MSX0入門

目次

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

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。

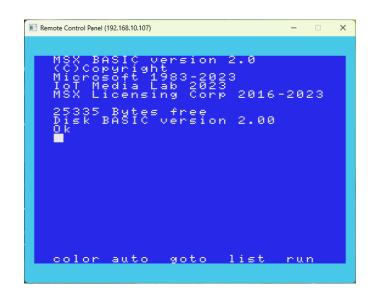


Fig.1 起動画面

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が種類があります。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 デバイスは対応する 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^2C (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 $\mathbf{I}^2\mathbf{C}$ 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、 $\mathbf{I}^2\mathbf{C}$ と \mathbf{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。 $\mathbf{M5}$ 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、同じ色の端子に繋げれば良いです。

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

【大書】

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

【た書】

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下さい。<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	磨薬	ME/O	備変
host/battery/current	整数	パッテリー電流	mA	ボトムベース装着時は測定不可
host/battery/level		パッテリー発量レベル	4	ALL SE - SCHOOL SELECTION AND ACTUAL
host/heap	整数整数	パッテリー残費レベル ヒーブメモリ残量	-	
	文字列	IP7FUA	-	
host/ip	义于列	IP/FUX	-	
host/media/dsk/*	-	ファイル名	-	
host/name	文字列	ホスト名		
host/power/off	整数	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 1を書き込むとシステム再起動
host/power/reboot	整数	システム再記動 シャットダウンから起動するまでの待ち時間		1を書き込むとシステム再起動
host/power/wait	整数	シャットダウンから記動するまでの待ち時間	sec	
host/sw_version	文字列	システムソフトウエアバージョン	100	
host/wifi/aplist/*	AT29	システムソフトウェアバージョン Wi-Fi アクセスポイントSSID Wi-Fi RSSI	-	
host (-15 /host	80-81-	HE EL DOOL	-	
host/wifi/level	整数	Wi-Fi RSSI	_	
host/wifi/restart	整数	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	整数整数	加速度センサX 加速度センサY		パッテリーボトムなどIMUデバイス接
device/accel/y	整数	加速度センサY		
device/accel/z	整数	加速度センサZ		
device/analog/in	整数	加速度センサZ アナログ入力(12bit)		
device/analog/out	整数	アナログ出力(12bit)		
				DHT20を利用する場合は起動時から
device/dht/humidity	整数	湿度	%	PortAに接続しておいて下さい
device/dht/temperature	整数	温度	°C	
device/i2c_i/*	バイト列	i2c通信パッファ(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	整数	ユニット番号		
msy/me/if/NET0/conf/adde	女学別	IPTELZ		
msx/me/if/NET0/conf/addr msx/me/if/NET0/conf/port	文字列 整数	IPアドレス ボート番号	_	
misx/me/f/NE1U/conf/port		1 H 7	-	
msx/me/if/NET0/connect	整数	(京都认取, 朝野	-	-
msx/me/if/NET0/msg	パイト列	接続状態。制御 メッセージボックス コンソール入力	-	
msx/me/kb	文字列	コンソール入力		
msx/me/pm/cpu/load	整数	Z80 CPUエミュレーション処理負荷	%	
msx/me/pm/cpu/percent	整数整数	780 CPUIL—b	%	
msx/me/pm/fps	務勘	国画画新フレー ムレート	fps	
msx/me/pm/reboot	10 No.	エミュレータの再起動 ディスクイメージファイル名	1,00	1を書き込むとエミュレータの再起動
msx/u0/drive/a	整数文字列	ニマスカノメージフェノルタ	-	IS BOX CCT CID - NO HARRIS
	大子列	71A71Z - 2271WG	+	
msx/u0/id	整数	ユニット番号	-	
msx/u0/if/NET0/conf/addr	文字列	IPアドレス	-	
msx/u0/if/NET0/conf/port	整数	ポート番号		
msx/u0/if/NET0/connect	整数	ホート番号 接続状態:制御 メッセージボックス コンソール人力		
msx/u0/if/NET0/msg	パイト列 文字列	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/opu/load	整数	Z80 CPUエミュレーション処理負荷	-	
