ANALOG DEVICES
想像を超える可能性を
AHEAD OF WHAT'S POSSIBLE™

バッテリの急速充電に関する設計ガイド【Part 2】

著者：Franco Contadini、フィールド・アプリケーション担当スタッフ・エンジニア
Alessandro Leonardi、フィールド・セールス担当アカウント・マネージャ

はじめに

前回の記事「バッテリの急速充電に関する設計ガイド【Part 1】」では、急速充電が可能なバッテリ・システムを設計する際に直面するいくつかの課題について解説しました。それを通して、バッテリ・パックに残量ゲージの機能を実装すれば、よりスマートな急速チャージャを実現できるということを明らかにしました。そのようなチャージャであれば、システムの高い柔軟性、消費電力の削減、安全な充放電を実現しつつ、ユーザ・エクスペリエンスを高めることができます。今回（Part 2）は、各種ICの評価用キットを利用して並列バッテリ向けの急速充電システムを構成し、その特性を評価する方法を詳しく解説します。システムの構成要素としては、チャージャ／残量ゲージIC「MAX17330」の評価用キット、降圧コンバータIC「MAX20743」の評価用キット、シングルボード・コンピュータ「Raspberry Pi」を使用することにします。

急速充電システムの構成方法

一般に、シンプルな充電システムの機能のテストや性能の評価は、評価用キットを使うことで容易に実施できます。その種のキットには、充電システムの構成（コンフィギュレーション）に必要なあらゆるハードウェア／ソフトウェアや、GUI（Graphical User Interface）ベースのツール、API（Application Programming Interface）などが含まれています。

しかし、複数のセルを使用する複雑なシステムの場合、その複雑さに比例して評価の難易度も高くなります。例えば、複雑なシステムには、特性評価が必要な複数のデバイスが含まれていることが多いでしょう。通常、システムを構成する様々なコンポーネントが生成する信号を取得／解析し、何らかの対応を図るには、コーディングの作業を伴うソフトウェアの開発が必要になるはずです。

本稿では、並列バッテリ向けの急速充電システムを例にとります。そのシステムは、バッテリ管理（バッテリ・マネージメント）用のICであるMAX17330を使用して構成することにしました。同ICを使用すれば、1個のリチウム・イオン・バッテリ・セルを管理することができます。データシートにも記載されていますが、同ICを2個使用すれば、2個のリチウム・イオン・バッテリ・セルを同時に充電／制御することが可能です。そこで、本稿で例にとる急速充電システムでは、それぞれ1個のLi+セルを管理する2個のMAX17330を使用することにします。また、出力電圧をオンザフライで変更する機能を備える降圧コンバータとしてMAX20743を使用することにします。

バッテリの充電に向けた構成／管理や、2つのICの間の通信を処理するためにはマイクロコントローラが必要です。これについては、Raspberry Piを使用することにしました。Raspberry Piは、システムのテストに適したプラットフォームとして一般的に使用されています。また、プログラミング言語としてはPythonを使用することにしました。

Raspberry Piを使用すれば、I^2Cを介した通信を管理することができます。また、評価とデバッグを行う際に役立つ重要なシステム・パラメータの値を記録することも可能になります。主要なパラメータとしては、充電電流、バッテリ電圧、バッテリの充電状態（SOC：State of Charge）などが挙げられます。これらの値は、オフラインで解析を行えるようにするためにExcelファイルに保存することにします。

1S2P構成のシステムの評価

本稿では、MAX17330が提供するチャージャ機能と残量ゲージ機能の評価方法を示します。また、並列充電によって期待できる実際の性能を明らかにします。最大限の柔軟性と制御性を得るために、デバイスのプログラミングは、I^2Cを介し、マイクロコントローラによって実施することにします。

図1は、1S2Pのシステムの構成例です。ここで、1S2Pとは、セルの直列接続の数が1、並列接続の数が2であることを表しています。図1には、2個のセルの並列充電について評価を行うための接続方法も示してあります。

