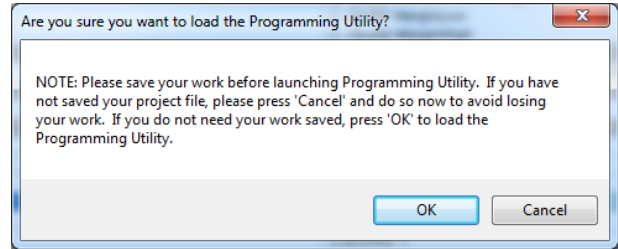
In the LTpowerPlay GUI:

- Make sure you save your current project file if you haven't done so already. The following steps will close the currently opened project file and discard any unsaved changes in order to open the project file for programming.
- Select Utilities->Programming Utility

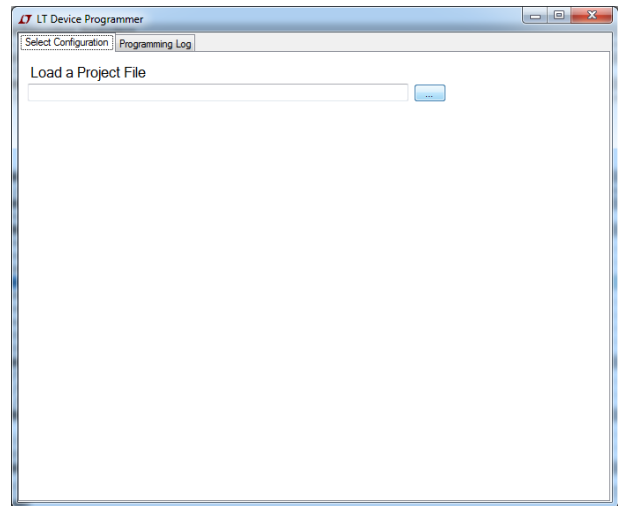


- LTpowerPlay will then ask you to confirm that you have followed the first step. If you have not saved your cur-

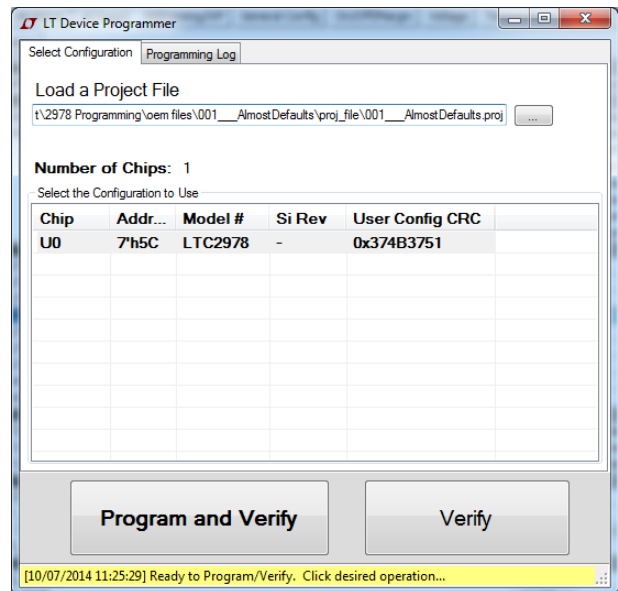
rently opened project and wish to, cancel this prompt and restart the process when ready



