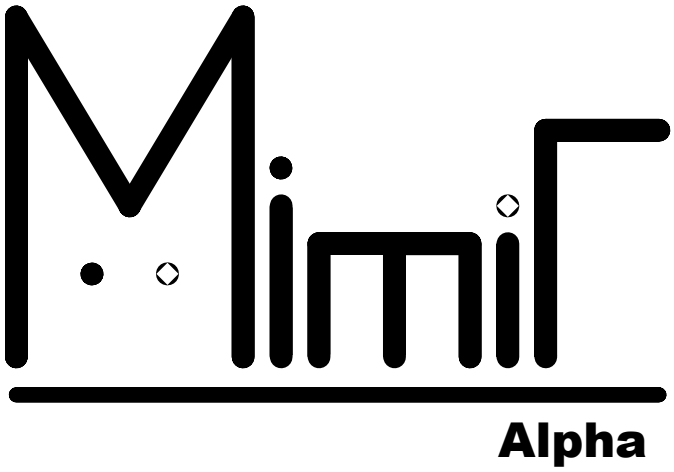
MiMic javascript API specification

2011.10.27 nyatla@nyatla.jp



本書は、MiMic javascript API の仕様書である。動作環境、提供するクラス、関数、定数について説明する。

Rev	日付	
1	2011/10/27	新規作成
2		

目次

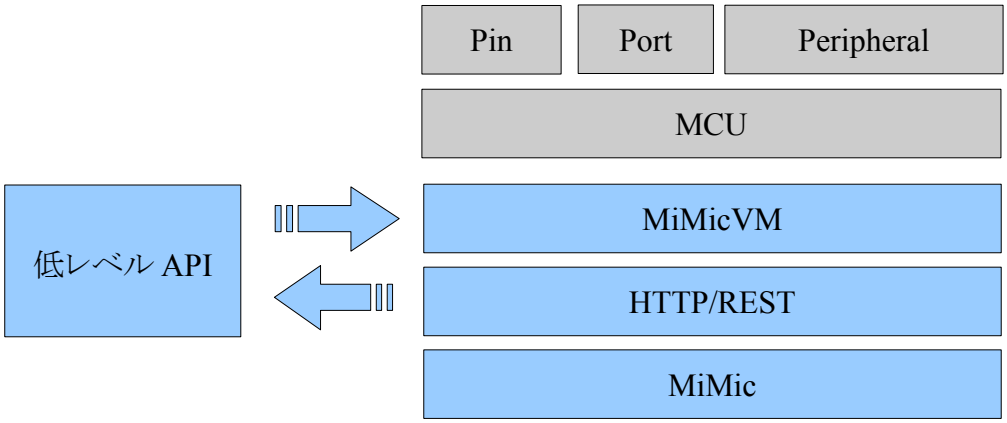
1 MiMic javascript API

MiMic javascript API は、MiMicRemoteMcu のリモート制御を目的とするプログラミングインタフェイスである。ウェブブラウザの Javascript 実行環境で動作し、低レベル API、高レベル API の2種類を提供する。

アプリケーション実装に使う API は、殆どの場合、高レベル API のみである。高レベル API のサポートしない機能を実装する場合は、低レベル API を使うことになる。

1.1 低レベル API

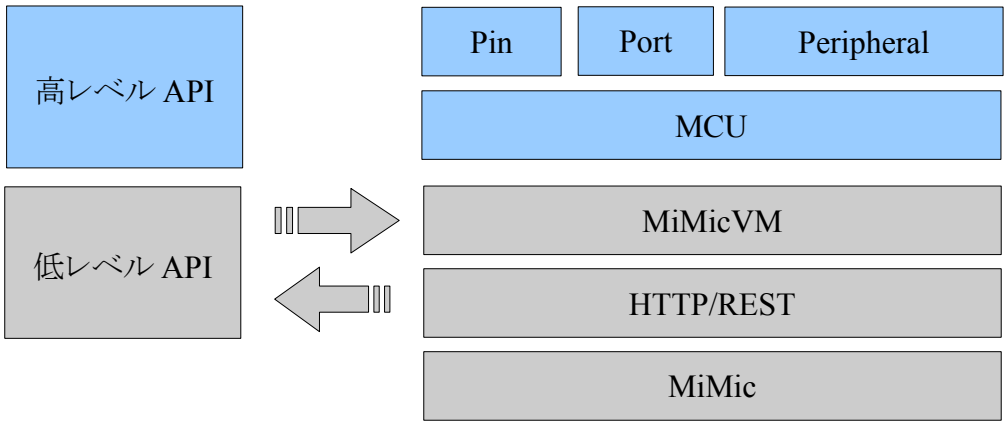
MiMicRemoteMCU の基本機能にアクセスするための API と、その管理関数群である。MiMic との通信層の制御、MCU を操作する為のバイトコード(MiMicBC)の転送機能を提供する。また MiMic javasctipt システムの例外処理機構、共通関数を定義する。



1.2 高レベル API

MiMicRemoteMCU は基本的に低レベル API の実行インタフェイスのみ提供する。低レベル API はアセンブラレベルの操作のみが可能なので、実際のピンやペリフェラル（周辺機器）を制御する為の高レベル API がある。

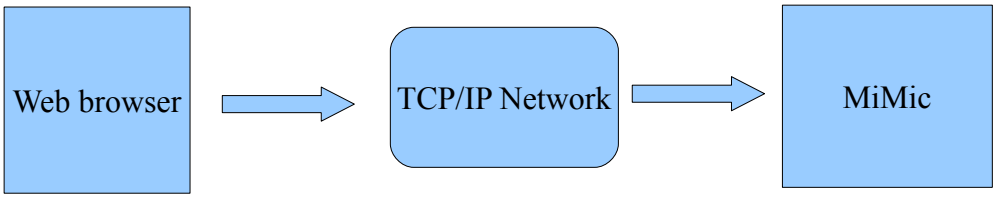
高レベル API は、ペリフェラルやその先にあるデバイス単位に定義し、それらを直感的に操作する関数を提供する。高レベル API は、実質、クライアントサイドで動作する、「ペリフェラルドライバ」である。低レベル API に入力する為の MCU の操作手順の組み立てと、返却データの解釈を行う。高レベル API は接続する MCU に依存する。本書では、LPCXpresso1769 に依存した高レベル API について説明する。



2 ネットワークトポロジー

MiMic javascript API は、ブラウザに搭載される javascript 実行環境で動作する。アプリケーションのソースファイルは、ブラウザからロードできる場所にあれば、何処に有ってもよい。

また、操作する MiMicRemoteMCU を搭載した LPCXpresso との通信は、全てクロスドメイン REST で行われる。ブラウザからアクセス可能ネットワークにあれば、LPCXpresso は何処に存在してもよい。※1



