

# MSX0 入門

## 目次

1. はじめに.....	3
2. IoT BASIC.....	4
3. Grove デバイス.....	5
4. IoT 命令.....	7
4.1. IOTPUT.....	7
4.2. IOTGET.....	7
4.3. IOTFIND.....	8
4.4. IOTINIT.....	8
5. デバイスノード.....	9
6. 制御例.....	10
6.1. バッテリー電流/レベルを取得する.....	10

# 1. はじめに

MSX0が登場しましたが、SNSを眺めてみると「どうやって使ったら良いか分からない」「なにができるかわからない」という声を見かけました。せっかく登場したMSX0、確かに解説書の類いが付属のドキュメントのみ。付属のドキュメントも従来のMSX用のものがメインで、MSX0から追加された内容の説明は必要最小限のレベル。技術書などを読み慣れている人が、なんとか実験などを繰り返せば使えるようになる、といった情報量なので、当然ながら技術書を読み慣れていない人など、多くの一般人には「難解な代物」に見えるのかもしれません。本書は、そういった方達に向けて、もう少しだけ分かりやすく解説できればと思い執筆しました。私個人としては、MSX0向けのアプリケーションが増えて、MSX0が盛り上がって、そしてMSX全体が盛り上がる手助けになれば幸いです。

本書では、MSX0で追加になっている項目に絞って解説したいと思います。従来のMSXと互換の部分に関しては、MSX0に付属のBASIC解説書（3\_MSX-BASIC Version 2.0.pdfや4\_MSX-BASIC Version 3.0.pdf）などをお読みいただければと思います。特に、本書ではMSX-BASICをある程度使えることを前提としています。

## 2. IoT BASIC

MSX0 には、IoT BASIC と呼ばれる「MSX-BASIC に IoT 関連の機能を拡張する仕組み」が追加されています。起動すると、Fig.1 起動画面のような画面になります。

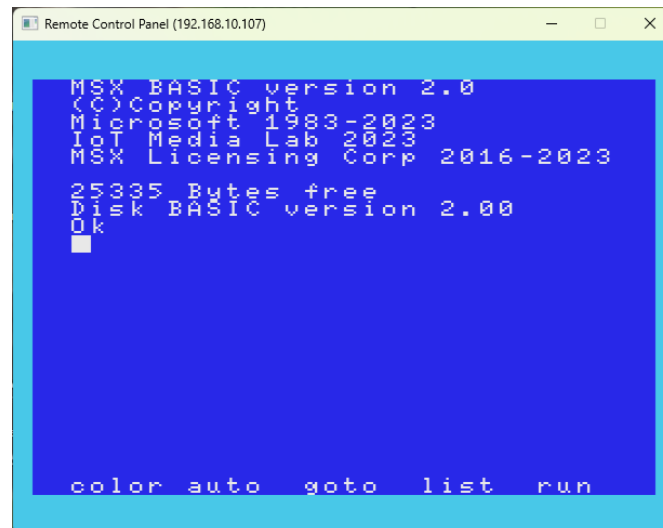


Fig.1 起動画面

従来の MSX を起動したときの表示と若干異なっています。IoT Media Lab 2023 の表示がありますね。この表示があると、IoT BASIC が使える状態になっています。「IoT BASIC が使える」とは、Grove デバイスの制御用命令が使えるということです。

IoT BASIC の追加命令はシンプルで、下記のコマンドのみになっています。後ほど一つずつ順番に説明していきます。

CALL IOTPUT( "デバイスノード", 値 )

CALL IOTGET( "デバイスノード", 変数名 )

CALL IOTFIND( "デバイスノード", 数値変数名 )

CALL IOTFIND( "デバイスノード", 配列変数名(0), 要素数 )

CALL IOTINIT()

命令の数は多くなく、たったの 5 種類です。これらは Grove デバイスに作用するのですが、命令の説明の前にまず Grove デバイスについて説明したいと思います。