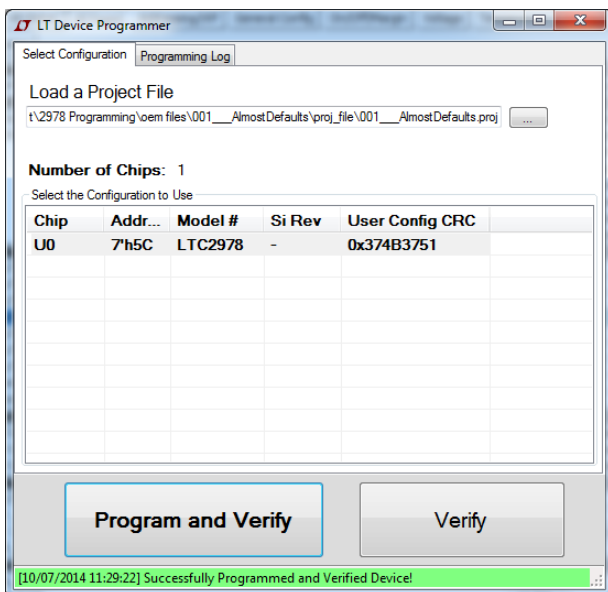
- The "LT Device Programmer" window will then pop-up. Make sure you are on the "Select Configuration" tab. Then click the "... " button to browse for a project file you wish to use as the source file to program devices



- Browse for your \*.proj file and when you select Open the LT Device Programmer window will change into this:



- You now have 2 additional large buttons near the bottom of the window.
  - The Program and Verify button will first program the device's NVM to the correct state and then verify its contents.
  - The Verify button will NOT program the device's NVM, but will attempt to verify the NVM contents with the project file's content. See Appendix B for more information.
- To program the loose device in the socket, we will select Program and Verify. The status message on the bottom will change to yellow and say "Programming" then "Verifying" and eventually when successful it will display "Successfully Programmed and Verified Device."



You can now insert a loose IC one at a time into the socket and, without reloading anything, just continue to press the Program and Verify button to program all loose devices. It is suggested that you disconnect the USB dongle from the PC before inserting, removing or adjusting the loose IC in the socket to prevent damage. Optionally, if the programming board has a power switch, you may switch power off while removing or inserting a device. The sockets are always powered and live when the dongle is connected, and the only way to ensure they are powered down is by removing all power sources, generally just the USB dongle.

It is highly recommended that you save the project file that you are using to program the devices to a separate version controlled repository so that later verification of programmed devices can be performed if ever required.

## OPTION 6: AUTHOR CUSTOM FIRMWARE

As shown in Appendix A, storing a configuration to nonvolatile memory (NVM) can be accomplished in multiple ways. The first involves writing individual registers via the appropriate I<sup>2</sup>C write commands, and finally, issuing a STORE\_USER\_ALL command to store this configuration into NVM. All PMBus devices support this implementation. The second method involves writing to the NVM directly from the I<sup>2</sup>C in a bulk programming mode which allows for the system to stay operational during the programming and will only update upon the next load of the NVM (after a Reset or RESTORE\_USER\_ALL command). Check your specific device's data sheet for full details and support of bulk programming mode.

Customers are free to implement their own firmware to communicate with the PSM devices over the I<sup>2</sup>C bus and we have sample code available. Custom firmware is considered the most difficult option because it involves an investment in authoring firmware, but once that investment is made it can be reused over the entire family of devices and products and allows you to achieve a high level of integration with the device during board bring-up, debugging, and normal operations.

If this has not scared you off, then you can take a look at some of the resources we have available for implementing in system programming and in-flight updates of configurations.

This webpage has detailed information on how to parse the LTC Programming Hex File: <http://www.linear.com/solutions/5710>.

There is another application note that has details on how to take the LTC Programming Hex file and do an actual in-flight system update with open source Linduino ([www.linear.com/Linduino](http://www.linear.com/Linduino)). Contact factory apps for more information.