Raspberry Piは、計3つの評価用キット（EVKIT）を制御します。1つは、降圧コンバータICであるMAX20743の評価キットです。残る2つはMAX17330の評価用キットです。図1に示したように、MAX17330の評価用キットは2つ使用します。また、先述したとおり、取得したデータはExcelファイルに記録することにします。

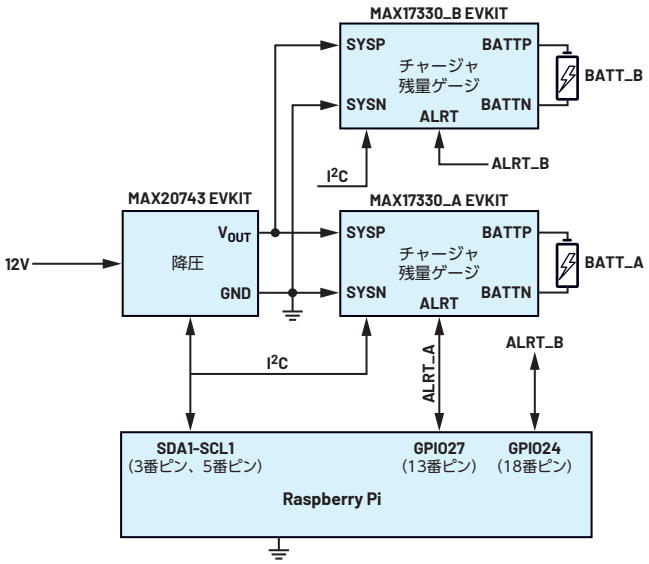

図1. 1S2Pの急速充電システム。Raspberry Piを利用することで、評価を実施できるようにしています。

MAX17330の評価用ソフトウェア（GUIベース）をダウンロードするには、同ICの製品ページにアクセスしてください。「ツールおよびシミュレーション」タブを選択すると表示されるリンクをクリックすることでダウンロードすることができます。このソフトウェアの「Configuration Wizard」（「Device」タブから選択）を使用することで、MAX17330の初期化用ファイル（拡張子は.INI）を生成することができます。このファイルには、同ICのレジスタを初期化するための情報が含まれています。その情報は「レジスタのアドレス/レジスタの値」の形式で記載されています。マイクロコントローラは、このファイルを使用してMAX17330の個々のレジスタの構成を行います。なお、「MAX17330EVKIT」のデータシートを見れば、初期化用ファイルの生成に必要なステップについて詳細を確認することができます。

本稿の例では、図2に示した構成を使用して並列充電を開始します。続いてステップ充電を有効にします（図3）。図4に示したのは、図3の構成によるステップ充電のプロファイルです。

図2. 並列充電用の構成

図3. ステップ充電を有効にするための構成

降圧コンバータであるMAX20743は、必要に応じて2つのMAX17330EVKITに印加する電圧を変更するために使用します。MAX20743の出力電圧は、同ICが内蔵するレジスタのアドレス0x21の値に応じて変更されます。この降圧コンバータは、I²Cを介して制御することが可能です。実際の制御は、Pythonで記述したプログラムによって行います。

