2017年1月

Linduino PSMを使用したフォルト・ログのデコード

Michael Jones

はじめに

LTCのパワーシステム・マネージメント・デバイスは、PMBus制御のPOL（Point-of-Load）コンバータ（LTC®388X）とパワーシステム・マネージャ（LTC297X）です。LTCの全てのパワーシステム・マネージメント（PSM）デバイスは、フォルト発生時にEEPROMに書き込まれるフォルト・ログを備えています。フォルト・ログが一度書き込まれると、（通常はそのフォルト・ログ・データが読み出された後に）再有効化されるまで、生成されなくなります。

LTpowerPlay®は、フォルト・ログの読み出し、人が判読可能な形式へのデコード、表示、ファイルへの保存により、システムのデバッグを支援します。ボード・マネージメント・コントローラ（BMC）を搭載したシステムは、EEPROMからフォルト・ログを読み出し、フォルト・ログを再有効化できます。ファームウェアは、PSMデバイスから読み出されたフォルト・データの格納、ネットワークを介した転送、デコードを実行できます。

Linduino® Sketchbookには、フォルト・ログを読み出すサンプル・スケッチと、PMBusおよびフォルト・ログのデコード用のサポート・ライブラリが含まれています。

このアプリケーション・ノートでは、Linduinoを使用したフォルト・ログの読み出しとデコードの基礎知識を説明します。

ハードウェア

このアプリケーション・ノートで使用するハードウェアは次のとおりです。

1. DC2026（Linduino）
2. DC2294（シールド）
3. DC1962（電源スティック）

ソフトウェア

このアプリケーション・ノートで使用するソフトウェアは次のとおりです。

1. Arduino 1.6.6
2. LTSketchbook

このアプリケーション・ノートが最も役に立つのは、ハードウェアをLTCから入手し、www.linear-tech.co.jp/linduinoからダウンロードしたLTSketchbookを使用して結果を複製する場合です。このアプリケーション・ノートを読むだけで、必要な知識を習得し、フォルト・ログの仕組みを理解できます。

フォルト・ログの基礎知識

LTCのPSMデバイスは、フォルト発生前のテレメトリ・データを常にRAMに格納しています。フォルトが発生すると、デバイスはRAMのデータをEEPROMに転送します。このフォルト・ログ・データには、フォルト発生前後のテレメトリが含まれます。

PSMデバイスは、テレメトリ・ループを使用して、循環RAMバッファにデータを格納します。ステート・マシンは、バッファの末尾に達すると、ラップアラウンドして書き込みを続けます。フォルトが発生すると、デバイスはもう少しデータを格納して現在のテレメトリ・ループを終了し、循環バッファをEEPROMに書き込みます。デバイスは、電源が失われるまで、バッファ全体をEEPROMに書き込もうとします。

フォルト・ログ・データには、以下のタイプの情報が含まれます。

1. フォルト・ログの原因
2. フォルト発生時のテレメトリMUXの位置
3. クロック・チック値
4. 電圧、電流、温度のピーク値
5. 複数のテレメトリ・ループ

フォルト・ログ・データがボード・マネージメント・コントローラ（BMC）によってデバイスから読み出されると、デバイスはデータのブロックを返します。各PSMデバイスのデータシートには、このデータの説明が記載されています。

http://www.linear-tech.co.jp/products/Digital_Power_System_Management

データの読み出し

PSMフォルト・ログに関連するPSM PMBusコマンドには次のものがあります。

- MFR_FAULT_LOG_STORE (0xEA)
- MFR_FAULT_LOG_CLEAR (0xEC)
- MFR_FAULT_LOG (0xEE)

LTC297xデバイスは次のコマンドも実装しています。

- MFR_FAULT_LOG_RESTORE (0xEB)
- MFR_FAULT_LOG_STATUS (0xED)

PSMコントローラ(LTC388Xおよび完全に統合されたµModules®)は、EEPROMからの直接の読み出しをサポートしています。PSM マネージャ(LTC297X)は、MFR_FAULT_LOG_RESTOREを使用してフォルト・ログをRAMに転送します。このデータはMFR_FAULT_LOGを使用して読み出されます。図1のフォルト・ログ・メモリは、LTC2977のメモリの構成を示しています。

MFR_FAULT_LOG_STOREコマンドは、LTC388XおよびLTC297Xの両ファミリにフォルト・ログを強制します。このコマンドはLinduinoを使用したテストに便利です。MFR_FAULT_LOGコマンドは、ブロック読み出しプロトコルを使用してフォルト・ログの内容を返します。MFR_FAULT_LOG_STOREコマンドは、バイト送信プロトコルを使用します。これはデータ・バイトなしのコマンドです。