# Application Note 145

## APPENDIX A: MEMORY ARCHITECTURE

The NVM compatible devices follow the standard PMBus model for configuration memory, implementing both operational registers (RAM) and nonvolatile memory. The following diagrams illustrate the configuration memory architecture without direct NVM access (Figure 6) and then with direct NVM access via bulk programming (Figure 7):

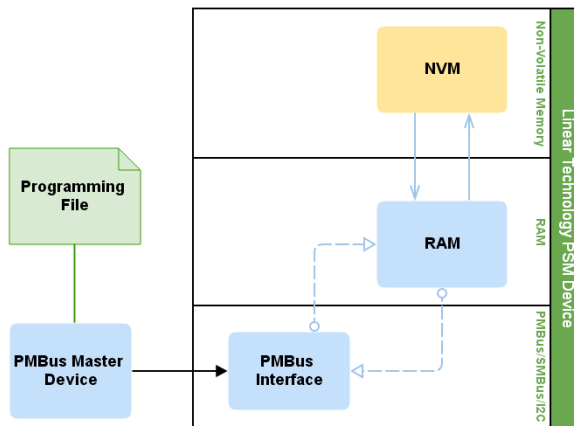


Figure 6. LTC PSM Device's Memory Architecture *Without* NVM Access

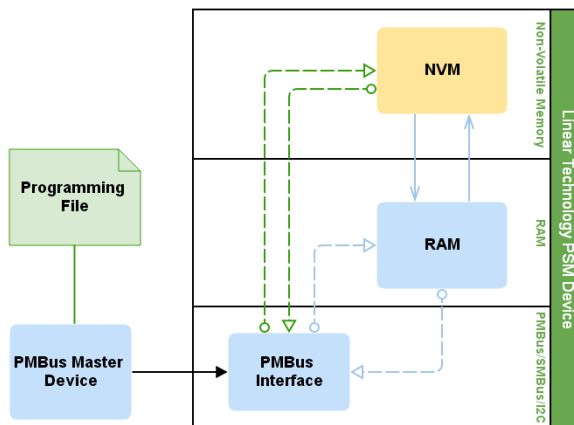


Figure 7. LTC PSM Device's Memory Architecture *With* NVM Access

The operational RAM contains a number of registers that control the operation of the chip. These operational registers can be read or modified using simple I<sup>2</sup>C/SMBus/PMBus read/write commands. Consult the "Register Command Set" table in the data sheet for a comprehensive list of these commands.

The device also contains on-board nonvolatile memory (NVM) that is used to store the desired value of these operational registers across power cycles.

Once the desired configuration has been programmed into RAM, storing the configuration to NVM is achieved with a single command called, STORE\_USER\_ALL (command code 0x15). When the chip receives the STORE\_USER\_ALL command, it copies all the registers that are stored in NVM from the operational RAM to the chip's onboard NVM for persistent storage. Consult the "Register Command Set" table in the data sheet for a full list of these registers.

Each time the device powers up (and also when a RESTORE\_USER\_ALL command is received), the device transfers the configuration values stored in its onboard NVM into its operational registers.

This architecture allows the device to power up and load the desired configuration autonomously with no I<sup>2</sup>C/firmware interaction required.

Additionally, some PSM devices support a bulk programming method that bypasses the RAM and directly programs the NVM contents. Because all the operational registers are in RAM, writing to the NVM directly is a behind-the-scenes method of programming. The changes will not appear until the RAM is reloaded from NVM, again by either a reset/power cycle or a RESTORE\_USER\_ALL command. This is useful for in-flight-updates where you do not want to bring the system down to load a new configuration.

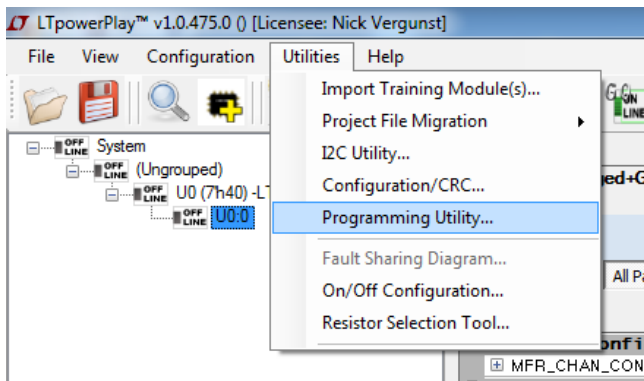
The actual method to program the RAM or NVM is dictated by the recipe inside the Programming File. More details on this can be found here: [http://www.LTpowerPlay.com/in\\_circuit\\_programming/](http://www.LTpowerPlay.com/in_circuit_programming/).