msx/u0/pm/opu/percent	整数	Z80 CPUL-F	8	
	38 KA			
msx/u0/pm/fps	整数	国面更新フレームレート	fps	
msx/u0/pm/reboot	整數	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	国国里新プレームレート エミュレータの再起動 ディスクイメージファイル名		
msx/u1/id	整数	ユニット番号 IPアドレス		
msx/u1/if/NET0/conf/addr		IPアドレス		
msx/u1/if/NET0/conf/port msx/u1/if/NET0/connect	整数	ポート等号		
msx/u1/if/NET0/connect	務勘	接続状態 割割	-	
msx/u1/if/NET0/msg	17.71.70	ボート番号 接続状態、制御 メッセージボックス	_	
msx/u1/m/NE10/msg msx/u1/kb	バイト列 文字列	メッセーンホックス 	-	
	X 于 列	コンソール入力	-	
msx/u1/pm/cpu/load	整数	Z80 CPUエミュレーション処理負荷	-	
msx/u1/pm/opu/percent		Z80 CPULTトンヨン処理負荷 Z80 CPULTト	%	
msx/u1/pm/fps	整数	国面更新フレームレート エミュレータの再起動	fps	
msx/u1/pm/reboot	整数	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
				0:0%, 1:1%, 2:2%, 3:5%, 4:10%
conf/bg_cpu_perf	整数	パックグラウンドUnitのZ80実行速度の割合		5:25%, 6:50%, 7:100%
conf/bg_snd_mix	整数	バックグラウンドUnitの音声出力ののON/OFF	_	
conf/bg_snd_mix conf/bk_color	整数	国家の展別の各名社会的に関	-	
		画面の周辺の色を強制的に黒 LCDパックライトの明るさ	-	010
conf/brightness	整数	しいハックライトの明るさ	-	0~10
conf/cartridge_id	整数	UnitUUカートリッン番号	-	0:Zanac, 1:Paipanio 0:OFF, 1:M5LCD
conf/disp_id	整数	メイン画面の出力先		0:OFF, 1:M5LCD
conf/dsk0	文字列	Unit0のDiskイメージ		
conf/dsk1		Unit1のDiskイメージ		
conf/lcd_mode	整数	液晶モード		0:ノーマル、1:左右反転
conf/ram_size_id	整数	液晶モード RAMのサイズ		0:ノーマル、1:左右反転 0:64K、1:128K、2:256K、3:512K、4:1M
conf/remote/enable	100 Str	III 上海的(DON/OFF		The state of the s
conf/remote/enable conf/save	整数整数	リモート機能のON/OFF 設定値保存	_	1を書き込むと設定値を保存
	Bit No	Control to the state of	_	1を書き込むとSetupUtility記動
conf/setup	整数整数	SetupUtilitying	_	TE TO A C C Setup Utility (C S)
conf/sound/filter		音声出力フィルタののON/OFF	-	
conf/sound/volume	整数	音量 ステータスモニタの表示モード	-	0~10
		ステータスモニタの表示モード		0:OFF, 1:SIMPLE, 2:SIMPLE(S),
		MSXシステムタイプ		0:MSX, 1:MSX2, 2:MSX2+
conf/system_type	整数			0:OFF, 1:KEYBOARD, 2:JOY1, 3:JOY
conf/system_type	整数			0:OFF, 1~3
conf/system_type conf/ui/m5faces_game	整数	3月 田ポート4 の連 針の連さ		
conf/system_type conf/ui/m5faces_game conf/ui/rensha_a	整数	汎用ポートAの連射の速さ 3日ポートBの連転の速さ	_	
conf/system_type conf/ui/m5faces_game conf/ui/rensha_a conf/ui/rensha_b	整数整数	汎用ポートAの連射の速さ 3日ポートBの連転の速さ		0:OFF, 1~3
conf/system type conf/ui/m5faces game conf/ui/rensha_a conf/ui/rensha_b conf/ui/touch joyped	整数整数	汎用ポートAの連射の速さ 3日ポートBの連転の速さ		0:OFF, 1~3
conf/system_type conf/ui/m0faces_game conf/ui/rensha_a conf/ui/rensha_b conf/ui/touch_joypad conf/ui/touch_pad	整数 整数 整数 整数	汎用ボートAの連射の連さ 汎用ボートBの連射の連さ タッチJoyパッド接続設定 タッチパッド接続設定		
conf/system type conf/si/moffaces game conf/si/moffaces game conf/si/rensha_a conf/si/rensha_b conf/si/touch_joypad conf/si/touch_pad conf/version	整数 整数 整数 整数 整数	汎用ボートAの連射の連さ 汎用ボートBの連射の連さ タッチJoyパッド接続設定 タッチパッド接続設定		0:0FF, 1:KEYBOARD, 2:JOY1, 3:JOY 0:0FF, 1:KEYBOARD, 2:JOY1, 3:JOY
conf/system type conf/si/moffaces game conf/si/moffaces game conf/si/rensha_a conf/si/rensha_b conf/si/touch_joypad conf/si/touch_pad conf/version	整数 整数 整数 整数 整数	汎用ボートAの連幹の速さ 乳用ボートBの連幹の速さ タッチJoyパッド接続設定 タッチブッド接続設定 configurationパージョン Wi-Fiの接続先		0:0FF, 1:KEYBOARD, 2:JOY1, 3:JOY 0:0FF, 1:KEYBOARD, 2:JOY1, 3:JOY
