用於電源系統管理的 Linduino（下）

大多數電源系統管理設計都遵循一種“設定後便不需再過問”的模型。電源系統管理 (PSM) 件的設定和調試利用 LTpowerPlay 是簡單易行的，而且當與一個量化程式設計解決方案組合時則無需韌體。不過，許多大型系統需要一個電路板管理控制器 (BMC)，因而提出了這樣的問題：“韌體能夠為 PSM 做什麼呢？”

■作者：Michael Jones / 凌力爾特應用工程經理

編寫一個簡單的 Sketch

最佳的學習方法是從頭編寫代碼，本節所涉及的內容即在於此。如果您自己執行這些步驟 (一次一個) 並在逐步執行的過程中驗證結果，那麼下面的示例將是最有幫助的。

該 Sketch 示例採用一個 DC1962 和在第一節中提到的另一個硬體。此示例接受 5 個簡單的命令：

1. 列印電壓
2. 裕度調節
3. 接通 / 關斷
4. 匯流排探測
5. 復位

第一步：創建一個空白的 Sketch

如圖 14(新的 Sketch) 所示，透過選擇 File|New 以創建一個新的 Sketch。然後，您將看到一個如圖 15(空的 Sketch) 所示的空 Sketch。

File|Save As…功能表，並為新的 Sketch 選擇一條路徑。要確保資料夾名稱和 Sketch 名稱匹配，如圖 16(Save As. . . .) 所示。該路徑必須在 LTSketchbook 的下方 (File|Preferences 對話方塊中的相同路徑) 以便其在 Sketchbook 功能表中顯現出來。

圖 14：新的 Sketch (New Sketch)

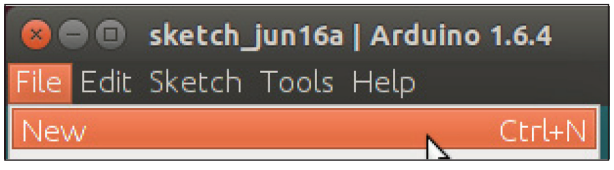
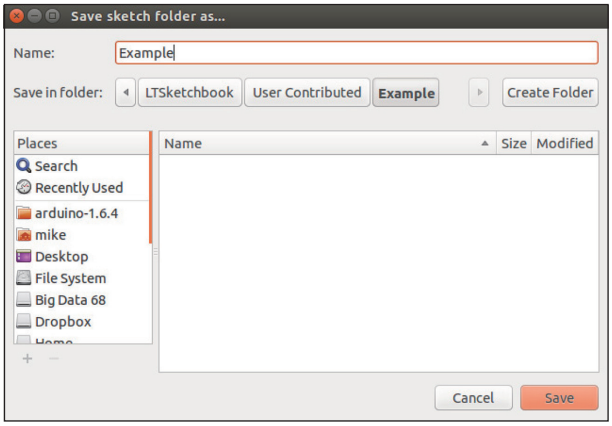


圖 15：空的 Sketch (Empty Sketch)

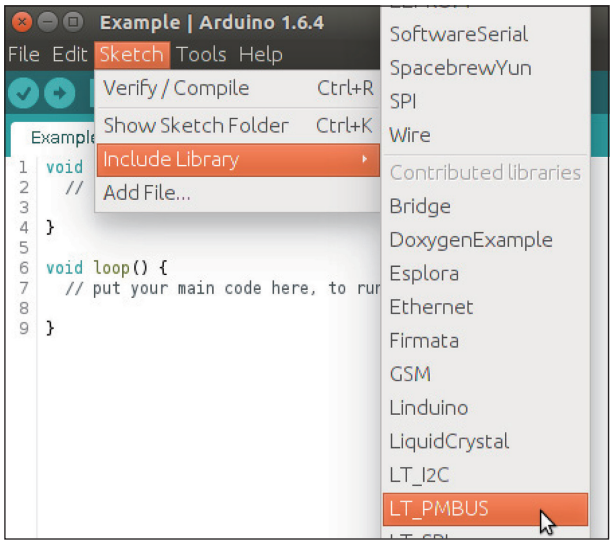
圖 16：Save As … (另存為 …)