## APPENDIX B: VERIFICATION OF PROGRAMMED PARTS USING LTpowerPlay

After you program your devices in any of the number of ways above, you can then use LTpowerPlay to verify the contents of the NVM. This feature uses the same feature that Option 1A and Option 5 use for programming with the major difference being we will not “Program and Verify” the system, we will use the big standalone “Verify” button.

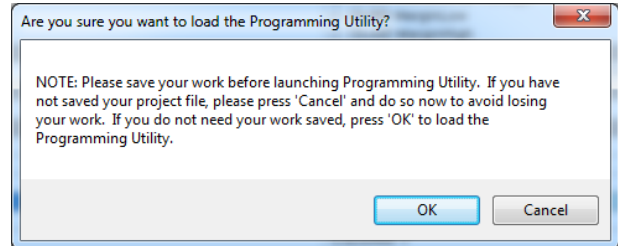
You will need a way to connect the LT USB-PMBus controller (DC1613A) to your PC via USB and to the target board. See Figure 1 and Figure 2 for a more detailed schematic of the process. If you do not have a header on your board, this is another good time to remind you that Linear Technology strongly recommends placing this header on every board. If you have loose parts, then you should use the accompanying Socketed Programming Board specific to your device.

- Retrieve the project file or programming archive file from your chosen version control repository and copy it locally so it can be accessed by LTpowerPlay
- Open LTpowerPlay and use the menu system to navigate to Utilities -> Programming Utility

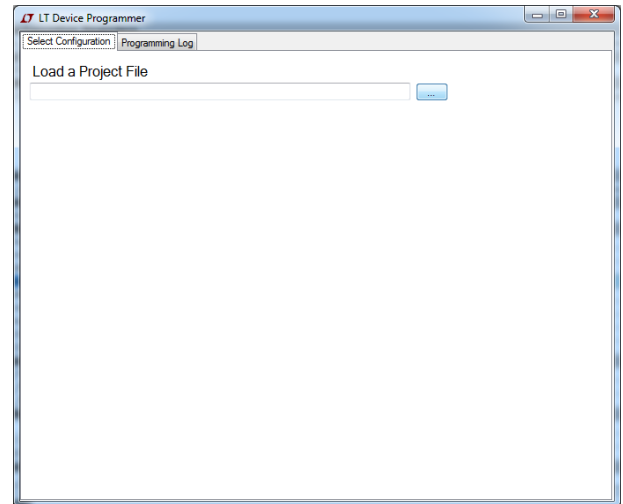


- LTpowerPlay will then ask you to confirm that you have saved your current project file if you have one and haven't saved it already. The following steps will close the currently opened project file and discard any

unsaved changes in order to open the project file for programming. If you have just launched LTpowerPlay, then you can ignore this warning. If you have not saved your currently opened project and wish to, cancel this prompt and restart the process when ready