### 3. Grove デバイス

MSX0 は、M5stack をベースにして、ハードウェアはほぼそのままに MSX エミュレーターをインストールしたものです。M5stack は、Grove 端子と呼ばれる接続端子を持っており、ここに接続したデバイスを制御したり、あるいはデバイスから得られる情報を読み取ることによって様々なことを行えるマシンです。M5stack をベースとする MSX0 にも Grove 端子が付いています。Grove 端子は、皆同じ形をしています種類があります。Fig.2 赤い Grove 端子や Fig.3 黒・水色の Grove 端子ですね。接続する Grove デバイス是对應する Grove 端子に接続する必要があります。



Fig.2 赤い Grove 端子



Fig.3 黒・水色の Grove 端子

赤い Grove 端子は、Port.A と呼ばれ、I<sup>2</sup>C（アイ スクエア シー）と呼ばれるインターフェースになっています。

黒い Grove 端子は、Port.B と呼ばれ、GPIO と呼ばれるインターフェースになっています。

水色の Grove 端子は、Port.C と呼ばれ、UART と呼ばれるインターフェースになっています。

「インターフェースってなんぞ？」と思う方もいるかもしれませんが、接続するデバイスとの対話の仕方みたいなものですね。赤いのは I<sup>2</sup>C というルールで繋がるぞ！ということです。赤い端子に、水色のデバイスを繋げても、I<sup>2</sup>C と UART では異なるルールなので、ちゃんと通信できない、ということですね。M5 用に売られているデバイス類は、接続コネクタに色が付いていますので、同じ色の端子に繋がれば良いです。

他のデバイス用に作られた Grove で同じ形状の端子のデバイスも存在しますが、I2C なのか GPIO なのか UART なのか調べて、ピンアサインが同じで、電圧も同じであれば対応する色の端子に接続して使えますが、ピンアサインとかよく分からないという人は、Fig.4 M5 用の Grove デバイスのような M5 のものを使うことをオススメします。スイッチサイエンス (<https://www.switch-science.com/>) から購入できます。



Fig.4 M5 用の Grove デバイス

## 4. IoT 命令

ここからは、各 IoT 命令について説明します。

CALL は、\_ と省略することが出来ます。CALL IOTGET は、\_IOTGET と書いても同じです。

### 4.1. IOTPUT

#### 【書式】

CALL IOTPUT( <デバイスノード> , <値> )

#### 【機能】

デバイスへ値を送信します。

<デバイスノード> を文字列で指定します。制御するデバイスを識別するために決められている文字列です。どのようなデバイスノードを指定できるかは、次章で説明します。

<値> は、指定のデバイスノードが示すデバイスへ送信する値です。数値を送信する場合は数値を、文字列を送信する場合は文字列を指定します。

デバイスノードによって、数値を送信するタイプと、文字列を送信するタイプとがありますので、デバイスノード表を参考に適切な方を指定して下さい。

### 4.2. IOTGET

#### 【書式】

CALL IOTGET( <デバイスノード> , <変数名> )

#### 【機能】

デバイスから値を取得します。

<デバイスノード> を文字列で指定します。制御するデバイスを識別するために決められている文字列です。どのようなデバイスノードを指定できるかは、次章で説明します。

<変数名> は、指定のデバイスノードが示すデバイスから受け取った値を格納する変数の名前です。数値を得たい場合は数値変数名、文字列を得たい場合は文字列変数名を指定します。

デバイスノードによって、数値が得られる場合と、文字列が得られる場合と異なりますので、デバイスノード表を参考に適切な方を指定して下さい。

## 4.3. IOTFIND

【書式 1】

CALL IOTFIND( <デバイスノード>, <数値変数名> )

【書式 2】

CALL IOTFIND( <デバイスノード>, <配列変数名>(0), <個数> )