LTpowerPlayとLinduinoは、フォルト・ログ管理用に共通のコマンドとPMBusプロトコルを使用します。

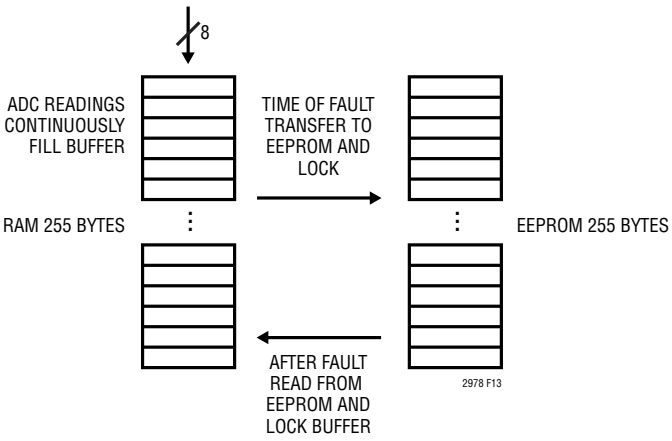

図1. フォルト・ログ・メモリ

データのデコード

LTC3880のフォルト・ログ・データは次の2つのエンティティで構成されます。

1. ヘッダ情報
2. 巡回データ

ヘッダには次の情報が含まれます。

1. フォルトの原因
2. フォルト発生時の内部時間値
3. ピーク・テレメトリ値

巡回データには次のデータが含まれます。

1. VOUT
2. IOUT
3. VIN
4. IIN
5. STATUS VOUT
6. STATUS WORD
7. STATUS MFR SPECIFIC

巡回データは5ブロックからなり、先頭が最新のデータ、末尾が最も古いデータになります。

表1. LTC3880のフォルト・ログ・ヘッダ

ヘッダ情報			
Position_Fault		BYTE	0
MFR_REAL_TIME	[7:0}	BYTE	1
	[15:8}	BYTE	2
	[23:16]	BYTE	3
	[31:24]	BYTE	4
	[39:32]	BYTE	5
	[47:40]	BYTE	6
MFR_VOUT_PEAK (PAGE 0)	[15:8]	LIN 16	7
	[7:0]	LIN 16	8
MFR_VOUT_PEAK (PAGE 1)	[15:8]	LIN 16	9
	[7:0]	LIN 16	10

LTC3880のデータシートには、フォルト・ログ・ヘッダに関するデータの詳しい説明が記載され(表1を参照)、データ・フォーマットも指定されています。例えば、MFR_VOUT_PEAKはLINEAR 16(L16)フォーマットです。データの値は、同じ名前のコマンドを読み出すことによって返されるのと同じ値、同じフォーマットになります。一番右の欄は、バイト・ブロック内のインデックスを示します。

LTC3880のPosition_Faultの値を表2に示します。.これは最初に発生するフォルトで、最も重要な情報です。これはほかの値の前にEEPROMに書き込まれます。

表2. LTC3880のPosition_Fault

Position_Faultの値の説明

POSITION_FAULTの値	フォルト・ログの発生要因
0xFF	MFR_FAULT_LOG_STORE
0x00	TON_MAX_FAULTチャネル0
0x01	VOUT_OV_FAULTチャネル0
0x02	VOUT_UV_FAULTチャネル0
0x03	IOUT_OC_FAULTチャネル0
0x05	OT_FAULTチャネル0
0x06	UT_FAULTチャネル0
0x07	VIN_OV_FAULTチャネル0
0x0A	MFR_OT_FAULTチャネル0
0x10	TON_MAX_FAULTチャネル1
0x11	VOUT_OV_FAULTチャネル1
0x12	VOUT_UV_FAULTチャネル1
0x13	IOUT_OC_FAULTチャネル1
0x15	OT_FAULTチャネル1
0x16	UT_FAULTチャネル1
0x17	VIN_OV_FAULTチャネル1
0x1A	MFR_OT_FAULTチャネル1

LTC297Xファミリは、フォルトの原因、ヘッダ、ループ・データで構成される、これとよく似たフォルト・ログ構造を持ちます。詳細については、LTC297Xのデータシートを参照してください。[1]