第二步：添加包括

Sketch|Include|Include Library 功能表，選擇

圖 17：包括 (Includes)



下面的庫 (一次一個庫)：

- UserInterface
- Linduino
- LT_PMBUS

圖 17(Includes) 所示為如何選擇 LT_PMBUS 庫。所有的庫都在同一個 Include Library 選單中。

在檔的頭一行上添加該包含語句：

#include <Arduino.h>

圖 18：初始化代碼

```
Example
#include <Arduino.h>
#include <UserInterface.h>
#include <Linduino.h>
#include <LT_Wire.h>
#include <LT_twi.h>
#include <LT_SMBusPec.h>
#include <LT_PMBusMath.h>
#include <LT_SMBus.h>
#include <LT_I2CBus.h>
#include <LT_SMBusGroup.h>
#include <LT_PMBus.h>
#include <LT_FaultLog.h>
#include <LT_SMBusNoPec.h>
#include <LT_SMBusBase.h>

#define LTC3880_I2C_ADDRESS 0x30
#define LTC2974_I2C_ADDRESS 0x32
#define LTC2977_I2C_ADDRESS 0x33

static uint8_t ltc3880_i2c_address;
static uint8_t ltc2974_i2c_address;
static uint8_t ltc2977_i2c_address;
static LT_SMBus *smbus = new LT_SMBusNoPec();
static LT_PMBus *pmbus = new LT_PMBus(smbus);

void setup() {
  Serial.begin(115200);
  ltc3880_i2c_address = LTC3880_I2C_ADDRESS;
  ltc2974_i2c_address = LTC2974_I2C_ADDRESS;
  ltc2977_i2c_address = LTC2977_I2C_ADDRESS;
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

現在，添加用於位址和 SMBus/PMBus 物件的靜態變數。增添設置 (Setup) 代碼以預置變數和串列匯流排物件。利用 File|Save 保存該代碼。最後，撳按工具列上的檢查按鈕以編譯代碼。

您的代碼應看似圖 18(初始化代碼) 的樣子。

第三步：設置選單

一個功能表需要列印選擇，以及對用戶選擇的回應。

添加一個 print_prompt( ) 函數以列印一個提示，並從設置函式呼叫它以在 Sketch 運行時列印功能表提示。代碼應看起來像圖 19 (Prompt)。

保存和編譯以確保代碼沒有錯誤。

圖 19：Prompt

```
void print_prompt()
{
  Serial.print(F("\n  1-Print Voltages\n"));
  Serial.print(F("  2-Margin High\n"));
  Serial.print(F("  3-Margin Low\n"));
  Serial.print(F("  4-No Margin\n"));
  Serial.print(F("  5-Go Off\n"));
  Serial.print(F("  6-Go On\n"));
  Serial.print(F("  7-Probe Bus\n"));
  Serial.print(F("  8-Reset\n"));
  Serial.print(F("\nEnter a command:"));
}

void setup() {
  Serial.begin(115200);
  ltc3880_i2c_address = LTC3880_I2C_ADDRESS;
  ltc2974_i2c_address = LTC2974_I2C_ADDRESS;
  ltc2977_i2c_address = LTC2977_I2C_ADDRESS;
  print_prompt();
}
```

第四步：添加選單回應

當一個功能表選項被鍵入控制台時，代碼程式必須讀取它並做出回應。

環路函數將處理使用者輸入。首先，它必須檢查串列匯流排是可用的。然後，它必須把輸入作為一個整數讀取並將之傳遞至一個開關語句。在開關語句的內部，它必須執行某些函數並隨後調用提示函數。用於每個命令的代碼將在開關語句每種情況的內部編寫，而提示函數將在之後調用。您的代碼框架應看上去與圖 20(用戶輸入) 相似。

對它進行保存和編譯以確保其沒有錯誤。

第五步：讀取電壓

現在是編寫可做些有用工作的實際 PMBus 代碼的時候了。

圖 21(讀取電壓) 顯示負責讀取所有電壓的代碼。代碼在情況 1 的內部示出。兩個變數保持電壓和頁面：一個用於電壓的 float，和一個用於頁面的 unit8_t，顯示在第 57~58 行。列印使用標準的 Arduino Serial.print…函數。字串周圍的 F() 把它們放入快閃記憶體中，這樣它們就不會使用寶貴的 RAM。對於每個元件，一個 for loop 透過調用

圖 20：用戶輸入