【機能】

デバイスから複数の値をまとめて取得します。

<デバイスノード> を文字列で指定します。制御するデバイスを識別するために決められている文字列です。どのようなデバイスノードを指定できるかは、次章で説明します。

書式 1 は、書式 2 で取得できる個数を取得します。

書式 2 は、<個数>で指定の数だけ値を取得して、<配列変数名>で指定した配列変数に格納します。

要注意なのが、この命令はあまりメモリチェックが厳密ではないので、配列変数は必ず<個数>で指定する数と同じか、それ以上の要素数にしておく必要があります。取得する個数よりも配列要素数の方が少ないと、メモリを破壊してバグりますのでご注意ください。<個数>を省略すると最大数取得します。先ほどの理由から、省略はあまりオススメしません。

## 4.4. IOTINIT

【書式】

CALL IOTINIT()

【機能】

MSX0 では、MSX-BASIC の出力をターミナルに流し込むために、MSX-BASIC や MSX-DOS で使われる文字表示ルーチンに細工をしています。CALL KANJI 等を使うと、この細工が壊れることがあり、その修復のための再初期化命令です。



## 5. デバイスノード

IOT 命令は、IOTINIT() 以外の命令は全て第 1 引数に <デバイスノード> を指定するようになっています。<デバイスノード> は、複数接続したり、いろいろな種類のデバイスを接続するので、どのデバイスを制御するのかを指定するキーワードです。