API と MCU の数の組み合わせは、自由である。ブラウザから複数の MCU を同時に制御することも、複数のブラウザから 1 台の MCU を操作することも可能である。※2　また、ブラウザを介して、MCU の制御するハードウェアと、既存の Web サービス、ブラウザの持つマルチメディア機能を連携させることも、自由である。

アプリケーションモデルについては、MiMic Application design.pdf で詳しく説明する。※3

なお、HTML5,XhtmlHttpRequest2 に対応しない InternetExproler、古いブラウザでは、MiMic はそのまま動作しない。代理サーバを立てるなどの工夫が必要である。

※1.広域ネットワークや 3G 回線等で RTT が低下すると制御が難しくなるので、注意すること。

※2.1MCU 対多クライアントモデルでは、MCU の前段にリクエストのシリアル化（仮想化）を行うプロキシ等を設置することを推奨する。MiMic は、REST 通信のみを行うので、PHP 等で開発が可能である。

※3.まだ書いてない。

MiMic は、ブラウザに対して特別な実装をすることなく、MCU に対するほぼ全ての制御機能を、提供する。開発者は、Web ページの作成と同じように、ハードウェア制御アプリケーションを作ることが出来る。アプリケーションは製造と同時にインターネットで共有可能であり、インターネットに存在する情報は、全て MiMic の管理するハードウェアと共有することが出来る。

3 用語

API 説明中にある、用語について説明する。

3.1 機能名

機能の名称である。”GPIO”や”AD”等がある。API 中では、文字列として表現する。

3.2 ピン機能名

MCU のピンに割り当てられた機能の名称である。API 中では、文字列として表現する。
LPCXpresso1769 の場合、この名称は、UM10360 の Chapter 8: LPC17xx Pin connect block にある、Pin function select register の Function name である。(GPIO のみ、GPIO Port 0.0 -> GPIO0.0 のように省略する。)

3.3 ペリフェラル名

論理ペリフェラル（物理ペリフェラルと 1 対 1 で対応しない）の名称である。API 中では、文字列として表現する。”GPIO”や”ADC”等がある。高レベル API のドライバクラスは、この名前ごとに定義する。

3.4 ピン識別子

物理ピンと1対1で対応する、識別値である。ピン 1 本単位に定義する。API 中では、オブジェクトとして表現する。

3.5 ペリフェラル識別子

物理ペリフェラルと1対1で対応する、識別値である。電源/クロック管理単位で定義する。API 中では、オブジェクトとして表現する。

4 低レベル API

4.1 MiMicCore.js

MiMic が内部的に使う標準関数 を格納した MiMicLib、エラー値テーブル MiMicError、例外クラス MiMicException、通信クラス MiMicRemoteMcuInterface を定義する。

4.1.1 依存モジュール

4.1.2 Public

4.1.2.1 MiMicLib

Type	Static Object
MiMic javascript API が使用する関数を定義する。 ユーザが使用してもかまわない。	
Definition	
var MiMicLib= { isUndef:function(a), } *isUndef:function(a)* a が undefined であるかを真偽値で返す。	
Sample	
-	

4.1.2.2 MiMicError

Type	Static Object

MiMicException の使用するエラーコードと、その解釈関数を定義する。

エラーコードは、[code:int,message:string]で表現する。
code:int
31bit のエラーコード。エラーコードのビットフィールドは以下の通り。

bit	description
30	Error:1,OK:0
29-24	Reserver
23-16	ModuleID 0x00:unknown 0x39:MiMic 0xF0-0xFF: user define Other:Reserved
15-8	ERROR_ClassID 0x00:unknown
7-0	ERROR_Code

Definition

```
var MiMicError=
{
	OK:[0x00000000,"OK"],
	NG:[0x40000000,"NG"],	//汎用 NG
	isOK:function(v){
		return (0x40000000 & v)==0x00000000;
	},
	MID_MiMic:0x00390000,
	CAID_RemoteMCU	:0x0100,	//
	CAID_LPCXPresso1769:0x0200	//リモート API
}
```

OK:object as [エラーコード]
成功を示すエラーコード。使うことは無い。

NG:object as [エラーコード]
失敗を示すエラーコード。不明な場所で、不明な何かが、不明なエラーを起こしたことを示す。

isOK:function(a)
a が失敗を示すエラーコードであるかを、真偽値で返す。

MID_MiMic,CAID_RemoteMCU,CAID_LPCXPresso:int
MiMicAPI が使う定数。

Sample

```
throw new MiMicException(MiMicError.NG);
```

4.1.2.3 MiMicException

Type	Constructor function
MiMic javascript API が生成する例外クラスのコンストラクタである。 関数ごとに MiMicException を使った try-catch を導入することにより、例外発生時にスタックトレースメッセージを得ることが出来る。	
Definition	
1	MiMicException() MiMicError.NG で例外を生成する。
2	MiMicException(e) 例外を引き継いで、例外を生成する。 *e:object as MiMicException*
3	MiMicException(o) o を格納した文字列とエラーコード MiMicError.NG から、例外を生成する。 *o:object* *オブジェクト。toString で文字列化する。*
4	MiMicException(s) o を格納した文字列とエラーコード MiMicError.NG から、例外を生成する。 *s:string* エラーメッセージ。
5	MiMicException(e,s) 例外を引き継いで、新しいメッセージを追加して例外を生成する。 e:object as MiMicException 引き継ぐ例外オブジェクトを指定する。 s:string エラーメッセージ。
6	MiMicException(c,s) *c:object as MiMicException* *s:string* エラーメッセージ。
Sample	
new MiMicException(“error!”); try{ //your program }catch(e){ throw new MiMicException(e); }	

4.1.2.4 MiMicException.alert

Type	Member function
格納している例外メッセージを alert 関数で表示する。 MiMicException の管理下にある例外は、関数のコールスタックを反映する。	
Definition	
1	alert()
Sample	

```
try{
 throw new MiMicException();
}catch(e){
 e.alert();
}
```

4.1.2.5 MiMicException.toString

Type	Member function
格納している例外メッセージを返す。 エラーメッセージは、関数コールスタックである。デバック等で役立つ。	
Definition	
1	toString()
Sample	

```
try{
 throw new MiMicException();
}catch(e){
 alert(e.toString());
}
```

