
PIC®とAVR®デバイス用JSON復号部

序説

著者: Alexandru Niculae, Microchip Technology Inc.

この資料はPIC®とAVR®マイクロコントローラのような組み込みデバイスを狙いとしたJSON(JavaScript Object Notation)制限版復号部の使い方と実装を記述します。

復号部はJSONオブジェクトの文字列形式をCデータ構造体表現に変換します。このようにしてプログラム作成者は符牒(Key)と値(Value)の対にアクセスすることができます。JSONオブジェクトの使用は応用の相互連絡を容易にします。

JSON復号部コードはGitHubで入手可能です。



GitHubでコード例を見てください。
貯蔵庫を閲覧するにはクリックしてください。

本書は一般の方々の便宜のため有志により作成されたもので、Microchip社とは無関係であることを御承知ください。しおりの[はじめに]での内容にご注意ください。

目次

序説	1
1. 概要	3
2. API	3
3. 実装	5
3.1. 入力緩衝部	5
3.2. 復号	5
3.3. 内部表現	5
3.4. 符号化	6
4. 実演 : PCからAVR-IoT上の4つのLEDを制御	6
4.1. 手順	6
5. 実演 : IoT節点との汎用クラウド通信	10
6. 追補	10
Microchipウェブ サイト	12
製品変更通知サービス	12
お客様支援	12
Microchipデバイスコード保護機能	12
法的通知	12
商標	13
品質管理システム	13
世界的な販売とサービス	14

1. 概要

JSONはウェブと机上応用で最も使われるデータ直列化形式です。これは事実上全ての最新のプログラミング言語で実装されるため、応用間のデータ交換に最適な道具です。これは人にとって読み易く、機械にとって解析と生成が容易な文字形式を持ちます。

JSONオブジェクトは符牒(Key)と値(Value)の対の組です。符牒は文字列にだけでき、値は色々にできます。この復号部実装は文字列、数値、オブジェクト値を支援します。

注: このライブラリを少メモリデバイスに適合させておくため、実装に対して機能の一部分だけが選ばれ、従って、配列、論理型(ブーリアン)、ヌル(null)値は支援されません。

以下のコードの塊はこの復号部の使い方の例を示します。2. API章はAPI関数を詳細に記述します。

```
#define TEST_JSON "{¥"main¥":{¥"key¥" : 10,¥"foo¥":¥"bar¥"}, ¥"alt¥":2}"

~

jsonNode_t *root, *objmain;
char str[64], foo[10];
int alt;

memcpy(str, TEST_JSON, sizeof(TEST_JSON));

JSON_DECODER_fromString(str)

JSON_DECODER_getRoot(&root);
JSON_DECODER_getObject(root, "main", &objmain);

JSON_DECODER_getNumber(root, "alt", &alt)
JSON_DECODER_getString(objmain, "foo", sizeof(foo), foo)
```

注: JSON内の最も外側のオブジェクトがルート(root)と呼ばれます。

2. API

JSON復号部の公開関数の詳細な説明についてはこの章を参照してください。

```
jsonDecoderStatus_t JSON_DECODER_fromString(char *str)
```

説明

JSON文字列をC表現に解析。文字列、数値、オブジェクト値を支援します。

パラメータ

名前	説明
str	JSONオブジェクトの既存文字列表現への位置指示子。解析後、文字列は変更されます。

戻り値

名前	説明
JSON_DECODER_OK	復号は成功しました。
JSON_DECODER_BAD_FORMAT	入力が無効です。

```
JSON_DECODER_getRoot(jsonNode_t **pNode)
```

説明

ルートと呼ばれるJSONの最も外側のオブジェクトを探します。この関数はJSON_DECODER_fromString前に呼ばれてはなりません。

パラメータ

名前	説明
pNode	jsonNode_tへの位置指示子。

戻り値

名前	説明
JSON_DECODER_OK	pNodeは今やルートを指示します。

```
jsonDecoderStatus_t JSON_DECODER_getObject(jsonNode_t *current, char *key, jsonNode_t **pNode)
```

説明

別のJSONオブジェクトで符牒によってJSONオブジェクトを探します。この関数はJSON_DECODER_fromString前に呼ばれてはなりません。

パラメータ

名前	説明
current	jsonNode_tへの位置指示子。
key	見つけるオブジェクトの符牒。
pNode	見つかった場合、そのオブジェクトへの位置指示子。

戻り値

名前	説明
JSON_DECODER_OK	オブジェクトが見つかり、今やpNodeはそれを指示します。
JSON_DECODER_KEY_NOT_FOUND	指定した符牒が存在しません。

```
jsonDecoderStatus_t JSON_DECODER_getString(jsonNode_t *current, char *key, uint8_t size, char *pVal)
```

説明

JSONオブジェクトで符牒によって文字列を探します。この関数はJSON_DECODER_fromString前に呼ばれてはなりません。

パラメータ

名前	説明
current	検索するJSONオブジェクト。
key	探す文字列の符牒。
size	文字列の最大の大きさ。pVal緩衝部の長さより大きくてはなりません。
pVal	見つかった場合、その文字列への位置指示子。