conf/system type conf/ui/m6faces game conf/ui/m6faces game conf/ui/rensha a conf/ui/rensha b conf/ui/couh joypad conf/ui/touh joypad conf/ui/touh, game conf/ui/touh, game conf/ui/found	整数 整数 整数 整数 整数	汎用ボートAの連幹の速さ 乳用ボートBの連幹の速さ タッチJoyパッド接続設定 タッチブッド接続設定 configurationパージョン Wi-Fiの接続先		0:OFF, 1~3
confi/system.type confi/ui/m5faces.game confi/ui/m5faces.game confi/ui/m5ha.a confi/ui/rensha.b confi/ui/touch.joypad confi/ui/touch.joypad confi/ui/touch.joyd confi/	整数 整数 整数 整数 整数 整数 整数 字字字	系用ポートの連載の選さ 風用ポートの連載の選さ タッチムのようい接触数定 タッチムの計算機数定 configurationパージョン WFFIの基機先 バスワート SSID		0:0FF, 1:KEYBOARD, 2:JOY1, 3:JOY 0:0FF, 1:KEYBOARD, 2:JOY1, 3:JOY
conf./svstem type conf./ui/mofisces game conf./ui/mofisces game conf./ui/monsha a conf./ui/touch jayped conf./ui/touch jayped conf./ui/touch jay conf./wifi/id conf./wifi/id conf./wifi/id	整数 整数 整数 整数 整数 整数 整数 字字字	系用ポートの連載の選さ 風用ポートの連載の選さ タッチムのようい接触数定 タッチムの計算機数定 configurationパージョン WFFIの基機先 バスワート SSID		0:0FF, 1:KEYBOARD, 2:JOY1, 3:JOY 0:0FF, 1:KEYBOARD, 2:JOY1, 3:JOY
conf./system_type conf./u/mfsteces_game conf./u/mfsteces_game conf./u/msteces_a conf./u/renshe_a conf./u/renshe_b conf./u/touch_joyped conf./wfu/couch_pad conf./wfu/d conf./w	整数数 整整数数 整整数数 整整文字字 文字字列	別用ボートAの連載の速さ 別用ボートBの連載の速さ タッチよのパッド接続数定 タッチスのド・接続数定 configuration バージョン W・Fiの接続先 //スワード		0:0FF, 1:KEYBOARD, 2:JOY1, 3:JOY 0:0FF, 1:KEYBOARD, 2:JOY1, 3:JOY
conf/system type conf/si/mfaces_game conf/si/mfaces_game conf/si/msha_b conf/si/fouch_loyped	整整整整整整整整整整整整整整整整整整整整整整整整整整整整整整整整整整整整整	原用ボートAの連載の速と 及用ボートAの連載の速と タッチュの場合では、 タッチュのメントで接続数定 タッチュのメントで接続数定 configuration (ージョン WF-Fの技能を //スフード SSID //スフード		0:0FF, 1:KEYBOARD, 2:JOY1, 3:JOY 0:0FF, 1:KEYBOARD, 2:JOY1, 3:JOY
confi status monitor confi systam type confi su'i m'faces game confi su'i m'faces game confi su'i m'faces game confi su'i rensha, a confi su'i rensha, b confi su'i touch, pay confi su'i touch, pay confi su'i touch, pad confi su'i touch, pad confi su'i touch, pad confi su'i fait su'i fait s	整数数 整整数数 整整数数 整整文字字 文字字列	別用ボートAの連載の速さ 別用ボートBの連載の速さ タッチよのパッド接続数定 タッチスのド・接続数定 configuration バージョン W・Fiの接続先 //スワード		0:0FF, 1:KEYBOARD, 2:JOY1, 3:JOY 0:0FF, 1:KEYBOARD, 2:JOY1, 3:JOY

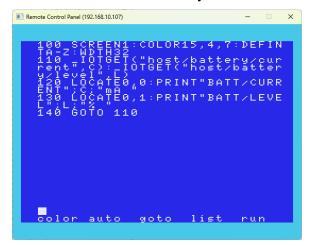
Fig.5 2_IoT_BASICノード情報.pdf

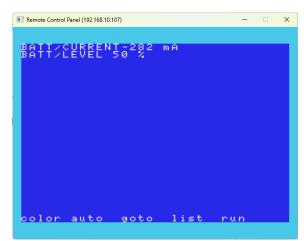
〈デバイスノード〉 によって、IOTPUT のみ対応・IOTGET のみ対応・両方対応などの差があるのですが、残念ながら現状その記載はありません。概要に記載されている内容から推測したり、あるいは実際に試してみて確認することになります。

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ので、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:DEFINTA-Z:WIDTH32
110
_IOTGET("host/battery/current",C):_IOTGET("host/battery/leve
l",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 $120\sim130$ 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。