```
47  void loop() {
48    uint8_t user_command;
49
50    if (Serial.available())
51    {
52      user_command = read_int();
53
54      switch (user_command)
55      {
56        case 1:
57          break;
58        case 2:
59          break;
60        case 3:
61          break;
62        case 4:
63          break;
64        case 5:
65          break;
66        case 6:
67          break;
68        case 7:
69          break;
70        case 8:
71          break;
72      }
73      print_prompt();
74    }
75  }
```

圖 21：讀取電壓

```
54      switch (user_command)
55      {
56        case 1:
57          float   voltage;
58          uint8_t page;
59
60          Serial.println(F(""));
61          for (page = 0; page < 2; page++)
62          {
63            pmbus->setPage(ltc3880_i2c_address, page);
64            voltage = pmbus->readVout(ltc3880_i2c_address, false);
65            Serial.print(F("LTC3880 VOUT "));
66            Serial.println(voltage, DEC);
67          }
68
69          for (page = 0; page < 4; page++)
70          {
71            pmbus->setPage(ltc2974_i2c_address, page);
72            voltage = pmbus->readVout(ltc2974_i2c_address, false);
73            Serial.print(F("LTC2974 VOUT "));
74            Serial.println(voltage, DEC);
75          }
76
77          for (page = 0; page < 8; page++)
78          {
79            pmbus->setPage(ltc2977_i2c_address, page);
80            voltage = pmbus->readVout(ltc2977_i2c_address, false);
81            Serial.print(F("LTC2977 VOUT "));
82            Serial.println(voltage, DEC);
83          }
84          break;
85        case 2:
86          break;
```

pmbus->setPage(…) 的頁面進行檢索，隨後利用 pmbus->readVout(…) 以讀取電壓。接著，代碼以十進位列印電壓 (採用 DEC 類型)。

您可以找到自己在 PMBus.h 檔中的 LT_PMBus 庫或 Doxigen 文檔裡使用的所有 API 函式宣告。

第六步：裕度調節和接通 / 關斷

裕度調節代碼比電壓代碼簡單，因為操作是全域的，這意味著所有的元件能夠回應一個命令，而

圖 22：裕度調節和接通 / 關斷

```
85          case 2:
86            pmbus->marginHighGlobal();
87            break;
88          case 3:
89            pmbus->marginLowGlobal();
90            break;
91          case 4:
92            pmbus->sequenceOnGlobal();
93            break;
94          case 5:
95            pmbus->sequenceOffGlobal();
96            break;
97          case 6:
98            pmbus->sequenceOnGlobal();
99            break;
```

且不需要頁面寄存器。此外，沒有任何東西要列印。

圖 22(裕度調節和接通 / 關斷) 顯示代碼。情況 4 是 No Margin 功能表選擇。採用 sequenceOnGlobal() 來結束裕度調節也許看起來很奇怪。在內部結構中用於這些操作的 PMMBus 命令是 OPERATION (0x01) 命令。

取自 LTC3880 產品手冊的圖 23(OPERATION 命令) 表明：沒有用於停止裕度調節的專用命令。裕度調節利用數值 0x80(它意味著接通) 來關斷。這就是採用 pmbus->sequenceOnGlobal() 來關斷裕度調節操作的原因。

當 On-Off_Config_Use_PMBus 使能 Operation_Control 時的

OPERATION 命令細節寄存器 OPERATION 資料內容

圖 23：OPERATION 命令

符號	動作	數值
位		
函數	立即關斷	0x00
	接通	0x80
	裕度調節低	0x98
	裕度調節高	0xA8
	序列關閉	0x40

第七步：匯流排探測和重定

探測匯流排是 SMBus API 的一部分，畢竟並非所有的元件均為 PMBus。圖 24(探測和復位) 所示：探測是對 smbus->probe(0) 的調用。0 是它用以實施探測的命令，其為 PAGE(0x00) 命令。探測將測試所有的有效位址並回送一個所發現之元件的清單。它將找到所有能夠確定收到一個讀命令 0x00 的元件。

復位命令不太明顯。LTC388X 和 LTC297X 系列的重定方法不同。LTC388X 元件支援一個 MFR_RESET (0xFD) 命令，但是 LTC297X 元件則不支援。例如，在 LTC2977 上 0xFD 命令為 MFR_TEMPERATURE_MIN，而不是 MFR_RESET。使一個管理器復位的正確方法是從非易失性記憶體 (NVM) 恢復 RAM，因為在該變換之後元件復位。

然而，如欲使所有的元件同時復位，則使用群組命令協定。這把所有的操作編組為單個事務，在

圖 24：探測和復位