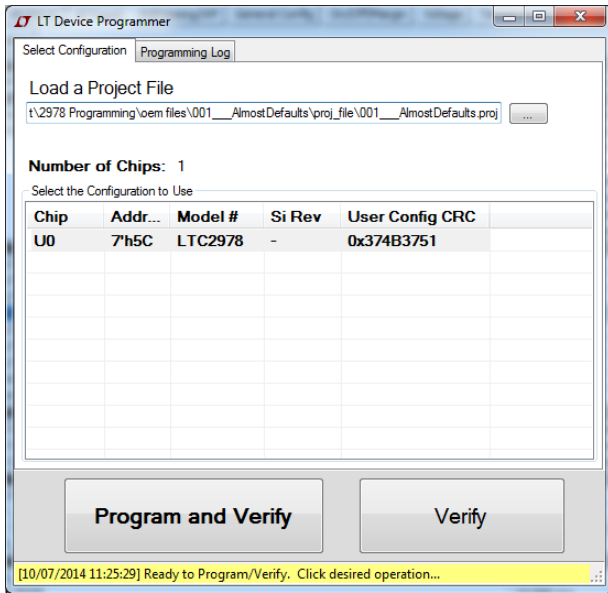


- The “LT Device Programmer” window will then pop-up. Make sure you are on the “Select Configuration” tab. Then click the “...” button to browse for a project file you wish to use as the source file to program devices. If you have saved the programming \*.zip archive, then you will need to unzip the archive to a folder. In that unzipped folder there will be a project file that corresponds with that archive. This is the file to use when prompted.

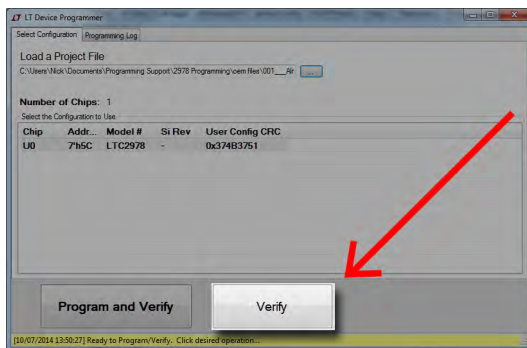


# Application Note 145

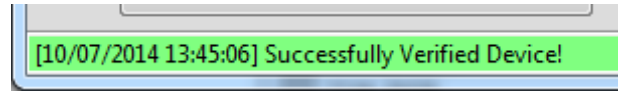
- Browse for your \*.proj file and when you select Open the LT Device Programmer window will change into this:



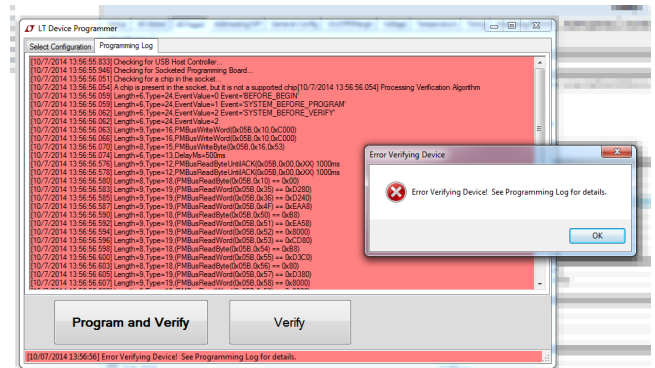
- You now have 2 additional large buttons near the bottom of the window.
  - The “Program and Verify” button will first program the device's NVM to the correct state and then verify its contents.
  - The “Verify” button will NOT program the device's NVM, but will attempt to verify the NVM contents with the project file's content.
- Because we are only verifying the contents of the Non-Volatile Memory, we will use the “Verify” button only.



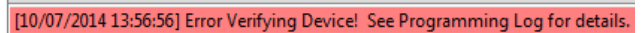
- After clicking “Verify”, LTpowerPlay will automatically and exhaustively verify all the non-volatile memory in the device against the desired values in the project file.
- When the verification is successful you will get a message that LTpowerPlay has “Successfully Verified Device!”



- If the verification is not successful, you will get an error message in a pop-up window along with a red window detailing the specifics of the failure. This text can be copied to an external file and saved for logging purposes



- There will also be a different status message in the bottom portion of the window.



Using this method is a quick and easy way of verifying that your devices have been programmed properly by any method. All Linear Technology approved methods include this verification check as an integral part of the programming process itself.

However this becomes useful to maintain peace of mind for the engineer or to validate that third party programming and contract manufacturing houses have delivered what you have ordered and more importantly put the correct part in the correct tube.