コード・ストラテジ

Linduinoコードは生データを読み出してデコードし、ネストされたC構造へ変換できます。ほとんどの場合は1対1のマッピングになりますが、場合によっては一部のデータはコピーする必要があります。データが構造化されると、Linduinoコードは、ログの要素を処理することや、人が判読可能なテキストを生成することができます。

一部のアプリケーションでは、BMCはデータを全く処理しません。この場合は、データのブロックをネットワーク経由で送信し、ほかのコンピュータ上で処理できます。そのコンピュータは同じストラテジでデータを処理できます。Linduinoコードは汎用コードなので、BMC上でも、サーバー、ノートPCなどでも実行できます。

スケッチを使用したログのダンプ

ライブラリ・コードを検討する前に、図2に示すように、DC1962のフォルト・ログを出力するLTSketchbookのスケッチを検討しましょう。DC1962は、LTC3880、LTC2974、およびLTC2977を搭載しています。

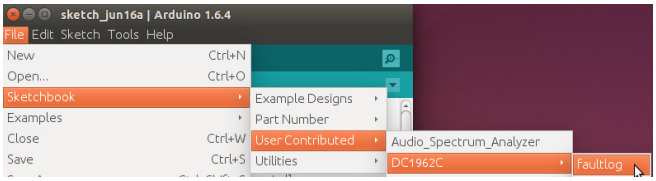

図2. フォルト・ログ・スケッチのロード

スケッチは、User ContributedセクションのDC1962Cの下にあります。スケッチをロードした後、コンパイル、アップロードを実行し、コンソール・ウィンドウを開きます(図3を参照)。

```
1-Dump Fault Logs
2-Clear Fault Logs
3-Clear Faults
4-Bus Probe
5-Reset
6-Store Fault Log

Enter a command:
```

図3. フォルト・ログ・スケッチのメニュー

このメニューには、3つの主なコマンド(Dump、Clear、Store)があります。Storeメニュー・アイテムを選択すると、フォルト・ログが生成されます。または、DC1962上でCREATE FAULTボタンを押して、フォルトを発生させることができます。

例えば、DC1962C上のCH0出力をGNDに短絡させて、メニューから[Dump Fault Log]を選択します。結果のLTC3880フォルト・ログを図4に示します。

データの一番上に注意してください。Fault Positionはチャネル0のIOUT_OC_FAULTです。低電圧コンパレータよりも前に、過電流コンパレータがフォルトになっています。デバイスの過電流および低電圧設定によっては、この条件で低電圧フォルトが先に発生する場合があります。次に時間が示されます。これはリセット以降にシステムがどれだけの時間動作していたかを示します。この時間は、ほかのデバイスのフォルト・ログと相関させるのに便利です。

次に、リセットまたはクリア・コマンド以降の全てのピーク値が示されます。このデータは、温度または出力電流が通常よりも高い場合の原因解明の手がかりとして役立ちます。

Note 1: Linduinoコードはフォルト・ログのデータを自動的に処理するため、データの構造に関する専門知識は必要ありません。

最後に、データはテレメトリのループを表示します。ループ0でChan0が0.0Vであることに注意してください。ループ0は直近のテレメトリで、そこでフォルトが発生しました。ループ1はフォルトの直前に測定されました。ループ1での電圧は0.849Vでしたが、ループ0では0.0Vです。テレメトリ・ループの所要時間は約100msで、フォルトが発生したとき、最後のループが完了します。したがって、フォルト発生前の出力電圧データを(ループ0内であっても)得ることが可能です。

注記:DC1962上の負荷が非常に小さいため、出力電流は小さくなっています。

```
LTC3880 Log Data
Fault Position IOUT_OC_FAULT Channel 0
Fault Time 0x00000002dcc5
187589 Ticks (200us each)

Header Information:
--------
VOUT Peak 0 0.852295
VOUT Peak 1 1.100830
IOUT Peak 0 0.002304
IOUT Peak 1 0.002975
VIN Peak 4.984375
Temp External Last Event Page 0
30.156250
Temp External Last Event Page 1
30.218750
Temp External Last Event 36.875000
Temp External Peak Page 0 30.468750
Temp External Peak Page 1 30.625000

Fault Log Loops Follow:
(most recent data first)
-------
Loop: 0
-------
Input: 4.929687 V, 0.099976 A
Chan0: 0.000000 V, -0.000168 A
  STATUS_VOUT:0x00
  STATUS_MFR_SPECIFIC:0x00
  STATUS_WORD:0x4851
Chan1: 0.094971 V, 0.000092 A
  STATUS_VOUT:0x00
  STATUS_MFR_SPECIFIC:0x01
  STATUS_WORD:0x1841
-------
Loop:1
-------
```