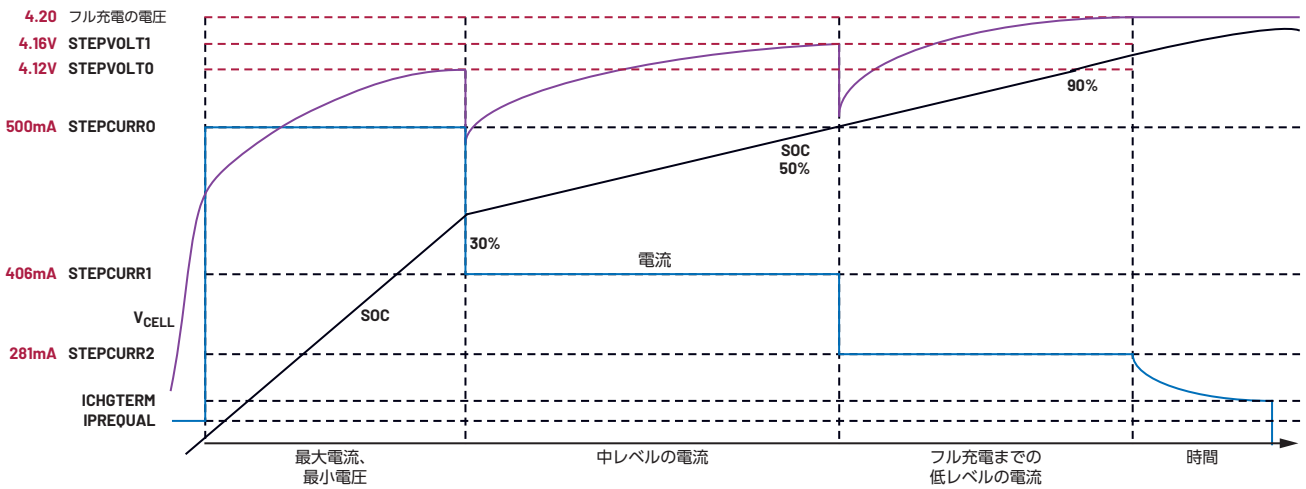

図4. 図3の構成に対応するステップ充電のプロファイル

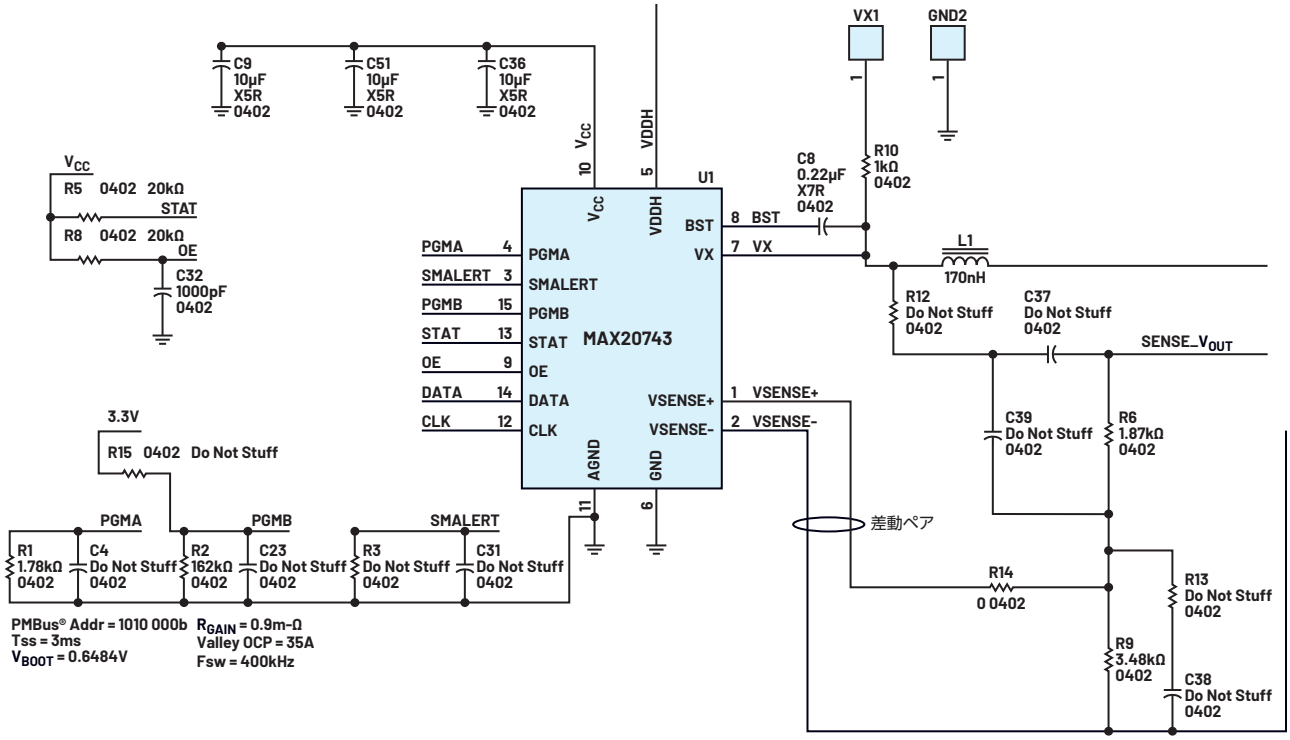

図5. 出力範囲が3V～4.6Vになるように変更した出力分圧器（R6 = 4K7、R9 = 1K3）

図5に示すように、MAX20743EVKITの出力分圧器は、出力範囲が3V～4.6Vになるように変更してあります（R6 = 4K7、R9 = 1K3の値を使用します）。

表1. MAX20743の0x21レジスタの値と出力電圧の関係

0x21レジスタの値	電圧
0x014E	3V
0x0150	3.05V
0x0158	3.1V
0x015C	3.15V
0x0162	3.2V
0x0166	3.25V
0x016E	3.3V
0x0172	3.35V
0x0178	3.4V
0x017C	3.45V
0x0182	3.5V
0x0188	3.55V
0x018E	3.6V
0x0192	3.65V
0x019E	3.7V
0x01A4	3.75V
0x01A9	3.8V
0x01AE	3.85V

表1.（続き）

0x21レジスタの値	電圧
0x01B4	3.9V
0x01BA	3.95V
0x01BF	4V
0x01C4	4.05V
0x01CB	4.1V
0x01D1	4.15V
0x01D6	4.2V
0x01DC	4.25V
0x01E2	4.3V
0x01E8	4.35V
0x01ED	4.4V
0x01F3	4.45V
0x01F8	4.5V
0x01FE	4.55V
0x0204	4.6V

表1は、レジスタの値とそれに対応する電圧の値についてまとめたものです。両者の関係は以下の式で表されます。

$$[\text{レジスタの値}] = 0 \times 014e + \left(\frac{x-3}{0.1 \times 11}\right)$$