```
100       case 7:
101         uint8_t *addresses;
102         addresses = smbus->probe(0);
103         while(*addresses != 0)
104         {
105           Serial.print(F("ADDR 0x"));
106           Serial.println(*addresses++, HEX);
107         }
108         break;
109       case 8:
110         pmbus->startGroupProtocol();
111         pmbus->reset(ltc3880_i2c_address);
112         pmbus->restoreFromNvm(ltc2974_i2c_address);
113         pmbus->restoreFromNvm(ltc2977_i2c_address);
114         pmbus->executeGroupProtocol();
115         break;
```

這裡所有的命令均由 PSM 元件在 STOP 執行。

圖 24(探測和復位) 中的 case 8 顯示怎樣設置一個群組協定事務。該事務受 pmbus->startGroupprotocol( ) 和 pmbus->executeGroupProtocol( ) 調用的束縛。

第八步：測試

這是編譯和運行應用程式並確信其工作狀況一切正常的好時機。

假如應用程式運行，但並不列印合理的資料，那麼您或許犯了某種錯誤。您可以使用下面的調試方法對其進行調試。或者，如果您沒有耐心，則可只進行複查：

■地址

■頁面

■中斷語句

調試

有幾種調試一個 Linduino PSM 應用程式、或在這方面的任何韌體應用程式的方法：

■列印

■ Spy 工具

■調試器

本應用指南將不追求第三種選項。對於簡單的 Sketch 它通常不是必需的。如果您希望瞭解有關調試器的更多資訊，則轉至 Arduino 網站論壇，看看其他人使用的是什麼工具。

您已經看見了在上面的實例中使用的列印。您

可透過添加更多的列印語句以進行調試。不過，應始終把字串放在 F() 巨集指令的內部，這樣 RAM 就不會被耗盡。當列印文本和數位時將其分成兩個調用，於是文本部分便在快閃記憶體中了。

PSM 庫採用這種方法。誤差 (比如：NACK 和 PEC 誤差) 在命令視窗中列印。因此，增添調試列印通常被限制在應用程式碼。

您已經看見了採用 DEC 的列印。您也可以使用 HEX 和其他格式。查閱 Arduino 文檔以獲得有關格式化的更多幫助。

圖 25：Beagle 追蹤

Index	m:s.ms.us	Dur	Len	Err	S/P	Addr	Record	Data
0	0:00.000.000						Capture start...	[Tue 16 Jun
1	0:08.432.357	305 us	2 B		SP	30	Write Transac...	00 00
2	0:08.432.702	209 us	1 B		S	30	Write Transac...	8B
3	0:08.432.912	302 us	2 B		SP	30	Read Transac...	9A 0D*
4	0:08.433.249	209 us	1 B		S	30	Write Transac...	20
5	0:08.433.459	207 us	1 B		SP	30	Read Transac...	14*
6	0:08.435.261	305 us	2 B		SP	30	Write Transac...	00 01
7	0:08.435.616	209 us	1 B		S	30	Write Transac...	8B
8	0:08.435.825	309 us	2 B		SP	30	Read Transac...	9A 11*
9	0:08.436.170	209 us	1 B		S	30	Write Transac...	20
10	0:08.436.380	207 us	1 B		SP	30	Read Transac...	14*
11	0:08.438.191	305 us	2 B		SP	32	Write Transac...	00 00
12	0:08.438.541	210 us	1 B		S	32	Write Transac...	8B
13	0:08.438.752	310 us	2 B		SP	32	Read Transac...	FC 2F*
14	0:08.439.096	209 us	1 B		S	32	Write Transac...	20
15	0:08.439.306	207 us	1 B		SP	32	Read Transac...	13*
16	0:08.441.132	305 us	2 B		SP	32	Write Transac...	00 01
17	0:08.441.478	215 us	1 B		S	32	Write Transac...	8B
18	0:08.441.693	302 us	2 B		SP	32	Read Transac...	9B 39*
19	0:08.442.031	209 us	1 B		S	32	Write Transac...	20
20	0:08.442.240	207 us	1 B		SP	32	Read Transac...	13*
21	0:08.444.047	305 us	2 B		SP	32	Write Transac...	00 02
22	0:08.444.397	209 us	1 B		S	32	Write Transac...	8B
