

## ロボット頭脳開発支援



|    |           |     |
|----|-----------|-----|
| 発行 | 2003.5.30 | 第1版 |
| 発行 | 2008.8.30 | 第2版 |
| 発行 | 2015.3.31 | 第3版 |

(有) 加藤エムイーピーシステム

## 目 次

|      |                     |       |    |
|------|---------------------|-------|----|
| 1.   | 概要                  | ----- | 3  |
| 2    | MEPS コマンドジェネレーター    | ----- | 5  |
| 2.1  | コマンドメニューとエディット画面    | /     | 7  |
| 2.2  | タスク制御コマンド           | /     | 11 |
| 2.3  | ジャンプコマンド            | /     | 13 |
| 2.4  | 信号入出力コマンド           | /     | 15 |
| 2.5  | 信号待ちコマンド            | /     | 17 |
| 2.6  | タイマーコマンド            | /     | 18 |
| 2.7  | フリッカーコマンド           | /     | 19 |
| 2.8  | 通信、時計、アナログコマンド      | /     | 20 |
| 2.9  | I、R、レジスター操作、変換コマンド  | /     | 21 |
| 2.10 | I、レジスター操作コマンド       | /     | 23 |
| 2.11 | C、レジスター操作、変換コマンド    | /     | 25 |
| 2.12 | その他の特殊、関数コマンド       | /     | 27 |
| 2.13 | シフトメモリーコマンド         | /     | 28 |
| 2.14 | テーブル処理コマンド          | /     | 29 |
| 2.15 | ステップ動作コマンド          | /     | 31 |
| 2.16 | AND/ORゲートコマンド       | /     | 32 |
| 2.17 | キー入力表示コマンド          | /     | 33 |
| 2.18 | ファイルメニュー            | /     | 35 |
| 2.19 | 編集メニュー              | /     | 35 |
| 2.20 | 信号メニュー              | /     | 35 |
| 2.21 | レジスターメニュー           | /     | 36 |
| 2.22 | C文字レジスター (C00-C99)  | /     | 36 |
| 2.23 | I正整数レジスター (I00-I99) | /     | 36 |
| 2.24 | R実数レジスター (R00-R99)  | /     | 37 |
| 2.25 | オーバータイマー値 (00-99)   | /     | 37 |
| 2.26 | データメモリー (D00-D99)   | /     | 37 |
| 2.27 | コントローラーの条件メニュー      | /     | 38 |
| 2.28 | デバッグメニュー            | /     | 39 |
| 2.29 | 編集メニューからの検索、置換      | /     | 40 |
| 2.30 | 行表示からの編集            | /     | 40 |
| 2.31 | 各命令からの編集            | /     | 41 |
| 2.32 | 実行プログラムの作成と転送       | /     | 41 |

|      |                               |       |    |
|------|-------------------------------|-------|----|
| 3    | I/Oコンパイラ命令                    | ----- |    |
|      | 42                            |       |    |
| 3.1  | タスク命令                         | /44   |    |
| 3.2  | ジャンプ命令                        | /47   |    |
| 3.3  | I/O命令                         | /50   |    |
| 3.4  | タイマー命令                        | /54   |    |
| 3.5  | 通信                            | /55   |    |
| 3.6  | アナログ入出力                       | /55   |    |
| 3.7  | 時計入力                          | /55   |    |
| 3.8  | 演算                            | /56   |    |
| 3.9  | 特殊関数                          | /61   |    |
| 3.10 | メモリー操作                        | /62   |    |
| 3.11 | キー入力                          | /64   |    |
| 3.12 | LCD表示                         | /65   |    |
| 4    | MEPS コントローラー 外観、コネクタ、仕様       | ----- | 67 |
| 5.   | Meps Controller Help (MCH) 画面 | ----- | 74 |
| 5.1  | メニュー画面                        | /74   |    |
| 5.2  | キーヘルプ画面                       | /74   |    |
| 5.3  | コントロール画面                      | /75   |    |
| 5.4  | オブジェクトコード受信画面                 | /76   |    |
| 5.5  | フラッシュメモリー書き込み画面               | /76   |    |
| 5.6  | デバッグ画面                        | /77   |    |
| 5.7  | 設定画面                          | /77   |    |
| 5.8  | データ設定画面                       | /78   |    |
| 5.9  | タイマー値設定画面                     | /78   |    |
| 5.10 | 通信パラメーター設定画面                  | /79   |    |
| 5.11 | その他の通信パラメーター設定画面              | /80   |    |
| 5.12 | カレンダークロック設定画面                 | /80   |    |
| 5.13 | オプション設定画面                     | /81   |    |
| 5.14 | 表示スピード選択画面                    | /81   |    |
| 5.15 | チェック画面                        | /82   |    |
| 5.16 | 入出力チェック画面                     | /82   |    |
| 5.17 | アナログ入力チェック画面                  | /83   |    |
| 5.18 | アナログ出力チェック画面                  | /83   |    |
| 5.19 | 通信バッファ参照画面                    | /84   |    |
| 5.20 | その他のチェック画面                    | /85   |    |