4.1.2.6 MiMicRemoteMcuInterface

Type	Constructor function
MiMicRemoteMcuInterface クラスのコンストラクタ。 MiMicRemoteMcuInterface クラスは、MCU で動作する MiMicRemoteMcu との通信機能と、接続状態の監視機能を提供する。 低レベル API 全てを実装する。 低レベル API は、MiMicRemoteMcu との通信を、関数コールに変換する。 インスタンス生成直後は、MiMicRemoteMcu との接続は断状態である。connect 関数を実行して、接続する必要がある。 通信仕様については、MiMicVM.pdf Appendix 1. MiMicVM HTTP Interface を参照参照すること。 メンバ関数、メンバ変数は、MiMicRemoteMcuInterface.?のオブジェクトである。	
Definition	
1	MiMicRemoteMcuInterface(server) *server : [string]* 接続する MiMicRemoteMcu のホストアドレスを指定する。IP アドレス、ホスト名等。 ex. “127.0.0.1” , “127.0.0.1:3939”
Sample	
//create a controlable MCU via network. var mcu=new MiMicRemoteMcuInterface(“192.168.0.39”);	

4.1.2.7 MiMicRemoteMcuInterface.isConnected

Type	Member function
接続状態を真偽値で返す。 接続状態の場合、execBc 等の低レベル API を使用できる。 接続時のステータスは、インスタンスが定期的に実行するプローブ信号でチェックされ、更新される。	
Definition	
1	isConnected()
Return	
true/false MiMicRemoteMcu に接続しているなら、true	
Sample	
//show connection status var mri=new MiMicRemoteMcuInterface(“192.168.0.1”); alert(mri.isConnected());	

4.1.2.8 MiMicRemoteMcuInterface.execBc

Type	Member function

接続中の MiMicremoteMCU へ、整形済みの MiMicBC を送信する。
関数の利用前に、connect 関数で MiMicremoteMCU へ接続する必要がある。

何らかのエラーが発生して MiMicRemoteMCU からの応答が得られない場合、関数は例外を発生する。
この状況は、TCP/IP エラー、HTTP エラー、HTTP ステータスエラー、MiMicVM のラインタイムエラー、MiMicRemoteMCU のフォールト等の原因で発生する。

関数は同期実行であるので、RemoteMCU が応答しないと制御がブロックする。非同期関数が実装できるまお待ちください。

Definition

1	execBc(i_bc) *i_bc:string* 整形済みの MiMicBC 文字列。MiMicBC については、MiMicVM.pdf の、MiMicBC の章を参照。

Return

MiMicVM のパース済み JavascriptObject である。
形式は、
{version: string,result: int,stream int[]}である。
詳細は、MiMicVM.pdf Appendix 1. MiMicVM HTTP Interface を参照。

Sample

```
//show connection status
var mri=new  MiMicRemoteMcuInterface(“192.168.0.1”);
mri.connect(function(){});
mri.execBc(“ZAZZ.E”);//NOP;EXIT;
```

4.1.2.9 MiMicRemoteMcuInterface.connect

Type	Member function
MiMicRemoteMCU へ接続する。	
Definition	
1	execBc(i_callback) *i_callback:function(b)* 回線状態を定期的に通知するコールバックハンドラ。 disconnect 関数を呼び出すまでの間、回線の状態を定期的に受け取る。 b は接続状態を表す真偽値である。true の時、接続中である。false の場合、外部要因により切断されている。 false が通知されるのは、disconnect が呼び出されるまでに非同期切断を検出したときだけである。disconnect で切断した場合には呼び出されない。
Return	
-	
Sample	
//show connection status var mri=new MiMicRemoteMcuInterface(“192.168.0.1”); mri.connect(function(b){if(!b){alert(“disconnected!”);}}); //kill connection	

4.1.2.10 MiMicRemoteMcuInterface.disconnect

Type	Member function
接続中の MiMicRemoteMCU から切断する。接続されていなければ何もしない。	
Definition	
1	disconnect()
Return	
-	
Sample	
//connect and disconnect var mri=new MiMicRemoteMcuInterface(“192.168.0.1”); mri.connect(function(b){if(!b){alert(“disconnected!”);}}); mri.disconnect();	

5 高レベル API

5.1 LPCXpresso1769.js

名前空間 LPCXPresso1769 と、システム全体に対する識別子テーブル、関数を定義する。

5.1.1 依存モジュール

1. MiMicCore.js

5.1.2 Public

5.1.2.1 LPCXpresso1769.FUNC_NAME

Type	Static Array
機能名のテーブル。 機能名として使用できる文字列の一覧表である。機能名は、機能を識別する目的で定義する。 この値は一覧の定義のみであるので、使用することは無い。	
Definition	
FUNC_NAME:["GPIO","AD","DA"],	

GPIO	GPIO 機能を表す。
AD	AD convert 機能を表す。
DA	DA convert 機能を表す。

Sample
-

5.1.2.2 LPCXpresso1769.PHY_NAME

Type	Static Array
ペリフェラル名のテーブル。 ペリフェラル名として使用できる文字列の一覧表である。ペリフェラル名は、ペリフェラルを識別する目的で定義する。 この値は一覧の定義のみであるので、使用することは無い。	
Definition	
PHY_NAME:["GPIO","ADC","DAC"]	

GPIO	GPIO 機能を表す。
ADC	AD converter 機能を表す。
DAC	DA converter 機能を表す。

Sample
-

5.1.2.3 LPCXpresso1769.PHY

Type	Static Array
LPCXpresso の Peripheral に対応するテーブルである。要素はペリフェラル識別子である。 UM10360 の Chapter 4: LPC17xx Clocking and power control を元に定義している。 要素はオブジェクトであり、次の情報を格納する。 ピン識別子の要素は、ピンにアクセスするための物理パラメタである。	
Definition	
PHY:{*PHY_NAME*:[*PCLKSELreg#,PCLKSELbit,PCONPbit*]}	
Sample	
LPCXpresso1796.PHY.ADC //ADC peripheral	

5.1.2.4 LPCXpresso1769.P?[?]

Type	Static Array
PCXpresso の PIN に対応する定数テーブルである。要素は PIN 識別子である。 ピン識別子は、ピン P?[?]を、P0[0]のように表現する。これは、UM10360 の Table 73. Pin description の Symbol P?[?]に対応している。Pn[m]の n,m の使用できる値は、基本的に Table 73. Pin description に一致する。 PIN 識別子から値を取り出すときは、専用の関数を使用するべきである。 ピン識別子の要素は、ピンにアクセスするための物理パラメタである。	
Definition	
P0:[[*PINSELreg#*,*PINMODEreg#*,*PINMODEODreg#*,*bitidx*,[*ピン機能名*]]] • PINFUNC,PINMODE のビット位置は、(bitidx%16)*2 で計算する。ODReg については、そのままの数値を使う。 • [ピン機能名]は、PINSEL 設定値に対応するテーブルである。配列のインデクス値が、そのまま PINSEL の値に対応する。 • 設定できない値のフィールドは null である。ピン機能名は Pin Function Select register のテーブルの定義値そのもである。ただし、GPIO については"GPIO Port "を"GPIO"に省略していることに注意。	
Sample	
//1 LPCXpresso1796.P0[0]; //P0[0] LPCXpresso1796.P1[1]; //P0[01] //2 var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”); var gpiopin=mcu.getPin(LPCXpresso1769.P0[0],"GPIO");//bind P0[0] with GPIO function	