戻り値

名前	説明
JSON_DECODER_OK	文字列が見つかり、今やpNodeはそれを指示します。
JSON_DECODER_KEY_NOT_FOUND	指定した符牒が存在しません。

```
jsonDecoderStatus_t JSON_DECODER_getNumber(jsonNode_t *current, char *key, int *pVal)
```

説明

JSONオブジェクトで符牒によって数値を探します。この関数はJSON_DECODER_fromString前に呼ばれてはなりません。

パラメータ

名前	説明
current	検索するJSONオブジェクト。
key	探す数値の符牒。
pVal	見つかった場合、その数値への位置指示子。

戻り値

名前	説明
JSON_DECODER_OK	数値が見つかり、今やpNodeはそれを指示します。
JSON_DECODER_KEY_NOT_FOUND	指定した符牒が存在しません。

3. 実装

本章はコードのより深い理解を促進する目的で実装の詳細を記述します。更に、メモリ使用と実装安全性を最適化させるいくつかの判断を記述します。例えば、再帰の使用が推奨されないため、算法の一部が内部整理されました。

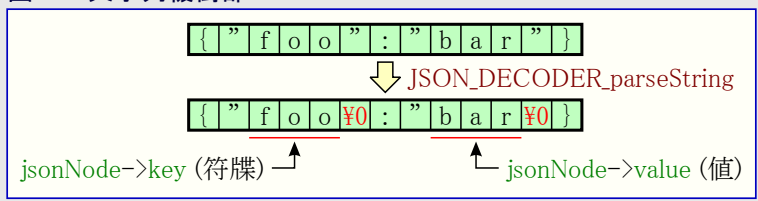
3.1. 入力緩衝部

入力緩衝部はMQTTドライバ、UARTドライバ、または他の通信方法によってJSON文字列を受け取る緩衝部です。この緩衝部はJSON_DECODER_parseString関数を使ってJSON復号部に渡されます。この関数が戻る時に緩衝部の内容が変更されています。

可能な限りメモリを少なく使うため、内部JSONオブジェクト表現はどの文字列も保持しませんが、元々入力緩衝部に置かれた文字列への位置指示子(ポインタ)を保持します。従って、文字列を分離するためその緩衝部内にヌル終了子が挿入されます。全ての符牒と文字列が入力緩衝部内です。

重大な影響はJSONデータが使われる前に緩衝部を書き換えないように注意が払われなければならないことです。緩衝部がJSON_DECODE_parseStringとJSON構造体アクセス間書き換えられる場合、これはデータを不正にします。

図3-1. 文字列緩衝部



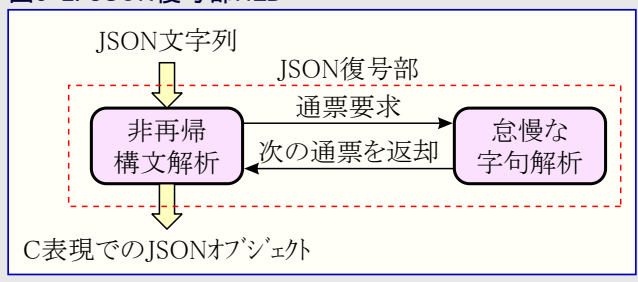
3.2. 復号

復号はJSONオブジェクトの文字列形式からCデータ構造体表現への変換を意味します。これは文法解析問題で、通常、字句解析と構文解析の2段階処理で解決されます。

字句解析は空白のような余分な文字を取り除くだけでなく、入力文を通票(トークン)に分離します。通票は単一文字(中括弧、コンマ、コロン)と複数文字(文字列、数値)の両方です。この実装は全ての通票の一覧を作成する代わりに意味する「怠慢な字句解析」を使い、これらは構文解析によって必要とされる時に1つつ取得されます。

構文解析は通票の流れがJSON文法の脈絡で意味を成すことを確認します。内部表現をCデータ構造体に構築します。構文解析は入れ子にされたJSONオブジェクトを処理するため、本来、本質的に再帰的です。再帰(とスタック溢れ異常の可能性)を避けるため、算法は実際のスタックの代わりにヒープスタックとWhile繰り返しを使うように内部整理されました。