ここで、xはMAX20743によって出力したい電圧の値です。少し誤差は生じますが、この式を使用すれば、出力したい電圧の値を基に、必要なレジスタの値を簡単に推定することができます。

表2. MAX17330のレジスタ

レジスタのページ	ロック	説明	2線式のノード・アドレス	2線式のプロトコル	2線式の外部アドレス範囲
00 h		ModelGauge m5 EZ用のデータ・ブロック	6チャンネル	I²C	00 h – 4 Fh
01 h – 04 h	Lock 2				
05 h – 0Ah		Reserved			
0 Bh	Lock 2	ModelGauge m5 EZ用のデータ・ブロック（続き）	6チャンネル	I²C	B0 h – BFh
0 Ch	SHA	SHA用のメモリ	6チャンネル	I²C	C0h – CFh
0 Dh	Lock 2	ModelGauge m5 EZ用のデータ・ブロック（続き）	6チャンネル	I²C	D0h – DFh
0 Eh – 0 Fh		Reserved			
10 h – 17 h		SBS用のデータ・ブロック	16チャンネル	SBS	00 h – 7 Fh
18 h – 19 h	Lock 3	ModelGauge m5 EZ用の不揮発性メモリ・ブロック	16チャンネル	I²C	80 h – EFh
1 Ah – 1 Bh	Lock 1	状態記録／構成用の不揮発性メモリ・ブロック			
1 Ch	Lock 4	構成用の不揮発性メモリ・ブロック			

表3. nPackCfg（1B5h）レジスタのフォーマット

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	S_Hib	THCfg		THType	000			0	ParEn	I²CSid		0001			