図4. LTC3880のフォルト・ログ

```
LTC2974 Log Data
Fault Time 0x00000002a218
172568 Ticks (200us each)

Peak Values and Fast Status
-----
Vout0: Min: 1.499512, Peak: 1.500732
Temp0: Min: 32.437500, Peak: 32.687500
Iout0: Min: 0.000122, Peak: 0.002777
Fast Status0
  STATUS_VOUT0: 0x00
  STATUS_IOUT0: 0x00
  STATUS_MFR0: 0x60

Vin:Min 4.937500, Peak: 4.992187

Vout1: Min: 1.799194, Peak: 1.802978
Temp1: Min: 32.312500, Peak: 32.437500
Iout1: Min: 0.000229, Peak: 0.003357
Fast Status1
  STATUS_VOUT1: 0x00
  STATUS_IOUT1: 0x00
  STATUS_MFR1: 0x60

Vout2: Min:1.999512, Peak: 2.000854
Temp2: Min:32.00000, Peak: 32.312500
Iout2: Min:0.000565, Peak: 0.004166
Fast Status2
  STATUS_VOUT2: 0x00
  STATUS_IOUT2: 0x00
  STATUS_MFR2: 0x68

Vout3: Min: 2.199219, Peak: 2.200928
Temp3: Min: 31.312500, Peak: 31.468750
Iout3: Min: 0.000076, Peak: 0.004303
Fast Status3
  STATUS_VOUT3: 0x00
  STATUS_IOUT3: 0x00
  STATUS_MFR3: 0x68

Fault Log Loops Follow:
(most recent data first)
-------
Loop:0
-------
CHAN3
  READ POUT: 0.009109 W
  READ IOUT: 0.004135 A
  STATUS IOUT: 0x00
  STATUS TEMP: 0x00
  READ TEMP: 31.312500 C
  STATUS_MFR: 0x18
  STATUS_VOUT: 0x00
  READ_VOUT: 2.199463 V
CHAN2:
```

図5. LTC2974のフォルト・ログ

LTC3880のフォルト・ログに続くのは、LTC2974のフォルト・データです(図5を参照)。短絡していないデバイスにフォルト・ログが存在するのはなぜでしょう。[2] Peak Values and Fast StatusのSTATUS_MFRnと、ループ0のSTATUS_MFRをご覧ください。STATUS_MFRnの値は0x60、0x60、0x68、0x68です。ループ0のSTATUS_MFRは0x18です。

LTC2974のSTATUS_MFR_SPECIFIC(表3を参照)は、STATUS_MFRnの値の意味を示します。Fast Statusから次のビットが“H”です。

- Status_mfr_fault1_in
- Status_mfr_fault0_in
- Status_mfr_dac_connected

DC1962上では、LTC3880のGPIOBラインがLTC2974のFAULTBラインに接続されています。LTC2974はこれらのラインがグランドに引き下げられたときにフォルトでオフになるように構成されています。これによってフォルト・ログが生成されます。全てのチャネルがfault0/1_statusイベントを報告し、これらのチャネルが共有フォルト・ピンに応答したことを示しています。チャネル0とチャネル1は、出力をオフにした後にDACから切断されたことも報告しています。その他のチャネルは、その後まもなくDACとの接続を失っています。

LTC2977のフォルト・ログは、LTC2974と同様のフォルトを示しています。

表3. LTC2974のSTATUS_MFR_SPECIFICのデータの内容

ビット	SYMBOL	動作	チャネル	フォルト
b[7]	Status_mfr_discharge	1 = オン・ステートに入ろうとしていたときにV_{OUT}放電フォルトが生じた。 0 = V_{OUT}放電フォルトは発生していない。	Current Page	Yes
b[6]	Status_mfr_fault1_in	FAULTB1ピンが“L”にアサートされているときにこのチャネルがオンになろうとした、または、最後のCONTROLピンのトグル動作、OPERATIONコマンドのオン、オフ・サイクル、またはCLEAR_FAULTSコマンドからFAULTB1ピンが“L”にアサートされることに応答して少なくとも1回このチャネルがシャットダウンされた。	Current Page	Yes
b[5]	Status_mfr_fault0_in	FAULTB0ピンが“L”にアサートされているときにこのチャネルがオンになろうとした、または、最後のCONTROLピンのトグル動作、OPERATIONコマンドのオン、オフ・サイクル、またはCLEAR_FAULTSコマンドからFAULTB0ピンが“L”にアサートされることに応答して少なくとも1回このチャネルがシャットダウンされた。	Current Page	Yes
b[4]	Status_mfr_servo_target_reached	サーボの目標値に達した。	Current Page	No
b[3]	Status_mfr_dac_connected	DACが接続され、V_{DAC}ピンをドライブしている。	Current Page	No
b[2]	Status_mfr_dac_saturated	最大または最小のDAC値で前のサーボ動作が終了している状態。	Current Page	Yes
b[1]	Status_mfr_auxfaultb_faulted_off	V_{OUT}またはI_{OUT}のフォルトにより、AUXFAULTBがデアサートされた。	All	No
b[0]	Status_mfr_watchdog_fault	1 = ウォッチドッグ・フォルトが生じた。 0 = ウォッチドッグ・フォルトは生じていない。	All	Yes