23	0:08.444.606	310 us	2 B		SP	32	Read Transac...	02 40*
24	0:08.444.951	209 us	1 B		S	32	Write Transac...	20
25	0:08.445.161	207 us	1 B		SP	32	Read Transac...	13*
26	0:08.446.967	305 us	2 B		SP	32	Write Transac...	00 03
27	0:08.447.322	209 us	1 B		S	32	Write Transac...	8B
28	0:08.447.532	310 us	2 B		SP	32	Read Transac...	65 46*
29	0:08.447.877	209 us	1 B		S	32	Write Transac...	20
30	0:08.448.086	207 us	1 B		SP	32	Read Transac...	13*
31	0:08.449.903	305 us	2 B		SP	33	Write Transac...	00 00
32	0:08.450.258	214 us	1 B		S	33	Write Transac...	8B
33	0:08.450.473	302 us	2 B		SP	33	Read Transac...	CD 1C*
34	0:08.450.810	209 us	1 B		S	33	Write Transac...	20
35	0:08.451.020	212 us	1 B		SP	33	Read Transac...	13*
36	0:08.452.816	305 us	2 B		SP	33	Write Transac...	00 01
37	0:08.453.171	209 us	1 B		S	33	Write Transac...	8B
38	0:08.453.381	302 us	2 B		SP	33	Read Transac...	01 20*
39	0:08.453.718	209 us	1 B		S	33	Write Transac...	20
40	0:08.453.928	207 us	1 B		SP	33	Read Transac...	13*
41	0:08.455.725	305 us	2 B		SP	33	Write Transac...	00 02
42	0:08.456.075	209 us	1 B		S	33	Write Transac...	8B

用於 PMBus 的最好調試器是 Spy 工具。Spy 工具之所以很好，原因是您能看見匯流排上的通訊量，而且在需要幫助的時候您可以把一個追蹤記錄與您的代碼一起發送給 LTC 應用技術工程師。

本應用指南將重點關注由與 Total Phase Beagle 對話的 Total Phase Data Center 應用程式產生的資料。在 Total Phase 的網站上提供了說明您安裝 Data Center Application 的資訊 (www.totalphase.com)。

啓動開發工作最簡單的方法是使用您剛剛創建的 Sketch 來追蹤匯流排。將採用功能表選擇 3(讀取電壓)。

圖 25(Beagle 追蹤) 顯示資料。讓我們直接投入並對某些事務進行解密，並採用索引以記錄我們所在的位置。

在 Index #1(I1) 和 index #6(I6)，有兩項寫位元組事務。在 SMBus 中，這是寫位元組協定 (Write Byte Protocol)。位址為 0x30，這是 LTC3880，如在代碼中可以看到的那樣。第一個位元組是命令，其為 0x00，它是頁面 (PAGE) 命令。

表 2 中的 LTC3880 產品手冊示出了 PAGE 命令。這張表是對 Beagle 資料進行解碼的快速方法。請注意 Type 列顯示的是 R/W Byte。這意味著暫存器是讀 / 寫位元組協定，因此兩個方向均得到支援。

回過頭去看一下 I1 和 I6，第二個位元組是一個 0x00 和 0x01。代碼將把 PAGE 寄存器設定為 0 和 1。重新提及圖 21(讀取電壓) 中的第 63 行，這是設定頁面的地方。我們應查看在緊接此之後的 I2 至 I5 上處於讀取之中的電壓。

我們看到：

寫 0x8B，讀 0x9A 0x0D

寫 0x20，讀 0x14

0x8B 是一個 READ_VOUT 命令。產品手冊中的表 2 顯示它是一個 R 字協議，而且採用的是 L16 格式。

0x20 是一個 VOUT_MODE 命令。產品手冊中的表 2 顯示它是一個 R 位元組命令。

導致這些發生的 Linduino 調用示於圖 21(讀取電壓) 中的第 64 行。

為什麼單個函式呼叫發佈了兩項事務？為弄清原因，我們必須瞭解 API 背後的代碼，示於圖 27(讀取 VOUT 代碼)。

該代碼示出了一個後隨 smbus_.readByte(address, VOUT_MODE) 的 smbus_.readWord(address, READ_VOUT)，而且 5 個位從模式抽取並被置入變數 exp。數學轉換然後採用指數 exp 實現從 L16 至浮點的轉換。

大致說來，用於讀取電壓的代碼是負責讀取用於把 L16 轉換為浮點之指數的通用代碼。LTC388X 和 LTC297X 元件採用了一個不同的指數。這就是存在兩項事務的原因。

注：代碼可利用指數的某種先驗知識來編寫，而且它的運行速度快一點。不過，通用代碼將具有較少的漏洞，且對於應用程式編寫者更加容易。當編寫自己的代碼時這是您必須做出的權衡。

在結束之前，讓我們看一個更加有趣的事務：復位。

看一下圖 28(復位追蹤)，並需注意到有三個寫事務。在 S/P 列中，有兩個 S 和一個 SP。這意味著 I1 是一個啓動 (Start)，I2 是一個重複啓動

圖 27：讀取 VOUT 代碼