表4. I2CCmd（12Bh）レジスタのフォーマット

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0										GoToSID		0			IncSID

起動と初期化

MAX17330は、最初にバッテリに接続する際、デフォルトのレジスタの設定によって強制的にシャットダウンの状態になります。同ICを起動するには、PKWKボタンを押します。それにより一時保護用のMOSFETが導通（ショート）して、2つのMAX17330EVKITが起動します。

次に、I²Cを介してRaspberry Piと3つのICの間の通信を実現する必要があります。I²Cに対応するハードウェアは、ICのアドレスの競合が生じないようにするために慎重に初期化しなければなりません。デフォルトでは、2つのMAX17330EVKITはI²Cの同じアドレスを使用するようになっています。そのため、最初のステップとして、2つのうち一方のアドレスを変更しなければなりません。

MAX17330は、揮発性と不揮発性のレジスタを備えています。不揮発性のレジスタには「n」の接頭辞が付加されています。ノードのアドレスは6Ch（揮発性レジスタ）と16h（不揮発性レジスタ）のペアになっています。

MAX17330のデバイス・ノードのアドレスは、以下の2つの方法で変更できます。

- I²CSid フィールドを使用して不揮発性レジスタ nPackCfg の設定を行います。この作業は、Configuration Wizard を使用することで容易に実施できます（表 3）。
- I2CCmd レジスタによって、I²C のバスを動的に変更します (表 4)。

ここでは、使いやすさを優先して2つ目の方法でアドレスを変更することにします。そうすれば、同じ初期化用ファイルを使って両方のICを初期化することができます。2つのICで共用できる事柄を増やせば、設定作業を簡素化することが可能です。このことは、アドレスを入力する際に人的ミスが生じる可能性を低減することにつながります。

2つのMAX17330は同じI²Cバスを共有します。そのため、上記の処理を行うには、一方のデバイスのALRT信号がハイになっている間、もう一方のALRT信号をローにする必要があります。

表5. I²CによるALRTの設定

GoToSID	Alertがハイ	Alertがロー
	プライマリ／セカンダリ・アドレス	プライマリ／セカンダリ・アドレス
0b00	ECh/96h	6Ch/16h
0b01	64h/1Eh	ECh/96h
0b10	E4h/9Eh	64h/1Eh
0b11	6Ch/16h	E4h/9Eh

表4は、MAX17330のデータシートから引用したものです。この表から、I2CCmdレジスタにより、ALERT GPIOピンの値に基づいて同ICのアドレスを変更できることがわかります。ここでは、GoToSIDフィールドとIncSIDフィールドを使用して、I²Cのアドレスを以下のようにして変更します。

表6. nProtCfg（1D7h）レジスタのフォーマット

D15	D14	D13	D12	D11	D10	D9	D8
ChgWDTEn	$\overline{\text{nChgAutoCtrl}}$	FullEn	SCTest		CmOvrdEn	ChgTestEn	PrequalEn
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Reserved	PFEn	DeepShpEn	OvrdEn	$\overline{\text{UVRdy}}$	FetPFEn	BlockDisCEn	DeepShp2En

表7. Config2（OABh）レジスタのフォーマット

D15	D14	D13	D12	D11	D10	D9	D8
POR_CMD	0	AtRtEn	0	0	0	0	0
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
dSOCen	TAlrtEn	0	1	DRCfg		CPMode	BlockDis

表8. Config（O0Bh）レジスタのフォーマット

D15	D14	D13	D12	D11	D10	D9	D8
0	SS	TS	VS	0	PBen	DisBlockRead	$\overline{\text{ChgAutoCtrl}}$
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
SHIP	COMMSH	FastADCen	ETHRM	FTHRM	Aen	CAen	PAen

- Set ALRT_A logic low
- Set ALRT_B logic high
- Write I2CCmd = 0 × 0001
  → MAX17330_A address remains at 6Ch/16h
  → MAX17330_B address set to ECh/96h