Note 2: 3つのデバイス全てに同時にフォルトが発生し、スケッチは3つのデバイス全てからフォルト・ログを読み出しました。

スケッチ・コード

スケッチ・コードは、ライブラリ・ファイルLT_PMBus/LT_FaultLog.hにあるAPIを使用します(図6を参照)。

```
bool hasFaultLog (uint8_t address);
void enableFaultLog(uint8_t address);
void disableFaultLog(uint8_t address);
void clearFaultLog(uint8_t address);
virtual void print(Print* printer = 0) = 0;
```

図6. フォルト・ログのAPI

このAPIの背後にあるコードは、デバイスのタイプを検出し、「適切な処置」を実行します。このAPIを使用すれば、図7のフォルト・ログ・スケッチ・コードに示すように、フォルト・ログを出力するコードを簡単に作成できます。FaultLogクラスは一般または基礎クラスであり、サブクラスは各デバイス・タイプ用に特化されます。

```
static LT_3880FaultLog *faultLog3880 =
  new LT_3880FaultLog(smbus);

if (
  faultLog3880->hasFaultLog(ltc3880_i2c_address))
{
  faultLog3880->read(ltc3880_i2c_address);
  faultLog3880->print(&Serial);
  faultLog3880->release();
}
else
  Serial.println(F("No LTC3880 Fault Log"));
```

図7. フォルト・ログ・スケッチ・コード

このコードは、デバイス固有のオブジェクトを使用するログがあるかどうかをチェックし、ログがあればログを読み出し、出力し、リリースする、という一連の作業のみを行います。「read」関数はログ・データ用の内部メモリを割り当て、「print」はデータを出力し、「release」はメモリを解放します。この構造に直接アクセスする必要がある場合は、各サブクラスに「get」関数を使用できます。この関数はそのデバイス・タイプの構造を返します。この構造を使用して、生データを直接処理するコードを作成できます。

コードを設計する際は、FaultLogクラスが共通のAPIを定義し、各サブクラスがデバイス固有の動作を実装すると考えてください。デバイスごとにPMBusコマンドは多少異なり、データ・フォーマットも異なるため、サブクラスは共通のAPIを使用して各デバイスに特化した動作を実装します。

移植

ライブラリ・コードは、print関数以外は汎用コードです。このコードを非Arduinoプラットフォームに移植するには、print関数を削除するか、置き換えます。また、構造がパックされていること、ポインタのキャストにメモリ・アライメントの問題がないことを確認します。

ユーザー独自のPMBusライブラリを使用する場合は、ライブラリに合わせてSMBus/PMBusコールを変更する必要があります。

まとめ

フォルト・ログはパワーシステム・マネージメント・デバイスの非常に強力な機能であり、フォルトに関する履歴を不揮発性メモリで提供します。LTC388XおよびLTC297Xファミリの生データの読み出しと解釈は、提供されるライブラリを使用して簡単に行えます。

Linduino PSMのフォルト・ログ・ライブラリとサンプル・スケッチは、共通のAPIを使用して、このジョブを簡単に処理できます。コードの移植は簡単です。Arduino固有のコードはプリント用コードだけで、置き換えが簡単です。デバイスのタイプごとにバイトをデコードしたりPMBusコマンドを学習したりする必要はありません。

Linduino PSMのフォルト・ログ・ライブラリまたはLinduino PSMについてご不明な点などありましたら、弊社または弊社代理店までお問い合わせください(www.linear-tech.co.jp/contact)。

リニアテクノロジー株式会社
〒102-0094 東京都千代田区紀尾井町3-6紀尾井町パークビル8F
TEL 03-5226-7291 • FAX 03-5226-0268 • www.linear-tech.co.jp

LT 0117 • PRINTED IN JAPAN