5.1.3 Public[System]

ユーザ向けではない公開関数。

- LPCXpresso1769.completePinFunctionName
- LPCXpresso1769.getPinRegInfo
- LPCXpresso1769.getPinSelByFunctionName
- LPCXpresso1769.hasPinFunctionName

5.2 LPCXpresso1769.Mcu.js

クラス LPCXpresso1769.Mcu を宣言する。

5.2.1 依存モジュール

1. MiMicCore.js
2. LPCXpresso1769.js

5.2.2 Public

5.2.2.1 LPCXPresso1769.Mcu()

Type	Constructor function
LPCXPresso1769.MCU(MCU)クラスのコンストラクタ。 MCU クラスは、物理 MCU をインスタンス化する。MiMicRemoteMcuInterface の管理機能と、MCU の物理機能への接続手段を定義する。 提供する機能を以下に示す。 • 物理 MCU との接続管理機能。 • MCU を元に作られた操作インスタンスの生存管理機能。 • MCU の持つ物理デバイスの操作インスタンス(Pin,Peripheral 等)の取得機能（インテリジェント接続）。 このクラスのインスタンスは、1 つの状態値を持つ • active インスタンスが物理 MCU を操作できるかを表すステータス値である。値は、true/false である。取得は、isActive 関数で行う。状態の変更は、activate,deactivate 関数で行う。変更イベントは、events.onActivateChanged イベントで通知する。 メンバ関数、メンバ変数は、LPCXPresso1769.Mcu.?のオブジェクトである。	
Definition	
1	Mcu(i_mimic_addr,[i_isactivate=true]) *i_mimic_addt : [string]* 接続する MiMicRemoteMcu のホストアドレスを指定する。IP アドレス、ホスト名等。 ex. “127.0.0.1” , “127.0.0.1:3939” *i_isactivate:[boolean]* インスタンス生成と同時に MiMicRemoteMcu との接続を確立するかのフラグ。省略時は true とみなす。true の場合、関数は即座に MiMicRemoteMCU に接続する。このとき、events.onActivateChanged をハンドリングすることが出来ないので注意すること。ハンドリングが必要なら、false を指定して、改めて activate 関数をコールする。
Sample	
//create a controlable MCU via network. var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”);	

5.2.2.2 LPCXpresso1769.Mcu.isActive

Type	Member function
Mcu のアクティブ状態を真偽値で返す。true の時、MCU が使用可能である。	
Definition	
1	function isActive()
Return	
return :[boolean] インスタンスが物理 MCU に接続しているかを返す。	
Sample	
var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”,false); mcu.events.onActivateChanged=function(f){ if(!f){alert(f)}; } mcu.activate();	

5.2.2.3 LPCXpresso1769.Mcu.events

Type	Member object
イベントハンドラの連想配列。メンバにイベントハンドラ関数を格納する。	
Definition	
1	events:{ onActivateChanged:null } *onActivateChange :[void] function(f:[boolean])* インスタンスのアクティブ状態が変化したときに呼び出されるイベントである。f パラメータには、状態変化後のアクティブ状態値が入る。このハンドラが呼び出されるのは、ユーザが状態を切り替えたときと、システムが状態を維持できなくなったとき（例えば MCU が応答しない）である。
Sample	
//show that MCU became active. var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”,false); mcu.events.onActivateChanged=function(f){ if(!f){alert(f)}; } mcu.activate();	

5.2.2.4 LPCXpresso1769.Mcu.activate

Type	Member function
インスタンスの状態をアクティブに切り替える。 コンストラクタで生成と同時にアクティブ化した場合は、コンストラクタ内で自動的にコールされる。 既にアクティブの場合、アクティブ化に失敗した場合に例外が発生する。	
Definition	
1	function activate()
Return	
-	
Sample	
// MCU will be active. var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”,false); mcu.activate();	

5.2.2.5 LPCXpresso1769.Mcu.deactivate

Type	Member function
インスタンスの状態を非アクティブに切り替える。 既に非アクティブの場合、例外が発生する。	
Definition	
1	function activate()
Return	
-	
Sample	
// MCU will be deactive. var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”); mcu.events.onActivateChanged=function(f){ if(!f){alert(f)}; } mcu.deactivate();	

5.2.2.6 LPCXpresso1769.Mcu.callMiMic

Type	Member function
MiMicBC をリンクしている MCU へ MiMic Low level API で同期送信し、結果を取得する。 MiMic Low level API の result 値が得られない場合、例外が発生する。この状況は、TCP/IP エラー、HTTP エラー、HTTP ステータスエラー、MiMicVM のラインタイムエラー、MiMicRemoteMCU のフォールト等の原因で発生する。 関数が値を返した場合は、MiMicRemoteMCU が処理を完了し、正しい形式のステータスを受信できた場合である。例外が発生した場合は、リンクが破壊されている場合がある。リンクが破壊された場合は、1 分以内にアクティブ状態が変化するので、それを検出して判断する。	

Definition	
1	function callMiMic(i_mimicbc) 整形済みの MiMicBC を送信する。整形済みの MiMicBC は、MiMicTXT,MiMicDB を連結した文字列である。固定命令を送信するときに使う。 *i_mimicbc:[string]* MiMicBC。MiMicBC のフォーマットは、MiMicVM.pdf MiMicBC で定義する。
2	function callMiMic(i_mimictxt,i_mimicdb) 整形済みの MiMicTXT と、数値配列の MiMicDB を連結して送信する。固定命令+パラメータで擬似関数を実現するときに便利である。 *i_mimictxt:string* MiMicTXT。MiMicTXT のフォーマットは、MiMicVM.pdf MiMicBC で定義する。 *mimicdb: Array[int]* 配列の INT 値。値は関数により MiMicDB に変換され、MiMicTXT に連結され、送信される。

Return
応答値を格納した連想配列である。 詳細は 低レベル API。MiMicRemoteMcuInterface.execBc 関数、または、MiMicVM.pdf Appendix 1. MiMicVM HTTP Interface を参照。差異がある場合は、MiMicVM.pdf を優先する。

Sample
//send 2 MiMic operations. var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”); mcu.callMiMic(“ZAZZ.E”,[]); //NOP .END mcu.callMiMic(“ZAZZ.E”);//NOP .END

5.2.2.7 LPCXpresso1769.Mcu.callMiMicWithCheck

Type	Member function
callMiMic 関数のラッパーである。 callMiMic 関数成功後に、result が成功(MVM_OK=0x0)であるかを確認し、それ以外であれば例外を発生させる。 result が 0 以外想定されない MiMicBC を実行するときに便利である。	
Definition	
1	function callMiMic(i_mimicbc) callMiMic 関数を参照。
2	function callMiMic(i_mimictxt,i_mimicdb) callMiMic 関数を参照。
Return	
callMiMic 関数を参照。	
Sample	
//send 2 MiMic operations. var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”); mcu.callMiMicWithCheck(“ZAZZ.E”,[]); mcu.callMiMicWithCheck(“ZAZZ.ENOOOO”);//exception!	