これで各MAX17330にそれぞれ固有のアドレスが割り当てられることになります。その結果、システム全体を単一のコントローラによって制御することが可能になります。

マイクロコントローラによってI^2Cの構成を完了させるためのスクリプトを以下に示します。これは、システムの初期化作業の一部に相当します。

- Load .INI file
- Assert ALRT_A and ALRT_B to keep the path between SYSP and BATTP open
- Read V_{BATT}_A and V_{BATT}_B
- V_{MAX} = max (V_{BATT}_A, V_{BATT}_B)
- Set V_{OUT} = V_{MAX} + 50 mV
- Release ALRT_A and ALRT_B
- Set nProtCfg.OvrdEn = 0 to use ALRT as Output

上記のスクリプトについては、表6を参考にしてください。

不揮発性レジスタの中には、ファームウェアを再起動しなければ変更内容が有効にならないものがあります。そのため、以下のステップが必要になります。

- Assert Config2.POR_CMD to restart firmware

これについては表7を参照してください。

次に、チャージャからの割り込みを有効にする必要があります（以下参照）。

- Set (Config.Aen and Config.Caen) = 1

これについては表8をご覧ください。以上でデバイスの初期化が完了したことになります。

データのロギングと割り込み

データを記録し、ALERT GPIOライン上で割り込み信号が生成されているかどうかを確認するためには、レジスタの値を読み出せるようにする必要があります。そのためのスクリプトを以下に示します。

- Set 500 ms Timer
- V_{MIN} = min (V_{BATT}A, V_{BATT}_B)
- Vsys_min = nVEmpty[15:7]
- CrossCharge = False
- If (V_{MIN} < Vsys_min) → CrossCharge = True

  最小バッテリ電圧がシステムの最小動作電圧を上回っているかどうかを評価する

- If FProtStat.IsDis = 0

  充電信号が検出された場合

- Clear Status.AllowChgB

  チャージャの存在を全バッテリに通知する

- If (V_{BATT} > V_{MIN} + 400 mV and !Cross Charge)

  クロス充電を防ぐために、どのバッテリをブロックするのかを判定する

表9. FProtStat（0Dah）レジスタのフォーマット

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X										IsDis	X		Hot	Cold	Warm

表10. Status（000h）レジスタのフォーマット

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
PA	Smx	Tmx	Vmx	CA	Smn	Tmn	Vmn	dSOCi	Imx	AllowChgB	X	Bst	Imn	POR	X

表11. Config2（0ABh）レジスタのフォーマット

D15	D14	D13	D12	D11	D10	D9	D8
POR_CMD	0	AtRtEn	0	0	0	0	0

D7	D6	D5	D4	D3	D2	D1	D0
dSOCen	TAlrtEn	0	1	DRCfg		CPMode	BlockDis

表12. Status（000h）レジスタのフォーマット

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
PA	Smx	Tmx	Vmx	CA	Smn	Tmn	Vmn	dSOCi	Imx	AllowChgB	X	Bst	Imn	POR	X

表13. ChgStat（0A3h）レジスタのフォーマット

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Dropout	X	X	X	X	X	X	X	X		X	X	CP	CT	CC	CV

Config2.BlockDis = 1

else

Config2.BlockDis = 0

低残量のバッテリの残量が高残量のバッテリの残量よりもかなり少ない場合、放電を許可する

上記のスクリプトについては表9～11を参照してください。

MAX17330からALRTがアサートされた場合、ホストは以下の処理を実行します。

Read Status register data

If Status.CA is set

Read ChgStat register

If ChgStat.Dropout = 1 → increase V_{OUT}

If (ChgStat.CP or ChgStat.CT) = 1 → decrease V_{OUT}

Clear Status.CA

上記のスクリプトについては表12と表13をご覧ください。

図6に示したのは、並列充電システムの評価結果です。これらのプロットは、ログ・データ（Excelファイル）を基にして作成しました。ステップ充電のプロファイルに即した結果が得られている点に注目してください。