```
VOUT_L16 = SMBUS_.READWORD(ADDRESS, READ_VOUT);
EXP = (INT8_T)

  (SMBUS_.READBYTE(ADDRESS, VOUT_MODE) & 0X1F);
RETURN MATH_.LIN16_TO_FLOAT(VOUT_L16, EXP);
```

圖 26：PAGE 命令

Table 2. Summary (Note: The Data Format abbreviations are detailed at the end of this table.)

COMMAND NAME	CMD CODE	DESCRIPTION	TYPE	PAGED	DATA FORMAT	UNITS	NVM	DEFAULT VALUE	PAGE
PAGE	0x00	Channel or page currently selected for any command that supports paging.	R/W Byte	N	Reg			0x00	63

圖 28：復位追蹤 (Reset Trace)

Index	m:s.ms.us	Dur	Len	Err	S/P	Addr	Record	Data
0	0:00.000.000						Capture start...	[Tue 1
1	0:04.867.488	209 us	1 B		S	30	Write Transac...	FD
2	0:04.867.698	244 us	1 B		S	32	Write Transac...	16
3	0:04.867.942	250 us	1 B		SP	33	Write Transac...	16
4	0:08.899.681						Capture stop...	[Tue 1

(Repeated Start)，I3 是一個重複啓動 (Repeated Start)，之後是一個停止 (Stop)。此外，它們各自的地址是不同的：0x30、0x32 和 0x33。

這是群組命令協議。所有的命令都將在 I3 的末端 (STOP) 處理。這與圖 24(探測和復位) 中所示代碼的第 110~114 行相關聯。

如果您在自己的 Beagle 軌跡解碼上花些時間，就將對 PSM 元件的 PMBus 命令獲得更加完整的瞭解。另一方面，假如您的目標是使代碼正常運行，那麼 PSM 庫非常願意為您完成繁重的任務。

採用 LTpowerPlay 的高級調試

回到引言部分，那裡說大多數系統是“設定後便不需再過問”的，而且有些系統具有一個 BMC。事實是那些具有一個 BMC 的系統把“設定後便不需再過問”與韌體組合在一起。為什麼讓一個 BMC 承擔了全部的設置責任呢？這是因為把某種基本設定寫入 PSM 元件、然後僅將 BMC 韌體用於添加的價值功能要容易得多。另外，這還造就了更加可靠的系統，因為大多數韌體負責讀取遙測資料、裕度、並實施輕微的電壓變更，不需要讓它控制排序或始終是靜態的 PWM 頻率等關鍵功能。