5.2.2.8 LPCXpresso1769.Mcu.getPin

Type	Member function

物理ピンを制御する Pin インスタンスを得る。
制御ペリフェラル、Pin を順に生成する方法と異なり、機能の実現に必要なインスタンスを自動的に準備できる。
Pin が生成できない場合、例外が発生する。

Definition

1	function getPin(i_pin_function_name) ピン機能名から、Pin インスタンスを生成する。 生成されたピンは、ピン機能名の割り当てられている物理ピンで実現される。 ピン機能名は、 LPCXpresso1769.P?[?]の[ピン機能名]で定義する文字列である。これは、UM10360 3.1.2.3 Chapter 8: LPC17xx Pin connect block の function name で定義される、PINSEL レジスタに設定する文字列と一致する。 *i_pin_function_name:string* ピン機能名の文字列。 ex. “GPIO0.0” ,”AD0.0” GPIO については、”GPIO Port 0.0”を”GPIO0.0”のように、省略すること。
2	function getPin(i_pin,i_function_name) ピン識別子と機能名から、機能を割り当てたピンを生成する。 組み合せによっては、ピンを生成できない。組み合わせは、UM10360 3.1.2.3 Chapter 8: LPC17xx Pin connect block を参照すること。 *i_pin: object as pin 識別子* ピン識別子。指定できるのは、LPCXpresso1796.P?[?]である。 *i_function_name:[string as 機能名* 機能名は文字列である。 指定できる文字列は、LPCXpresso1769.FUNC_NAME で定義する。

Return

返却される Pin インスタンスの型は、機能により異なる。機能と Pin インスタンスのクラスの対応は、以下の通りである。

- GPIO - LPCXpresso1769.GpioPin
- AD - LPCXpresso1769.AdcPin

Sample

```
var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”);
//create gpio pin at PIN0[0]
var gpiopin1=mcu.getPin("GPIO0.0");
//create gpio pin at PIN0[1]
var gpiopin2=mcu.getPin(LPCXpresso1769.P0[1],"GPIO");
//or  mcu.getPin(LPCXpresso1769.P0[1],LPCXpresso1769.FUNC_SYMBOL.GPIO);

//create AD at PIN0[23]
var adpin1=mcu.getPin(LPCXpresso1769.P0[23],"AD");
//or  mcu.getPin(LPCXpresso1769.P0[23],LPCXpresso1769.FUNC_SYMBOL.AD);

//create AD at PIN0[3] (exception)
var adpin2=mcu.getPin(LPCXpresso1769.P0[3],"AD");
```

5.2.2.9 LPCXpresso1769.Mcu.getPort

Type	Member function
ポート（複数ピンで構成するIO）を制御するPortインスタンスを得るする。 制御ペリフェラル、Portを順に生成する方法と異なり、機能の実現に必要なインスタンスを自動的に生成する。 Portが生成できない場合、例外が発生する。	
Definition	
1	
2	
Return	
-	
Sample	
-	

5.2.2.10 LPCXpresso1769.Mcu.getPeripheral

Type	Member function
ペリフェラル名から、ペリフェラルオブジェクトを得る。 関数は、初めて要求されたペリフェラルについては、オブジェクトを生成し、MCUインスタンスに保存する。同じ名前のペリフェラルが再度要求されると、過去に生成したペリフェラルを返す。	
Definition	
1	function getPeripheral(i_phy_name) ペリフェラル名に対応したペリフェラルオブジェクトを生成する。 *i_phy_name: string as ペリフェラル名* ペリフェラル名は文字列である。指定できる文字列は、LPCXpresso1769.PHY_NAMEで定義する。
Return	
返却されるペリフェラル型は、ペリフェラル名により異なる。機能とPinインスタンスのクラスの対応は、以下の通りである。 • GPIO - LPCXpresso1769.Gpio • ADC - LPCXpresso1769.Adc	
Sample	
//get GPIO peripheral var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”); var gpio=mcu.getPeripheral(“GPIO”);	

5.2.3 Public[System]

ユーザ向けではない公開関数。

- LPCXpresso1769.Mcu.registerPeripheral

5.3 LPCXpresso1769.Memory.js

クラス LPCXpresso1769.Memory を宣言する。

5.3.1 依存モジュール

1. MiMicCore.js
2. LPCXpresso1769.js
3. LPCXpresso1769.Mcu.js

5.3.2 Public

5.3.2.1 LPCXPresso1769.Memory

Type	Constructor function
LPCXPresso1769.Memory (Memory)クラスのコンストラクタ。 Memory クラスは、MCU のメモリ空間へアクセスする手段を提供する。 このクラスは、メモリアクセス対してなんら保護機能を持たない。MCU のメモリマップに十分に注意する必要がある。 メンバ関数、メンバ変数は、LPCXPresso1769.Memory.?のオブジェクトである。	
Definition	
1	Memory(i_mcu,[i_base=0x0]) *i_mcu : object as LPCXpresso1769.Mcu* インスタンスを結びつける Mcu オブジェクト。 *i_base: uint* メモリにアクセスするときのベースアドレス。省略可能である。省略時は 0x0 とみなす。4 バイト境界でなければならない。クラスの提供する関数でアドレスを指定した時には、全てこの値が加算される。
Sample	
//create a controlable MCU via network. var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”); var memory=new LPCXpresso1769.Memory(mcu);	