MSX0 に付属のドキュメント「Fig.5 2\_IoT\_BASIC ノード情報.pdf」に一覧表があります。

ノード名	ノードタイプ	概要	単位	備考
host/battery/current	数値	バッテリー電圧	mA	ボトムベース調整時は測定不可
host/battery/level	数値	バッテリーレベル	%	
host/heap	数値	ヒープメモリ容量		
host/ip	文字列	IPアドレス		
host/media/disk/*	文字列	ファイル名		
host/name	文字列	ホスト名		
host/power/off	数値	システムシャットダウン		1を書き込むとシャットダウン
host/power/reboot	数値	システム再起動		1を書き込むとシステム再起動
host/power/wait	数値	シャットダウンから起動するまでの待ち時間	sec	
host/sw_version	文字列	システムソフトウェアバージョン		
host/wifi/aplist/*	数値	Wi-Fi アクセスポイントSSID		
host/wifi/level	数値	Wi-Fi RSSI		
host/wifi/start	数値	Wi-Fi 設定値反映		1を書き込むと設定値を反映し、必要に応じて再接続
device/accel/x	数値	加速度センサ X		バッテリーボトムなどDMUデバイス接続
device/accel/y	数値	加速度センサ Y		
device/accel/z	数値	加速度センサ Z		
device/analog/in	数値	アナログ入力 (12bit)		
device/analog/out	数値	アナログ出力 (12bit)		
device/dht/humidity	数値	湿度	%	DHT20を利用する場合は起動時からPortA1に接続しておいて下さい
device/dht/temperature	数値	温度	°C	
device/ds1307/*	バイト列	2c高層バスプロセッサメッセージボックス		
msx/me/drive/a	文字列	ファイル名		msx/meは動作しているUnit自身
msx/me/id	数値	ユニット番号		
msx/me/#/NET0/conf/addr	文字列	IPアドレス		
msx/me/#/NET0/conf/port	数値	ポート番号		
msx/me/#/NET0/connect	数値	接続状態、制御		
msx/me/#/NET0/msg	バイト列	メッセージボックス		
msx/me/kb	文字列	コンソール入力		
msx/me/pm/cpu/load	数値	Z80 CPUエミュレーション処理負荷	%	
msx/me/pm/cpu/percent	数値	Z80 CPUレート	%	
msx/me/pm/fps	数値	画面更新フレームレート	fps	
msx/me/pm/reboot	数値	エミュレータの再起動		1を書き込むとエミュレータの再起動
msx/u0/drive/a	文字列	ディスクイメージファイル名		
msx/u0/id	数値	ユニット番号		
msx/u0/#/NET0/conf/addr	文字列	IPアドレス		
msx/u0/#/NET0/conf/port	数値	ポート番号		
msx/u0/#/NET0/connect	数値	接続状態、制御		
msx/u0/#/NET0/msg	バイト列	メッセージボックス		
msx/u0/kb	文字列	コンソール入力		
msx/u0/pm/cpu/load	数値	Z80 CPUエミュレーション処理負荷	%	
msx/u0/pm/cpu/percent	数値	Z80 CPUレート	%	
msx/u0/pm/fps	数値	画面更新フレームレート	fps	
msx/u0/pm/reboot	数値	エミュレータの再起動		1を書き込むとエミュレータの再起動
msx/u1/drive/a	文字列	ディスクイメージファイル名		
msx/u1/id	数値	ユニット番号		
msx/u1/#/NET0/conf/addr	文字列	IPアドレス		
msx/u1/#/NET0/conf/port	数値	ポート番号		
msx/u1/#/NET0/connect	数値	接続状態、制御		
msx/u1/#/NET0/msg	バイト列	メッセージボックス		
msx/u1/kb	文字列	コンソール入力		
msx/u1/pm/cpu/load	数値	Z80 CPUエミュレーション処理負荷	%	
msx/u1/pm/cpu/percent	数値	Z80 CPUレート	%	
msx/u1/pm/fps	数値	画面更新フレームレート	fps	
msx/u1/pm/reboot	数値	エミュレータの再起動		1を書き込むとエミュレータの再起動
conf/bg_cpu_perf	数値	バックグラウンドUnitのZ80実行速度の割合		0.0%, 1.1%, 2.2%, 3.5%, 4.10%, 5.25%, 6.50%, 7.100%
conf/bg_and_mlx	数値	バックグラウンドUnitの音源出力のON/OFF		
conf/bk_color	数値	画面の背景の色を強制的に黒		
conf/brightness	数値	LCDバックライトの明るさ		0~10
conf/canridge_id	数値	Unit00のカードリッジ番号		0:Zanag, 1:Palanis
conf/disp_id	数値	メイン画面の出力先		0:OFF, 1:MSX00
conf/disk0	文字列	Unit00/Diskイメージ		
conf/disk1	文字列	Unit10/Diskイメージ		
conf/fdd_mode	数値	フロッピーモード		0:ノーマル, 1:左右反転
conf/ram_size_id	数値	RAMのサイズ		0:64K, 1:128K, 2:256K, 3:512K, 4:1M
conf/remote/enable	数値	リモート接続のON/OFF		
conf/save	数値	設定値保存		1を書き込むと設定値を保存
conf/setup	数値	SetupUtility起動		1を書き込むとSetupUtility起動
conf/sound/fixer	数値	音声出力フィルタのON/OFF		
conf/sound/volume	数値	音量		0~10
conf/status_monitor	数値	ステータスモニタの表示モード		0:OFF, 1:SIMPLE, 2:SIMPLE(S)
conf/system_type	数値	MSXシステムタイプ		0:MSX, 1:MSX2, 2:MSX2+
conf/ui/mfFaces_game	数値	MSPPーム1ビット接続設定		0:OFF, 1:KEYBOARD, 2:JOY1, 3:JOY2
conf/ui/rensha_a	数値	汎用ポート0の接続の選定		0:OFF, 1~3
conf/ui/rensha_b	数値	汎用ポート0の接続の選定		0:OFF, 1~3
conf/ui/touch_joypad	数値	タッチJoyPad接続設定		0:OFF, 1:KEYBOARD, 2:JOY1, 3:JOY2
conf/ui/touch_pad	数値	タッチJoyPad接続設定		0:OFF, 1:KEYBOARD, 2:JOY1, 3:JOY2
conf/version	数値	ソフトウェアバージョン		
conf/wifi/id	数値	Wi-Fiの接続先		0:OFF, 1~3
conf/wifi/net/1/pass	文字列	パスワード		
conf/wifi/net/1/ssid	文字列	SSID		
conf/wifi/net/2/pass	文字列	パスワード		
conf/wifi/net/2/ssid	文字列	SSID		
conf/wifi/net/3/pass	文字列	パスワード		
conf/wifi/net/3/ssid	文字列	SSID		