由於 LTpowerPlay 是一款用於設計、調試和調通 PSM 系統的通用型工具，因此調試韌體必須與物理時脈和資料線上的另一個 PMBus 主控器進行競爭。

在進入兩個主控器的實際意義之前，最好是仔細研究一下當一根 PMBus 具有兩個主控器時會發生什麼。PMBus 基於 SMBus，這包括多主控。

時脈和資料線是開路漏極。這意味著任何元件 (主控器或受控器) 都能夠下拉一根線的位準，但不能上拉其位準。有這樣一條規則：當一個主控器未下拉資料線電平、並檢測到資料線為低位準時，則其認為有另一個主控器在把資料線拉至低位準，於是中止其自身的事務，從而允許另一個主控器繼續完成其事務。

該方法有時被稱為“位優勢仲裁”(bit dominance arbitration)，這是一種指明“在資料中將一個零置為有效的主控器總是獲勝”的奇特說法。

Linduino 和 LTpowerPlay(DC1613) 多主控器，而且您也許認為一切都很完美了。不過，還有一項更為關鍵的考慮因素。

PMBus 定義了一個 PAGE 命令 (0x00)，它像一個進入資料的位址。頁面像是通道一樣。例如，一個 LTC2977 能夠管理 8 個電源：它擁有 8 個通道，各由 PAGE 暫存器 / 命令進行定址。

實際上這意味著：為讀取某種數值 (如電壓)，它需要兩項事務：一項用於 PGAE，一項用於 READ_VOUT。如果兩個主控器同時試圖從同一個受控器讀取遙測資料，而且倘若一個主控器在另一個主控器的頁面命令和遙測命令之間插入了一個頁面命令，則它將讀取錯誤的頁面。

當 LTpowerPlay 啓動並運行時，它所做的首要之事是讀取遙測資料，這將保持其狀態顯示為最新，於是您就能查看輸出的曲線圖、故障和其他重要資訊。猜猜韌體通常做什麼？它讀取遙測資料！

更糟糕的是，假設韌體在啓動的時候執行一項 VID(電壓識別) 功能。假使韌體由於 LTpowerPlay 修改了 PAGE 寄存器而把一個電壓值寫至錯誤的頁面，將會怎麼樣呢？系統有可能發生故障關斷，甚至更糟地造成某些東西受損。(幸運的是，VOUT_MAX 寄存器通常可防止系統損壞)

PAGE 命令的基本問題是 PMBus 規格中固有的。它並不是 LTC 電源系統管理元件所特有的，而且必須對其進行處置。

有兩種允許 LTpowerPlay 與韌體共存並避免發生 PAGE 問題的基本方法。第一種方法就是簡單地不讓兩個主控器同時談話。第二種方法是在一個或另一個主控器上使用 PAGE PLUS 協議和其他的技巧。