5.3.2.2 LPCXPresso1769.Memory.read32

Type	Member function
メモリから値を読み出して、値セットの配列、又は値を返す。 関数はバイト単位のアライメントでメモリにアクセスする。メモリアドレス、取得サイズは 4 バイト境界に一致させなければならない。	
Definition	
1	function read32(i_offset) 指定したアドレスの値を 32 ビット単位で取得する。 *i_offset:[int]* コンストラクタで指定したアドレスからのオフセット位置を指定する。4 バイト境界でなければならない。
2	function read32(i_offset,i_size) 指定したオフセットから、i_size バイトのメモリに格納した値を、それぞれ 32 ビット単位で値を取得する。i_offset の位置から、32bit 単位で i_size/4 個の値を取得することになる。シーケンシャルアクセスに使用する。 *i_offset:[int]* コンストラクタで指定したアドレスからのオフセット位置を指定する。4 バイト境界でなければならない。 *i_size:[int]* 取得するバイト数を指定する。4 バイト単位でなければならない。 返却される値の個数は、i_size/4 個になることに注意すること。
3	function read32(i_offsets) オフセットの配列要素に対応した値を、それぞれ 32 ビット単位で取得する。 ランダムアクセスに使用する。 *i_offsets: array[int]* オフセットアドレスの配列。それぞれ 4 バイト境界でなければならない。 返却される値の個数は、i_offset の長さと同じになる。
Return	
返却値の個数により異なる。 取得した値が 1 個の場合、int である。 2 個以上の場合、int 配列[int]である。	
Sample	

```
var   mcu=new LPCXpresso1769.Mcu("192.168.0.39");
var mem=new LPCXpresso1769.Memory(mcu);   //create instance
alert(mem.read32(0x00000000)); //show 1 value

mem.read32(0x0,8); //return 2 values.
mem.read32([0x0,0x14,0x08] //return 3 values by specified order.
var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”);
```

5.3.3 Public[System]

ユーザ向けではない公開関数。

5.4 LPCXPresso1769.Pin.js

クラス LPCXpresso1769.Pin を宣言する。

5.4.1 依存モジュール

1. MiMicCore.js
2. LPCXpresso1769.js
3. LPCXpresso1769.Mcu.js

5.4.2 Public

5.4.2.1 LPCXpresso1769.Pin

Type	Constructor function
LPCXPresso1769.Pin (Pin)クラスのコンストラクタ。 ピン識別子を元に、MCU に関連付けらした Pin インスタンスを生成する。 Pin クラスは、MCU の物理ピン単位に、操作インタフェイスを定義する。 PINSEL,PINMODE,PINMODE_OD レジスタを管理する。 このクラスは、上位クラスへピンの基本的な操作機能を提供する為のものであり、ユーザが直接使用することは（あまり）ない。 この関数は、MiMic の管理しているピン（ENET_?）も操作することが出来るが、操作してしまうと MiMicRemoteMcu とのコネクションが破壊されるので、注意すること。 メンバ関数、メンバ変数は、LPCXPresso1769.Pin.?のオブジェクトである。	
Definition	
1	Pin(i_mcu,i_pin,[i_opt=undefined]) *i_mcu : object as LPCXpresso1769.Mcu* インスタンスを結びつける Mcu オブジェクト。 *i_pin: object as ピン識別子* 生成するピンのピン識別子。 *i_opt: object* インスタンス生成と同時に setOpt 関数を実行する場合に使用するパラメタ。省略時は無視する。 詳細は setOpt 関数を参照。
Sample	
//create pin instance at P0[0] var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”); var pin=new LPCXpresso1769.Pin(mcu,LPCXpresso.P0[0]);	

5.4.2.2 LPCXpresso1769.Pin.setOpt

Type	Member function
パラメータを元に、ピンコンフィギュレーションを実行する。	
Definition	
1	function setOpt(i_opt) ペリフェラル名に対応したペリフェラルオブジェクトを生成する。 *i_opt:object as {sel,mode,od}* ピンコンフィグレーションのパラメタである。必要な値を格納した連想配列で指定する。 全ての値を省略することは出来ない。 *sel* 2bit の int 値。 PINSEL? レジスタに指定する値。UM10360 Chapter8.LPC17xx Pin connect block を参照。 *mode* 1bit の int 値。PINMODE? レジスタに指定する値。UM10360 Chapter8.LPC17xx Pin connect block を参照。 *od* 1bit の bit 値 PINMODE_OD? レジスタに指定する値。UM10360 Chapter8.LPC17xx Pin connect block を参照。 関数は、レジスタの位置に合わせてパラメータ値を自動的にシフトする。
Return	
-	
Sample	
//set GPIO,mode=1,open drain=0 var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”); var pin=new LPCXpresso1769.Pin(mcu,LPCXpresso.P2[3]); pin.setOpt({sel:0,mode:1,od:0});	

5.4.3 Public[System]

ユーザ向けではない公開関数。

- LPCXpresso1769.Pin.BCF_setOpt

5.5 LPCXPresso1769.Peripheral.js

クラス LPCXpresso1769.Peripheral を宣言する。

5.5.1 依存モジュール

1. MiMicCore.js
2. LPCXpresso1769.js
3. LPCXpresso1769.Mcu.js

5.5.2 Public

5.5.2.1 LPCXpresso1769.Peripheral

Type	Constructor function
LPCXPresso1769.Peripheral (Peripheral)クラスのコンストラクタ。 ペリフェラル識別子を元に、MCU に関連付けした Peripheral インスタンスを生成する。 Peripheral クラスは、物理 Peripheral（主に電源/クロックブロック単位）単位に操作インタフェイスを定義する。 PCLKSEL,PCONP レジスタを管理する。 このクラスは、上位クラスへペリフェラルの基本的な操作機能を提供する為のものであり、ユーザが直接使用することは（あまり）ない。 この関数は、MiMic の管理しているペリフェラル（ENET）も操作することが出来るが、操作してしまうと MiMicRemoteMcu とのコネクションが破壊されるので、注意すること。 メンバ関数、メンバ変数は、LPCXPresso1769.Peripheral.?のオブジェクトである。	
Definition	
1	Pin(i_mcu,i_phy,[i_opt=undefined]) *i_mcu : [object as LPCXpresso1769.Mcu]* インスタンスを結びつける Mcu オブジェクト。 *i_phy: object as ペリフェラル識別子* 生成するペリフェラルのペリフェラル識別子。 *i_opt: object* インスタンス生成と同時に setOpt 関数を実行する場合に使用するパラメタ。省略時は無視する。 詳細は setOpt 関数を参照。
Sample	
//create GPIO pheripheral var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”); var pin=new LPCXpresso1769.Pheripheral(mcu,LPCXpresso.PHY.GPIO);	

5.5.2.2 LPCXpresso1769.Peripheral.setOpt

Type	Member function
パラメータを元に、ペリフェラルコンフィギュレーションを実行する。	

Definition	
1	function setOpt(i_opt) ペリフェラル名に対応したペリフェラルオブジェクトを生成する。 *i_opt:object as {power,clock}* ペリフェラルピンコンフィグレーションのパラメタである。必要な値を格納した連想配列で指定する。 全ての値を省略することは出来ない。 *power* 1bit の int 値。PCONP? レジスタに指定する値。Chapter 4: LPC17xx Clocking and power control を参照。 *clock* 2bit の int 値。PCLKSEL? レジスタに指定する値。Chapter 4: LPC17xx Clocking and power control を参照。 関数は、レジスタの位置に合わせてパラメータ値を自動的にシフトする。