図3-2. JSON復号部HLD

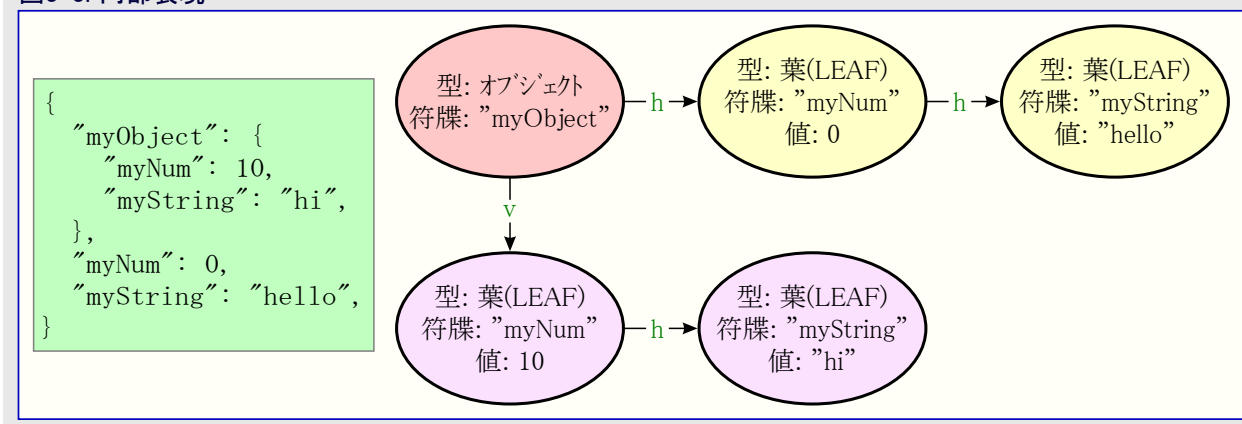


3.3. 内部表現

文字列形式構文解析後、JSONオブジェクトは樹状様の構造体としてメモリに格納されます。API関数使用で、この木構造は符牒名によって値を取得するために走査されます。各節点は符牒に対応し、(文字列または数値を持つ)葉(LEAF)節点またはオブジェクト節点のどちらかであり得ます。

図3-3はJSONオブジェクトとjsonNode_t構造体の木構造としてのその表現を示します。節点構造体は型領域、符牒領域、値領域と隣接節点への2つの位置指示子領域を持ちます。現在の節点が型オブジェクトの場合、h位置指示子は同じJSONオブジェクト内の次の符牒を指示し、v位置指示子は自身の最初の符牒を指示します。

図3-3. 内部表現



注: JSONオブジェクトが復号されるために持ち得る符牒の最大数は静的に定義されたjsonNode_t配列の長さによって制限されます。組み込み系では動的メモリ配置が避けられるべきです。

注: 入れ子にされたオブジェクトの最大数は構文解析部によって使われるヒープの大きさによって決められます。

3.4. 符号化

例えこの実装が対応する符号化を含んでいなくても、これはprintf/sprintf関数を使い、単にこれらの関数の形式(フォーマット)パラメータを使うことによって達成することが容易です。以下のコード行は文字列形式でJSONオブジェクトを作成します。これは測定したデータを報告するのにMQTT/UART等を渡って送ることができます。

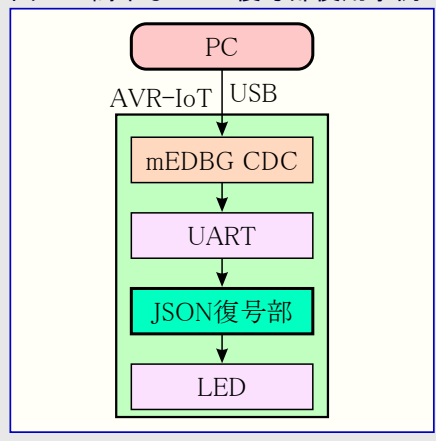
```
char json[30];
sprintf(json, "{\"Light\":%d,\"Temp\":%d}\", 10, 25);
```

4. 実演 : PCからAVR-IoT上の4つのLEDを制御

本章は簡単なMCCプロジェクトにJSON復号部を追加する方法とTera Term端末模倣器から送るJSON形式化された命令を送ることによってAVR-IoT-WG基板上的の4つのLEDを制御する方法を説明します。

この作業に使うJSONオブジェクトは同時に1、2、3、または4つのLEDを制御することができる明確な形式と処理論理を持つ柔軟な単一命令をもたらします。

図4-1. 簡単なJSON復号部使用事例



4.1. 手順

AVR-IoT基板、MPLAB® X 5.25、AVR® MCU周辺機能ライブラリ2.0.2(より古い版も動くかもしれませんが)を持つMCC 3.85.1、それとGitはこれらの項での手順に従うことが必要とされます。

4.1.1. MCCでのプロジェクト自動生成

1. ATmega4808デバイス用の新しいMPLAB Xプロジェクトを作成してください。
2. プロジェクトの構成設定を開始するためにMCCアイコンをクリックしてください。
3. Pin Manager: Grid View(ピン管理部: 格子表示)ウィンドウでPD0～PD3のピンを出力として記してください。

図4-2. ピン管理部: 格子表示部(Pin Manager: Grid View)