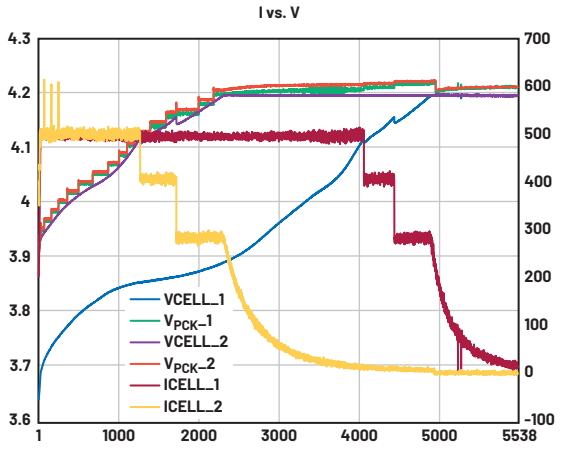

図6. 並列充電の評価結果

オプションを1つ紹介しておきましょう。MAX17330が定電流（CC：Constant Current）のフェーズから定電圧（CV：Constant Voltage）のフェーズに移行した後に、次のようにすれば降圧コンバータの出力電圧を低下させることができます。

▶ If V_{BATT} = ChargingVoltage

Read ChgStat Register

If ChgStat.CV = 1 → decrease V_{OUT} until V_{PCK} = ChargingVoltage + 25 mV

以上が、1S2P構成の充電システムを管理するために必要な全ステップです。MAX17330-usercode.zipというファイルには、MAX17330（チャージャ／残量ゲージ）とMAX20743（降圧コンバータ）を構成するためのPythonのコードが含まれています。また、重要なパラメータの値を記録してステップ充電のプロファイルを評価するためのデータ・ログ（Excelファイル）も含まれています。MAX17330によって生成されるアラート信号を管理することで、マイクロコントローラは同ICのリニア・チャージャをドロップアウト電圧の近くで動作させ、消費電力を最小限に抑えつつ高い充電電流を供給することができます。MAX17330を採用したバッテリ・パックには、効率的な急速充電を実現するためにマイクロコントローラが必要とするバッテリのパラメータの値が保存されます。それにより、性能や信頼性を損なうことなく、標準的なチャージャICを、よりシンプルで安価な降圧コンバータに置き換えることが可能になります。

まとめ

機器の充電時間は、優れたユーザ・エクスペリエンスを提供する上で非常に重要な要素になります。MAX17330のような製品を採用することで、実装面積を抑えつつ、非常に多くの電流を効率的に管理して充電時間を短縮することができます。また、MAX17330を2個使用すれば、非常に多くの電流によって並列充電を行うことが可能になります。つまり、充電時間を最小限に抑えつつ、複数のバッテリを安全かつ確実に充電するための手段が得られるということです。

著者について

Franco Contadiniは、アナログ・デバイセズのフィールド・アプリケーション・エンジニアです。35年以上にわたってエレクトロニクス業界に従事。基板の設計とASICの設計を10年間経験した後、フィールド・アプリケーション・エンジニアとなりました。産業、通信、医療の各分野のお客様を対象とし、パワー・マネージメントやバッテリ管理（バッテリ・マネージメント）、シグナル・チェーン、暗号化システム、マイクロコントローラなどのサポートを行っています。シグナル・チェーンや電源に関する複数のアプリケーション・ノートや記事の執筆を担当。イタリアジェノバのITISでエレクトロニクスについて学びました。

Alessandro Leonardiは、アナログ・デバイセズ（ミラノ）でフィールド・セールスを担当するアカウント・マネージャです。ミラノ工科大学で電子工学の学士号と修士号を取得。その後、アナログ・デバイセズのフィールド・アプリケーション・トレーニー・プログラムに参加しました。

*英語版技術記事はこちらよりご覧いただけます。

TA24441-5/23

※初出典　2023年 TECH+（マイナビニュース）