Return
-

Sample
//set DAC power on var mcu=new LPCXpresso1769.Mcu("192.168.0.39"); var pin=new LPCXpresso1769.Peripheral(mcu,LPCXpresso.PHY.ADC); pin.setOpt({power:1});

5.5.3 Public[System]

ユーザ向けではない公開関数。

- LPCXpresso1769.Peripheral.BCF_setOpt

5.6 LPCXPresso1769.Adc.js

クラス LPCXpresso1769.Adc と、LPCXpresso1769.AdcPin を宣言する。

5.6.1 依存モジュール

1. MiMicCore.js
2. LPCXpresso1769.js
3. LPCXpresso1769.Mcu.js
4. LPCXpresso1769.Peripheral.js
5. LPCXpresso1769.Pin.js

5.6.2 Public

5.6.2.1 LPCXpresso1769.Adc

Type	Constructor function
LPCXPresso1769.Adc (Adc)クラスのコンストラクタ。 MCU に関連付けした AD ペリフェラルを生成する。 AD ペリフェラルは、MCU の ADC ペリフェラル全体を管理する。 メンバ関数、メンバ変数は、LPCXPresso1769.Adc.?のオブジェクトである。 ハードウェア的には、ADC ペリフェラルはバーストモードで動作する。サンプリングレートは 200KHz 固定である。取得タイミングの制御はハードウェア依存である。	
Definition	
1	Adc(i_mcu,[i_opt={phy:{power:1}}]) *i_mcu : object as LPCXpresso1769.Mcu* インスタンスを結びつける Mcu オブジェクト。 *i_opt: object* インスタンス生成と同時に setOpt 関数を実行する場合に使用するパラメタ。省略時は、{phy:{power:1}}とみなす。 詳細は setOpt 関数を参照。
Sample	
//create AD (logical)pheripheral var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”); var ad=new LPCXpresso1769.Adc(mcu);	

5.6.2.2 LPCXpresso1769.Adc.getPin

Type	Member function

AD 機能を持つピンを取得する。
ピン識別子で指定されるピンを AD ペリフェラルと結合して、AdcPin を生成する。

関数は、AdcPin オブジェクトのコンストラクタをコールして、AdcPin を生成する。失敗すると、例外をスローする。
生成ルールについての詳細は、AdcPin を参照。

注意:インスタンス制御をしていない為、呼び出すたびに新しいオブジェクトが生成される。

Definition

1	getPin(i_pin,[i_opt]) *i_pin : object as ピン識別子* AD 機能を割り当てる PIN の識別子である。 *i_opt: object* AdcPin のコンストラクタに渡すオプション値を指定する。省略時は無視する。詳細は AdcPin のコンストラクタを参照。

Return

AdcPin クラスのオブジェクトである。

Sample

```
//create AdcPin
var mcu=new LPCXpresso1769.Mcu("192.168.0.39");
var adc=new LPCXpresso1769.Adc(mcu);
var adpin=adc.getPin(LPCXpresso1769.P0[23]);
```

5.6.2.3 LPCXpresso1769.AdcPin

Type	Constructor function
LPCXPresso1769.AdcPin (AdcPin)クラスのコンストラクタ。 Adc ペリフェラルオブジェクトにピン識別子で指定されたピンを関連付けて、AD 機能を持つピンを生成する。 関数は、ピン識別子を元に、そのピンが AD 機能に接続できるかを調べる。ピンに AD 機能を割り当てられない場合、例外が発生する。どのピンに AD 機能が割り当てられるかは、MCU のスペックシートを参照すること。 ピンが AD 機能を持たない場合、例外が発生する。 メンバ関数、メンバ変数は、LPCXPresso1769.AdcPin.?のオブジェクトである。	
Definition	
1	AdcPin(i_adc,i_pin,[i_opt]) *i_adc : object as LPCXpresso1769.Adc* インスタンスを結びつける Adc オブジェクト。 *i_opt: object* インスタンス生成と同時に setOpt 関数を実行する場合に使用するパラメタ。現在は指定できないので省略すること。
Sample	
//create AD0.0 var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”); var adc=new LPCXpresso1769.Adc(mcu); var adcpin=new LPCXpresso1769.AdcPin(adc,P0[23]);	

5.6.2.4 LPCXpresso1769.AdcPin.getValue

Type	Member function
ピンに入力されているサンプリング値を、12bit 整数で返す。	
Definition	
1	getValue()
Return	
12bit の AD 変換値を返す。値の意味は、UM10360 Chapter 29: LPC17xx Analog-to-Digital Converter (ADC)を参照。	
Sample	
//show value of AD0.0 pin var mcu=new LPCXpresso1769.Mcu("192.168.0.39"); var pin=mcu.getPin("AD0.0"); alert(pin.getValue());	

5.6.3 Public[System]

ユーザ向けではない公開関数。

- LPCXpresso1769.Adc.BCF_setSel
- LPCXpresso1769.Adc.BCF_getAD0DR

5.7 LPCXPresso1769.Gpio.js

クラス LPCXpresso1769.Gpio と、LPCXpresso1769.GpioPin を宣言する。

5.7.1 依存モジュール

1. MiMicCore.js
2. LPCXpresso1769.js
3. LPCXpresso1769.Mcu.js
4. LPCXpresso1769.Peripheral.js
5. LPCXpresso1769.Pin.js

5.7.2 Public

5.7.2.1 LPCXPresso1769.Gpio

Type	Constructor function
LPCXPresso1769.Gpio (Gpio)クラスのコンストラクタ。 MCU に関連付けした Gpio ペリフェラルを生成する。 GPIO ペリフェラルは、物理的には存在しない仮想ペリフェラルである。GPIO を集中管理するために定義している。 メンバ関数、メンバ変数は、LPCXPresso1769.Gpio.?のオブジェクトである。	
Definition	
1	Gpio(i_mcu) *i_mcu : object as LPCXpresso1769.Mcu* インスタンスを結びつける Mcu オブジェクト。
Sample	
//create AD (logical)pheripheral var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”); var gpio=new LPCXpresso1769.Gpio(mcu);	

5.7.2.2 LPCXpresso1769.Gpio.getPin

Type	Member function

GPIO 機能を持つピンを取得する。
ピン識別子で指定されるピンを GPIO ペリフェラルと結合して、GpioPin を生成する。

関数は、GpioPin オブジェクトのコンストラクタをコールして、GpioPin を生成する。失敗すると、例外をスローする。
生成ルールについての詳細は、GpioPin を参照。

