MSX0 入門

目次

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。

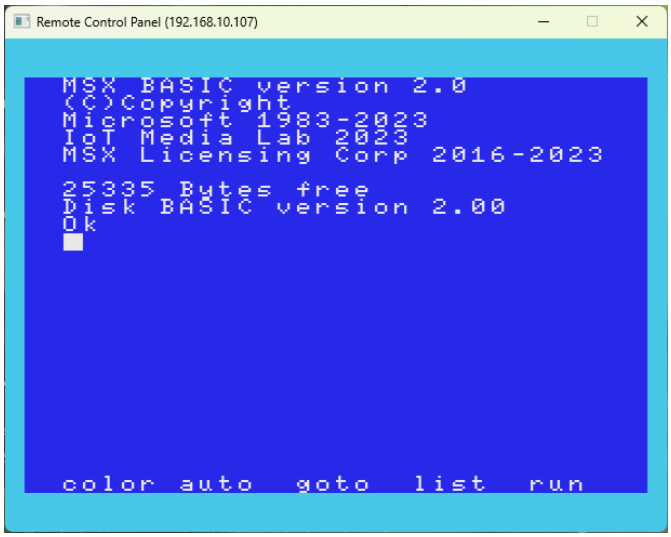

Fig.1 起動画面

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

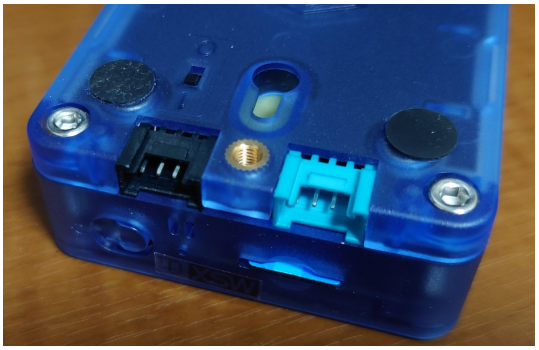

Fig.3 黒・水色の Grove 端子

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 I²C 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、I²C 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

【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

6. 制御例