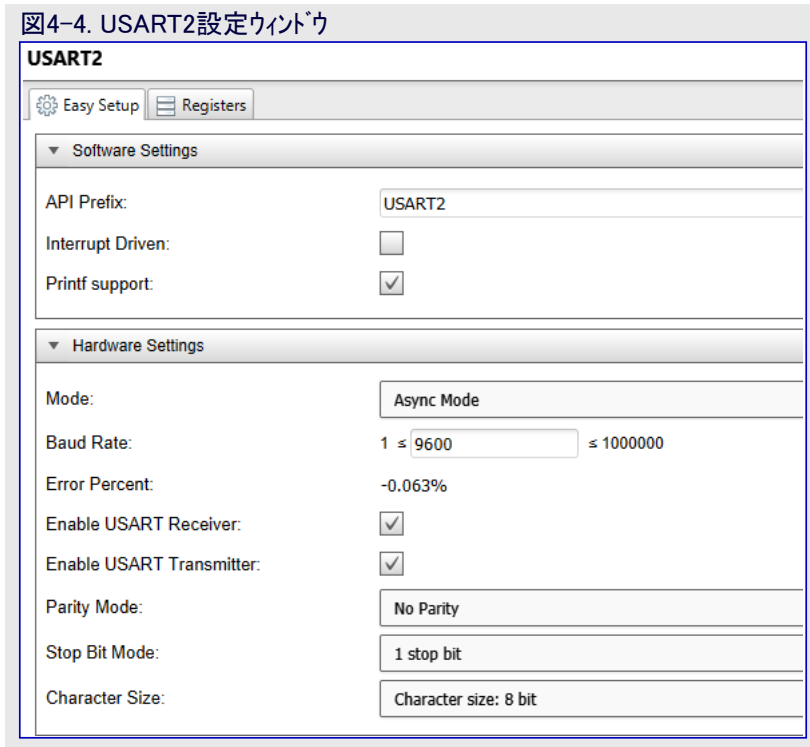
Output	Search Results	Notifications [MCC]	Pin Manager: Grid View x																			
Package:	QFP32	Pin No:	30	31	32	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
			Port A ▼				Port C ▼				Port D ▼											
Module	Function	Direction	0	1	2	3	4	5	6	7	0	1	2	3	0	1	2	3	4	5	6	
CLKCTRL ▼	CLKI	input	<input checked="" type="checkbox"/>																			
	CLKO	output								<input checked="" type="checkbox"/>												
	TOSC1	input																				
	TOSC2	input																				
Pin Module ▼	GPIO	input	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	GPIO	output	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
RSTCTRL	RESET	input																				

4. Pin Module(ピン単位部)ウィンドウでピンをLED_RED、LED_YELLOW、LED_GREEN、LED_BLUEに改名し、その後にそれら全てに対してINVEN(反転I/O許可)をチェックしてください。

図4-3. ピン単位部(Pin Module)ウィンドウ

Pin Name	Module	Function	Custom Name	OUTPUT	START HIGH	INVEN
PD0	Pin Module	GPIO	LED_RED	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
PD1	Pin Module	GPIO	LED_YELLOW	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
PD2	Pin Module	GPIO	LED_GREEN	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
PD3	Pin Module	GPIO	LED_BLUE	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

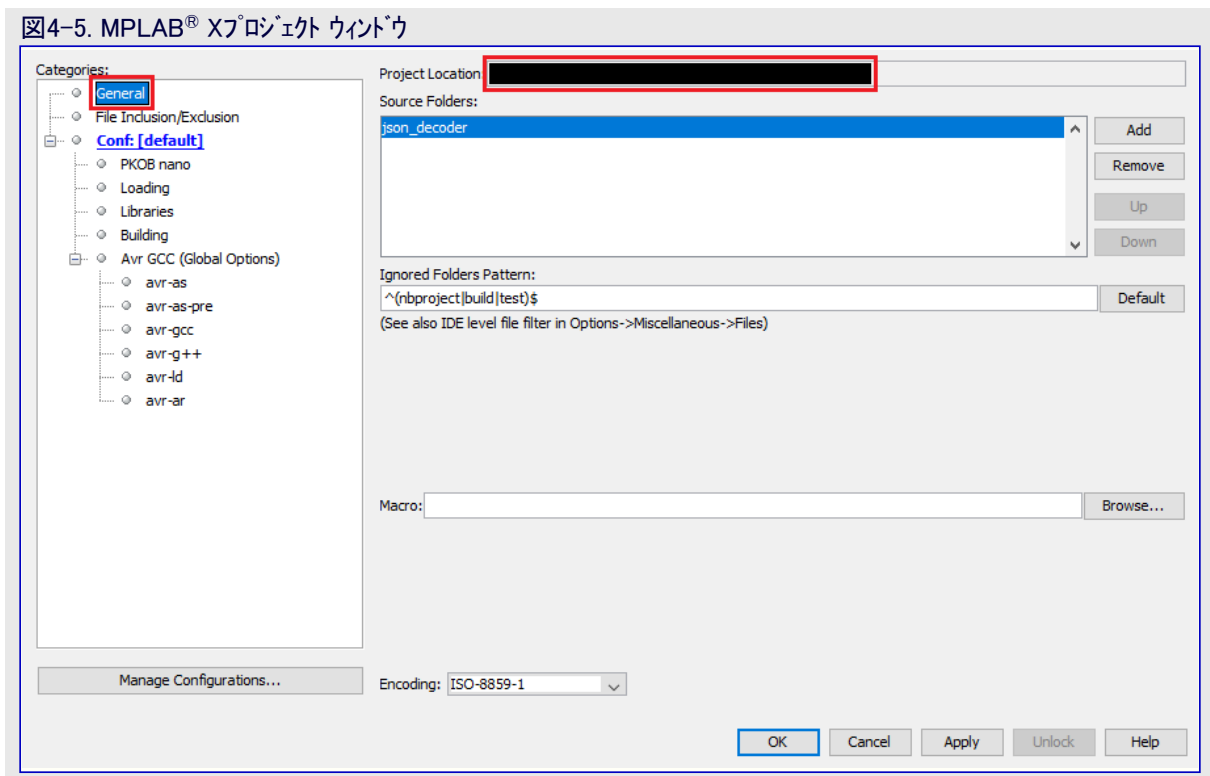
5. Device Resources(デバイス資源)からプロジェクトにUSART2を追加してください。
6. Pin Manager: Grid View(ピン管理部: 格子表示)ウィンドウでUSART2がPF0とPF1を使うことを確実にしてください。
7. USART2設定ウィンドウでPrintf support(printf支援)がチェックされ、他の全ての設定が既定であることを確実にしてください。