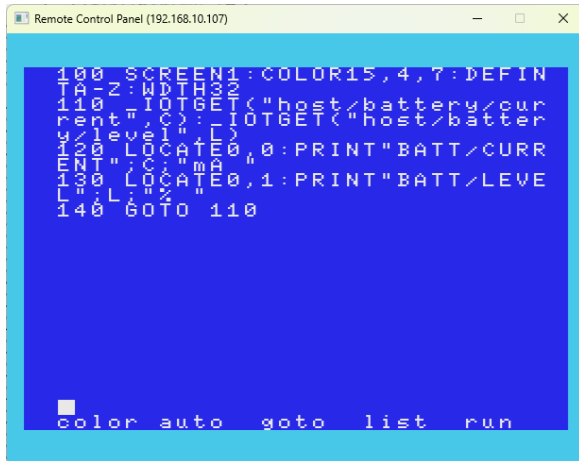
Fig.5 2\_IoT\_BASIC ノード情報.pdf

<デバイスノード> によって、IOTPUTのみ対応・IOTGETのみ対応・両方対応などの差があるのですが、残念ながら現状その記載はありません。概要に記載されている内容から推測したり、あるいは実際に試してみて確認することになります。

例えば、ノード名「host/battery/level」ですが、バッテリー残量を読み取るのみです。で、CALL IOTGET()のみ対応です。

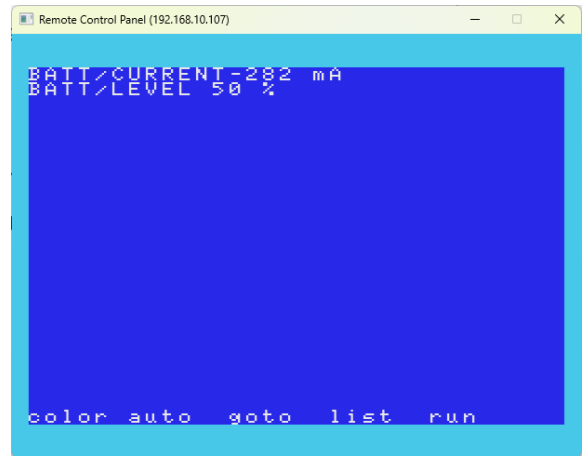
## 6. 制御例

### 6.1. バッテリー電流/レベルを取得する



```
100 SCREEN1:COLOR15,4,7:DEFIN
TA-Z:WIDTH32
110 _IOTGET("host/battery/cur
rent",C):_IOTGET("host/batter
y/level",L)
120 LOCATE0,0:PRINT"BATT/CURR
ENT";C;"mA "
130 LOCATE0,1:PRINT"BATT/LEVE
L";L;"% "
140 GOTO 110

color auto goto list run
```



```
BATT/CURRENT-282 mA
BATT/LEVEL 50 %

color auto goto list run
```

```
100 SCREEN1:COLOR15,4,7:DEFIN TA-Z:WIDTH32
110
_IOTGET("host/battery/current",C):_IOTGET("host/battery/level",L)
120 LOCATE0,0:PRINT"BATT/CURRENT";C;"mA "
130 LOCATE0,1:PRINT"BATT/LEVEL";L;"% "
140 GOTO 110
```

行110で デバイスパス "host/battery/current" から値を取得して変数 C へ、  
デバイスパス "host/battery/level" から値を取得して変数 L へ格納。  
行120~130 でそれらの値を表示。  
行140で行110へ戻る。

IOTGET で簡単にバッテリーレベルを得られるので、バッテリーレベルに応じて何か表示を変えたりすることも容易に実現できます。