我們避開 PAGE PLUS，因為它不是常用的。PAGE PLUS 提供了一種原子事務，其在一項事務中包括 PAGE 和 COMMAND。由於並非所有的元件都支援它，所以它通常僅在特殊場合中使用，因此本應用指南將不把重點放在 PAGE PLUS 和其他訣竅上。如果您沒有解決此問題的其他方法，則可閱讀 LTC PSM 產品手冊，或致電當地的應用技術工程師以尋求幫助。

更常見的是防止 LTpowerPlay 和韌體同時與受控器對話。LTpowerPlay 具有一種非常簡單控制其運行的方法。圖 29(LTpowerPlay 啓動 / 停止) 所示為遙測曲線圖，它在其工具列上具有一個紅色方形按鈕。

當按下此按鈕時，它會停止匯流排上的所有遙測和所有的 LTpowerPlay 動作。然後，它變為一個

圖 29：LTpowerPlay 啓動 / 停止

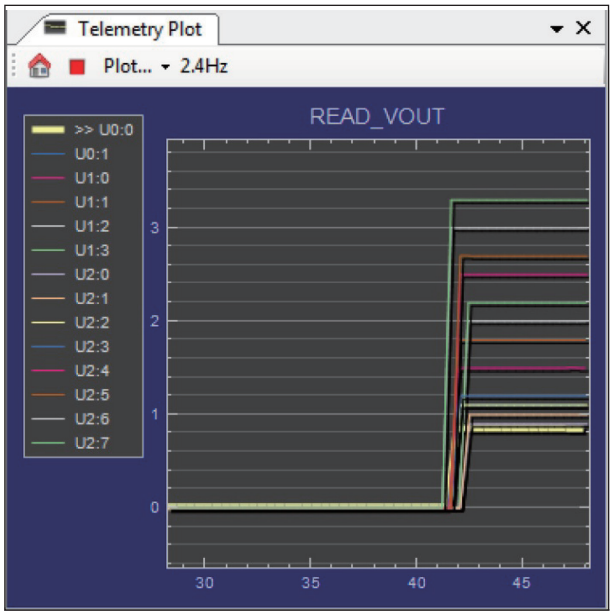
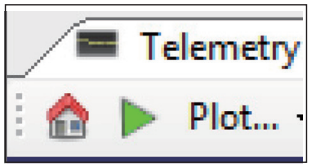


圖 30：停止的 LTpowerPlay



綠色箭頭，如圖 30(停止的 LTpowerPlay) 顯示。

不幸的是，韌體並非總是具有一種用於實現匯流排靜噪的內置機制。LTC 簡單地建議利用一種內置靜噪機制、或者透過提供一個硬體匯流排開關 / 多工器 (MUX) 或跳線來設計新型韌體。

一旦有了一種使匯流排主控器、LTpowerPlay 或韌體靜噪的方法，那麼調試只是一件根據需要在工具之間進行交替的簡單事。

總之，有兩種選擇：

1. 一次只允許一個主控器在匯流排上進行對話
2. 在 LTC 應用技術工程師的幫助之下粗略地瞭解 PAGE PLUS 及其他先進方法的細節

對於一款新設計而言，選項 1 始終是最佳的選擇。

概要

利用 Linduino PSM Sketchbook、一塊 Linduino 開發板和任選的 DC2294 遮罩，可使針對 PSM 元件的程式碼編寫變得簡單。軟體庫具有一個用於 SMBus 和 PMBus 的簡單 API。LTpowerPlay 仍可用於調試，而且可附接一個 Total Phase Beagle 或其他的 Spy Tool 以觀察匯流排上的通訊量。

不管您是希望移植代碼還是僅僅希望瞭解 PMBus 的工作原理，Linduino 都是啓動開發工作的一種絕佳的方法。一旦您加快了學習曲線，並瞭解了 PSM 程式設計的基礎知識，就能提升自己的工作進度並充滿信心地使用其他工具。

現在完成了本應用指南的閱讀，您或許希望察看與 LTSketchbook 配套提供的其他 Sketch。試用一些更高級的 Sketch，比如：故障記錄解碼 (Fault Log Decoding) 或系統內 / 運行中更新 (In System/ Flight Update)。CTA