8. generateをクリックしてください。

4.1.2. プロジェクトにjson_decoderを追加

1. MPLAB Xプロジェクト ウィンドウでそれを右クリックすることによってプロジェクト パスを見つけ、その後にProperties(プロパティ)をクリックしてください。General(全般)タブの最初の要素がProject Location(プロジェクト位置)です。



2. `cmd`またはPower Shellのような命令行を使ってプロジェクトのルートに誘導してください。
3. 以下の命令を使ってプロジェクトにjson_decoder保管場所を複製してください。

```
$ git clone https://github.com/MicrochipTech/json_decoder.git
```

4. プロジェクト ウィンドウの現在のプロジェクトでSource Files(ソース ファイル)を右クリックしてNew logical folder(新しい論理フォルダ)を選んでください。
5. 作成したフォルダをjson_decoderに改名してください。
6. json_decoderフォルダを右クリックしてAdd Existing Item ...(既存項目追加...)を選んでください。json_decoder複製フォルダから全ての.c ファイルを追加してください。
7. Header files(ヘッダ ファイル)下に新しい論理フォルダを追加し、それをjson_decoderに改名してjson_decoder複製フォルダから全ての.h ファイルを追加してください。
8. main.cでjson_decoder/json_decoder.hをインクルードしてください。

```
#include "json_decoder/json_decoder.h"
```

9. Clean and Build Main Project(主プロジェクトを解消して構築)アイコンをクリックしてください。プロジェクトが成功裏に構築するでしょう。

4.1.3. 簡単な応用を作成

1. main.cでMAX_COMMAND_LENを定義してください。

```
#define MAX_COMMAND_LEN 64
```

2. while繰り返しの前にmain(主)関数で以下の変数を定義してください。

```
char command[MAX_COMMAND_LEN];
uint8_t index = 0;
char c;
```

3. while繰り返しに以下のコードを追加してください。

```
c = USART2_Read();
if(c != '\n' && c != '\r')
{
    command[index++] = c;
    if(index > MAX_COMMAND_LEN)
    {
        index = 0;
    }
}

if(c == '\n')
{
    command[index] = '\0';
    index = 0;
    executeCommand(command);
}
```

注: USARTについてもっと学ぶには「TB3216 – USARTでの開始に際して (DS90003216B)」をご覧ください。

4. main(主)関数の上にvoid executeCommand(char *command)関数を定義してください。
5. 関数の内側に於いて命令をJSONオブジェクトに復号してルートを取得してください。

```
jsonNode_t *root = 0;
jsonDecoderStatus_t ret;
char state[5];

ret = JSON_DECODER_fromString(command);
if(JSON_DECODER_OK != ret)
{
    printf("Invalid JSON string. %r\n");
}

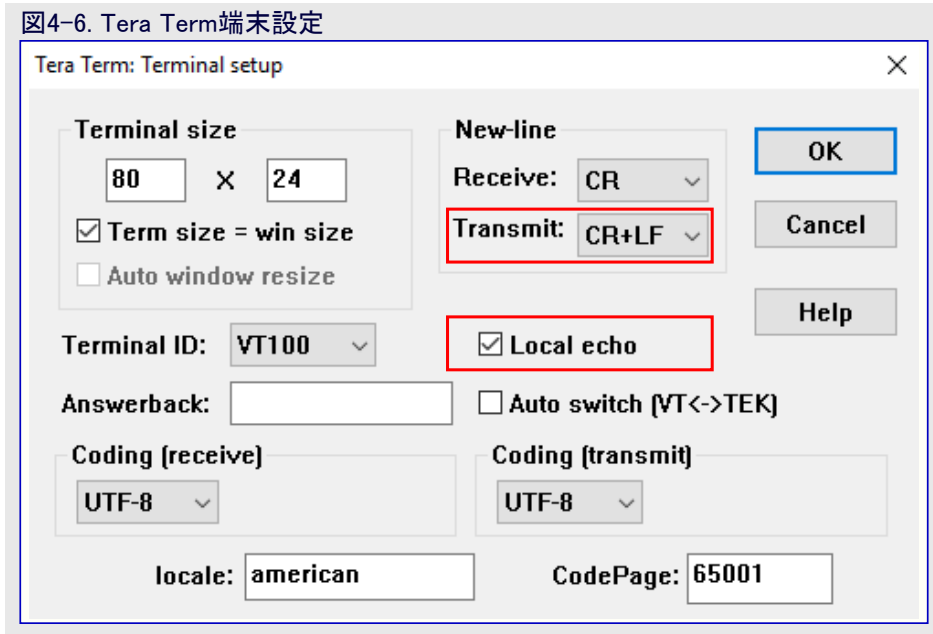
JSON_DECODER_getRoot(&root);
```