注意:インスタンス制御をしていない為、呼び出すたびに新しいオブジェクトが生成される。

Definition

1	getPin(i_pin,[i_opt]) *i_pin : object as ピン識別子* GPIO 機能を割り当てる PIN の識別子である。 *i_opt: object* GpioPin のコンストラクタに渡すオプション値を指定する。省略時は無視する。詳細は GpioPin.setOpt 関数を参照。

Return

AdcPin クラスのオブジェクトである。

Sample

```
//create GpioPin direction=out
var mcu=new LPCXpresso1769.Mcu("192.168.0.39");
var gpio=new LPCXpresso1769.Gpio(mcu);
var pin=gpio.getPin(LPCXpresso1769.P0[0],{dir:1});
```

5.7.2.3 LPCXPresso1769.GpioPin

Type	Constructor function

LPCXPresso1769.GpioPin (GpioPin)クラスのコンストラクタ。
Gpio ペリフェラルオブジェクトにピン識別子で指定されたピンを関連付けて、GPIO 機能を持つピンを生成する。

関数は、ピン識別子を元に、そのピンが GPIO 機能に接続できるかを調べる。ピンに GPIO 機能を割り当てられない場合、例外が発生する。どのピンに GPIO 機能が割り当てられるかは、MCU のスペックシートを参照すること。

ピンが GPIO 機能を持たない場合、例外が発生する。

メンバ関数、メンバ変数は、LPCXPresso1769.GpioPin.?のオブジェクトである。

Definition

1	GpioPin(i_gpio,i_pin,[i_opt]) *i_gpio : object as LPCXpresso1769.Gpio* インスタンスを結びつける Gpio オブジェクト。 *i_opt: object* インスタンス生成と同時に setOpt 関数を実行する場合に使用するパラメタ。詳細は setOpt 関数を参照。

Sample

```
//create GPIO Port 0.0 dir=out
var mcu=new LPCXpresso1769.Mcu(“192.168.0.39”);
var gpio=new LPCXpresso1769.Gpio(mcu);
var pin=new  LPCXpresso1769.GpioPin(gpio,P0[0],{dir:1});
```

5.7.2.4 LPCXpresso1769.GpioPin.setOpt

Type	Member function

パラメータを元に、ピンのコンフィギュレーションを行う。
設定項目は、Gpio 固有のものと、Pin 共通のものがある。

Definition

```
1  function setOpt(i_opt)

   i_opt:object as {dir,pin{pin オプション}}
   ペリフェラルピンコンフィグレーションのパラメタである。必要な値を格納した連想配列で指定する。
   全ての値を省略することは出来ない。

   dir
     1bit の int 値。 FIO_DIR レジスタの設定値である。output=1,input=0
   pin
    Pin.setOpt 関数に渡すパラメータ。sel パラメタについては関数が上書きするので、無視される。詳細は Pin.setOpt 関数を
   参照。

   関数は、レジスタの位置に合わせてパラメータ値を自動的にシフトする。
```

Return

-

Sample

```
//set P0[0] to output  GPIO and mode=repeter and open drain=1
var mcu=new LPCXpresso1769.Mcu("192.168.0.39");
var gpiopin=mcu.getPin(LPCXpresso1769.P0[0],"GPIO");
gpiopin.setOpt({dir:1,pin:{mode:1,od:0}});
gpiopin.setValue(1);
```

5.7.2.1 LPCXpresso1769.GpioPin.getValue

Type	Member function
ピンの入力状態を 1/0 で返す。 この関数は、direction を 0(input)に設定したピンで使用できる。output に設定したピンには使用できない。	
Definition	
1	getValue()
Return	
1,又は 0	
Sample	
//show P0[0] value var mcu=new LPCXpresso1769.Mcu("192.168.0.39"); var gpiopin=mcu.getPin(LPCXpresso1769.P0[0],"GPIO"); gpiopin.setOpt({dir:0,pin:{mode:0,od:0}}); alert(gpiopin.getValue());	

5.7.2.2 LPCXpresso1769.GpioPin.setValue

Type	Member function
ピンの出力状態を設定する。 この関数は、direction を 1(output)に設定したピンで使用できる。input に設定したピンには使用できない。	
Definition	
1	setValue(i_val) *i_val :int* 1 ビットの出力値。
Return	
-	
Sample	
//set P0[0] pin to “on”. If LED was connected pin it will turn on. var mcu=new LPCXpresso1769.Mcu("192.168.0.39"); var gpiopin=mcu.getPin(LPCXpresso1769.P0[0],"GPIO"); gpiopin.setOpt({dir:1,pin:{mode:1,od:0}}); gpiopin.setValue(1);	

5.7.2.3 LPCXpresso1769.GpioPin.outPatt

Type	Member function
ピンにビットパターンを出力する。 単純なビットパターンをピンに出力するときに使用する。 出力レートは MCU 依存であり、コントロールできない。 この関数は、direction を 1(output)に設定したピンで使用できる。input に設定したピンには使用できない。	
Definition	
1	outPatt(i_val_array) *i_val_array:Array[int]* ビットパターンの配列。1 ビットの値(0 or 1)の配列を指定する。最大数は 20 である。
Return	
-	
Sample	
//output 0101010100001010 to P0[0] var mcu=new LPCXpresso1769.Mcu("192.168.0.39"); var gpiopin=mcu.getPin(LPCXpresso1769.P0[0],"GPIO"); gpiopin.setOpt({dir:1,pin:{mode:1,od:0}}); gpiopin.outPatt([0,1,0,1,0,1,0,1,0,0,0,0,1,0,1,0]);	

5.7.3 Public[System]

ユーザ向けではない公開関数。

- LPCXpresso1769.Gpio.BCF_setDir
- LPCXpresso1769.Gpio.BCF_setValue
- LPCXpresso1769.Gpio.BCF_getValue

6 Sample

MiMicRemoteMCU を搭載した LPCXpresso1769 で LED2 を点滅させるのならば、以下の html をブラウザで開けばよい。html を開くと、LED2 が点滅を開始する。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<script src="MiMicCore.js"></script>
<script src="LPCXpresso1769.All.js"></script>
<script type="text/javascript">
function init(){
      var mcu=new LPCXpresso1769.Mcu("192.168.0.39");
      var pin=mcu.getPin(LPCXpresso1769.P0[22],"GPIO");
      pin.setOpt({dir:1,pin:{mode:1,od:0}});
      var i=0;
      setInterval(function(){pin.setValue((i++)%2);},1000);
}
</script>
</head>
<body onload="init();">
<h1>LED flash</h1>
</body>
```

7 文献

- MiMicVM.pdf
- lpcxpresso.lpc1769.schematic.pdf
  http://ics.nxp.com/support/documents/microcontrollers/pdf/lpcxpresso.lpc1769.schematic.pdf
- UM10360.pdf
  http://www.nxp.com/documents/user_manual/UM10360.pdf