7. 他の3つのLEDに対して同様に行ってください。

注: executeCommand関数に対する完全なコードは6. 追補です。

4.1.4. 応用を試験

1. Tera Termのような端末模擬器を開いてください。
2. コンピュータに接続されたAVR-IoT-WGのCOMポートを選んでください。
3. Tera TermのメニューでSetup(準備設定)⇒Terminal(端末)へ行き、New-line(新規行)のTransmit:(送信:)をCR+LFに設定し、その後にLocal echo(送信も表示)を許可してください。



4. 以下のようなJSON文字列命令を送ってください。

```
{ "red" : " on" }
```

- 赤色LEDがONに切り替わります。

```
{ "blue" : " on" }
```

- 青色LEDがONに切り替わります。

```
{ "yellow" : " on" , "blue" : " off" }
```

- 黄色LEDがONに切り替わります。

- 青色LEDがOFFに切り替わります。

```
{ "yellow" : " on" , "blue" : " off" }
```

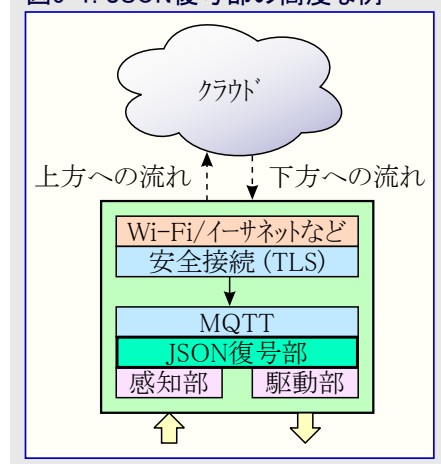
- 黄色と赤色のLEDがOFFに切り替わります。

- 青色と緑色のLEDがONに切り替わります。

5. 実演 : IoT 節点との汎用クラウド通信

IoT 節点との代表的なクラウド通信は安全な TLS 通信を渡って起こります。様々な応用規約がありますが、この筋書きで最も広く使われるのが MQTT (Message Queueing Telemetry Transport) です。実際のデータが一般的に JSON オブジェクトです。従って、JSON 復号単位部は IoT 節点での不可欠な部分です。右の画像はこのような応用の全体的な高位基本構造を説明します。

図5-1. JSON復号部の高度な例



6. 追補

ソフトウェア使用許諾契約

Microchip Technology Inc. (当社) によってここで供給されるソフトウェアは当社によって製造された製品とで単独且つ独占的に使うため、あなたや当社のお客様に対して意図されて供給されます。

このソフトウェアは当社やその供給者によって所有され、適用される著作権法下で保護されています。全ての権利が保留されています。前述の制限に違反するなどの使用も、適用される法律下で使用者を刑事処罰させ、更にこの許諾の契約条件の違反に対する民事責任を負う場合があります。

このソフトウェアは「現状のそのまま」の条件で提供されます。このソフトウェアに適用される特定の目的に対する商品性および適合性の黙示の保証を含みますが、これに限定されず、明示的、黙示的または法的な保証を含みません。当社は如何なる状況でも、特別な、偶発的または結果的な損害に対して如何なる理由でも責任を負いません。

```
void executeCommand(char *command)
{
    jsonNode_t *root = 0;
    jsonDecoderStatus_t ret;
    char state[5];

    ret = JSON_DECODER_fromString(command);
    if (JSON_DECODER_OK != ret)
    {
        printf("Invalid JSON string. %r\n");
    }

    JSON_DECODER_getRoot(&root);

    ret = JSON_DECODER_getString(root, "red", sizeof(state), state);
    if (JSON_DECODER_OK == ret)
    {
        if (strcmp(state, "on") == 0)
        {
            LED_RED_SetHigh();
        }
        else if (strcmp(state, "off") == 0)
        {
            LED_RED_SetLow();
        }
    }

    ret = JSON_DECODER_getString(root, "yellow", sizeof(state), state);
    if (JSON_DECODER_OK == ret)
    {
```

```
    if(strcmp(state, "on") == 0)
    {
        LED_YELLOW_SetHigh();
    }
    else if(strcmp(state, "off") == 0)
    {
        LED_YELLOW_SetLow();
    }
}

ret = JSON_DECODER_getString(root, "green", sizeof(state), state);
if(JSON_DECODER_OK == ret)
{
    if(strcmp(state, "on") == 0)
    {
        LED_GREEN_SetHigh();
    }
    else if(strcmp(state, "off") == 0)
    {
        LED_GREEN_SetLow();
    }
}

ret = JSON_DECODER_getString(root, "blue", sizeof(state), state);
if(JSON_DECODER_OK == ret)
{
    if(strcmp(state, "on") == 0)
    {
        LED_BLUE_SetHigh();
    }
    else if(strcmp(state, "off") == 0)
    {
        LED_BLUE_SetLow();
    }
}
}
```

Microchipウェブ サイト

Microchipは<http://www.microchip.com/>で当社のウェブ サイト経由でのオンライン支援を提供します。このウェブ サイトはお客様がファイルや情報を容易に利用可能にするのに使われます。利用可能な情報のいくつかは以下を含みます。

- **製品支援** – データシートと障害情報、応用記述と試供プログラム、設計資源、使用者の手引きとハードウェア支援資料、最新ソフトウェア配布と保管されたソフトウェア
- **一般的な技術支援** – 良くある質問(FAQ)、技術支援要求、オンライン検討グループ、Microchip設計協力課程会員一覧
- **Microchipの事業** – 製品選択器と注文の手引き、最新Microchip報道発表、セミナーとイベントの一覧、Microchip営業所の一覧、代理店と代表する工場

製品変更通知サービス

Microchipの製品変更通知サービスはMicrochip製品を最新に保つのに役立ちます。加入者は指定した製品系統や興味のある開発ツールに関連する変更、更新、改訂、障害情報がある場合に必ず電子メール通知を受け取ります。

登録するには<http://www.microchip.com/pcn>へ行って登録指示に従ってください。

お客様支援

Microchip製品の使用者は以下のいくつかのチャネルを通して支援を受け取ることができます。

- 代理店または販売会社
- 最寄りの営業所
- 組み込み解決技術者(ESE:Embedded Solutions Engineer)
- 技術支援

お客様は支援に関してこれらの代理店、販売会社、またはESEに連絡を取るべきです。最寄りの営業所もお客様の手助けに利用できます。営業所と位置の一覧はこの資料の後ろに含まれます。

技術支援は<http://www.microchip.com/support>でのウェブ サイトを通して利用できます。

Microchipデバイスコード保護機能

Microchipデバイスでの以下のコード保護機能の詳細に注意してください。

- Microchip製品はそれら特定のMicrochipデータシートに含まれる仕様に合致します。
- Microchipは意図した方法と通常条件下で使われる時に、その製品系統が今日の市場でその種類の最も安全な系統の1つであると考えます。
- コード保護機能を破るのに使われる不正でおそらく違法な方法があります。当社の知る限りこれらの方法の全てはMicrochipのデータシートに含まれた動作仕様外の方法でMicrochip製品を使うことが必要です。おそらく、それを行う人は知的財産の窃盗に関与しています。
- Microchipはそれらのコードの完全性について心配されているお客様と共に働きたいと思います。
- Microchipや他のどの半導体製造業者もそれらのコードの安全を保証することはできません。コード保護は当社が製品を”破ることができない”として保証するということを意味しません。

コード保護は常に進化しています。Microchipは当社製品のコード保護機能を継続的に改善することを約束します。Microchipのコード保護機能を破る試みはデジタル ミレニアム著作権法に違反するかもしれません。そのような行為があなたのソフトウェアや他の著作物に不正なアクセスを許す場合、その法律下の救済のために訴権を持つかもしれません。

法的通知

デバイス応用などに関してこの刊行物に含まれる情報は皆さまの便宜のためにだけ提供され、更新によって取り換えられるかもしれません。皆さまの応用が皆さまの仕様に合致するのを保証するのは皆さまの責任です。Microchipはその条件、品質、性能、商品性、目的適合性を含め、明示的にも黙示的にもその情報に関連して書面または表記された書面または黙示の如何なる表明や保証も**しません**。Microchipはこの情報とそれの使用から生じる全責任を否認します。生命維持や安全応用でのMicrochipデバイスの使用は完全に購入者の危険性で、購入者はそのような使用に起因する全ての損害、請求、訴訟、費用からMicrochipを擁護し、補償し、免責にすることに同意します。他に言及されない限り、Microchipのどの知的財産権下でも暗黙的または違う方法で許認可は譲渡されません。

商標

Microchipの名前とロゴ、Mmicrochipロゴ、Adaptec、AnyRate、AVR、AVRロゴ、AVR Freaks、BesTime、BitCloud、chipKIT、chipKITロゴ、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、HELDO、IGLOO、JukeBlox、KeeLoq、Kleer、LANCheck、LinkMD、maXStylus、maXTouch、MediaLB、megaAVR、Microsemi、Microsemiロゴ、MOST、MOSTロゴ、MPLAB、OptoLyzer、PacTime、PIC、picoPower、PICSTART、PIC32ロゴ、PolarFire、Prochip Designer、QTouch、SAM-BA、SenGenuity、SpyNIC、SST、SSTロゴ、SuperFlash、Symmetricom、SyncServer、Tachyon、TempTracker、TimeSource、tinyAVR、UNI/O、Vectron、XMEGAは米国と他の国に於けるMicrochip Technology Incorporatedの登録商標です。

APT、ClockWorks、The Embedded Control Solutions Company、EtherSynch、FlashTec、Hyper Speed Control、HyperLight Load、IntelliMOS、Liberio、motorBench、mTouch、Powermite 3、Precision Edge、ProASIC、ProASIC Plus、ProASIC Plusロゴ、Quiet-Wire、SmartFusion、SyncWorld、Temux、TimeCesium、TimeHub、TimePictra、TimeProvider、Vite、WinPath、ZLは米国に於けるMicrochip Technology Incorporatedの登録商標です。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BlueSky、BodyCom、CodeGuard、CryptoAuthentication、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、INICnet、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNetロゴ、memBrain、Mindi、MiWi、MPASM、MPF、MPLAB Certifiedロゴ、MPLAB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REALICE、Ripple Blocker、SAM-ICE、Serial Quad I/O、SMART-I.S.、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、View Sense、WiperLock、Wireless DNA、ZENAは米国と他の国に於けるMicrochip Technology Incorporatedの商標です。

SQTPは米国に於けるMicrochip Technology Incorporatedの役務標章です。

Adaptecロゴ、Frequency on Demand、Silicon Storage Technology、Symmcomは他の国に於けるMicrochip Technology Inc.の登録商標です。

GestICは他の国に於けるMicrochip Technology Inc.の子会社であるMicrochip Technology Germany II GmbH & Co. KGの登録商標です。

ここで言及した以外の全ての商標はそれら各々の会社の所有物です。

© 2020年、Microchip Technology Incorporated、米国印刷、不許複製

品質管理システム

Microchipの品質管理システムに関する情報については<http://www.microchip.com/quality>を訪ねてください。

日本語© HERO 2020.

本技術概説はMicrochipのTB3239技術概説(DS90003239A-2020年1月)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意識されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には()内に英語表記や略称などを残す形で表記しています。

青字の部分はリンクとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。

世界的な販売とサービス

米国	亜細亜/太平洋	亜細亜/太平洋	欧州
本社 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 技術支援: http://www.microchip.com/support ウェブアドレス: http://www.microchip.com	オーストラリア - シドニー Tel: 61-2-9868-6733 中国 - 北京 Tel: 86-10-8569-7000 中国 - 成都 Tel: 86-28-8665-5511 中国 - 重慶 Tel: 86-23-8980-9588 中国 - 東莞 Tel: 86-769-8702-9880 中国 - 広州 Tel: 86-20-8755-8029 中国 - 杭州 Tel: 86-571-8792-8115 中国 - 香港特别行政区 Tel: 852-2943-5100 中国 - 南京 Tel: 86-25-8473-2460 中国 - 青島 Tel: 86-532-8502-7355 中国 - 上海 Tel: 86-21-3326-8000 中国 - 瀋陽 Tel: 86-24-2334-2829 中国 - 深圳 Tel: 86-755-8864-2200 中国 - 蘇州 Tel: 86-186-6233-1526 中国 - 武漢 Tel: 86-27-5980-5300 中国 - 西安 Tel: 86-29-8833-7252 中国 - 廈門 Tel: 86-592-2388138 中国 - 珠海 Tel: 86-756-3210040	インド - ハンガロール Tel: 91-80-3090-4444 インド - ニューデリー Tel: 91-11-4160-8631 インド - フネー Tel: 91-20-4121-0141 日本 - 大阪 Tel: 81-6-6152-7160 日本 - 東京 Tel: 81-3-6880-3770 韓国 - 大邱 Tel: 82-53-744-4301 韓国 - ソウル Tel: 82-2-554-7200 マレーシア - クアラルンプール Tel: 60-3-7651-7906 マレーシア - ペナン Tel: 60-4-227-8870 フィリピン - マニラ Tel: 63-2-634-9065 シンガポール Tel: 65-6334-8870 台湾 - 新竹 Tel: 886-3-577-8366 台湾 - 高雄 Tel: 886-7-213-7830 台湾 - 台北 Tel: 886-2-2508-8600 タイ - バンコク Tel: 66-2-694-1351 ベトナム - ホーチミン Tel: 84-28-5448-2100	オーストリア - ウェルス Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 デンマーク - コペンハーゲン Tel: 45-4450-2828 Fax: 45-4485-2829 フィンランド - エスポー Tel: 358-9-4520-820 フランス - パリ Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 ドイツ - ガルピング Tel: 49-8931-9700 ドイツ - ハーン Tel: 49-2129-3766400 ドイツ - ハイムブロン Tel: 49-7131-72400 ドイツ - カールスルーエ Tel: 49-721-625370 ドイツ - ミュンヘン Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 ドイツ - ローゼンハイム Tel: 49-8031-354-560 イスラエル - ラーナナ Tel: 972-9-744-7705 イタリア - ミラノ Tel: 39-0331-742611 Fax: 39-0331-466781 イタリア - ハドバ Tel: 39-049-7625286 オランダ - デルネン Tel: 31-416-690399 Fax: 31-416-690340 ノルウェー - トロンハイム Tel: 47-72884388 ポーランド - ワルシャワ Tel: 48-22-3325737 ルーマニア - ブカレスト Tel: 40-21-407-87-50 スペイン - マドリッド Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 スウェーデン - イェテボリ Tel: 46-31-704-60-40 スウェーデン - ストックホルム Tel: 46-8-5090-4654 イギリス - ウォーキングム Tel: 44-118-921-5800 Fax: 44-118-921-5820
アトランタ Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 オースチン TX Tel: 512-257-3370 ホーストン Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 シカゴ Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 ダラス Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 デトロイト Novi, MI Tel: 248-848-4000 ヒューストン TX Tel: 281-894-5983 インディアナポリス Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 ロサンゼルス Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 ローリー NC Tel: 919-844-7510 ニューヨーク NY Tel: 631-435-6000 サンホセ CA Tel: 408-735-9110 Tel: 408-436-4270 カナダ - トロント Tel: 905-695-1980 Fax: 905-695-2078			