# 1. 概要

ロボット等、I/O制御するものとして、ラダー図に基づいた処理装置があるが、これはハードウェアに近い方法で、片や純ソフト的に各種のコンパイラでI/O制御する方法は、多いI/O点数を処理するには不便だし、I/O専一に処理するものとしては勿体ない。

元来コンパイラは関数演算、ループ構造を伴う処理、ファイル操作、操作キー、画面出力等々色々な機能を時系列で処理するべく作られてきました。

だからI/Oに関してはラダー図シーケンシャル処理に勝をゆづって来ました。

ハードウェア技術者にとって設計回路に近い方法のラダー図シーケンシャル処理はなじみのあるものである。しかしラダー図の先頭から末尾までを一瞬にして実行を繰り返すものであるから、時系列に処理を記述するには相当熟練しなければならない。しかも、ステップが増えれば増えるほど算術級数的よりも幾何級数的に複雑度が增大して行きます。

過去幾多の未熟な技術者が泥沼に入って長期間、脱却出来なかったのを見て来ました。

こういった場合でも熟練技術者によると短期間で完成してしまい、その実力差が大きかった。

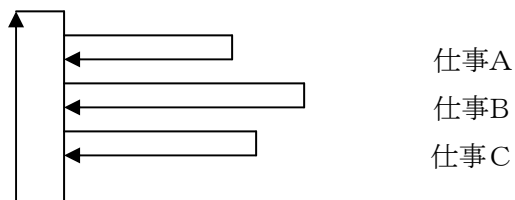
これは時系列に変化するシステムの、状態数の把握管理の仕方に個人的な個性、能力の差が出たのではないのでしょうか？その差をうめる方法がないものか、しかも早期に完成して信頼度も高く、誰でもメンテナンス出来る方法が無いかと、いつも問題意識をもっていました。

アセンブラーは、機械語に近く、システムに自由に適応させうるが、大変な労力を要するし、開発出来る人を探さねばいけません。

MEPS I/Oコンパイラは、実際このアセンブラーのマクロに近く、決して高級な言語では無いが、始めから並行処理を念頭に置き、無駄の無い開発と、期間短縮、信頼度向上をめざした、ソフトとハードを結合させる最良のものかと思います。

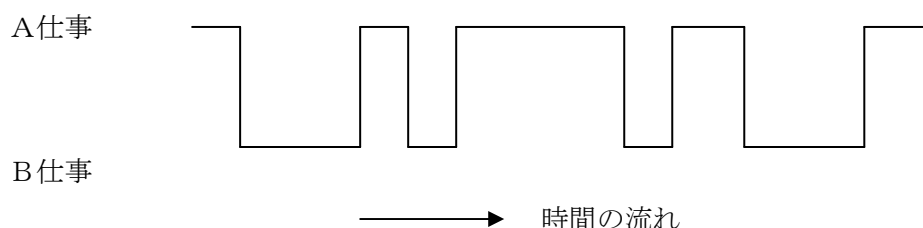
構造的には、タスクキャン処理と時系列をそれぞれの単一次元とした二次元モデルを厳然と構築する思想です。三次元以上の構造は、処理装置を別個に考えて、眼耳鼻舌身意コンピュータロボット等と、N次の多元分散処理として考えていけばよく、処理速度と、プログラム容量さえ問題にしなければ、二次元モデルに縮約できます。

二次元は、縦と横を考え、縦方向は、仕事を探し、あれば、即時実行すべく高速にループしていて、横方向は、時系列的に、それぞれの仕事の処理をしてゆくのです。



横方向一つをタスク（仕事の単位）として、マルチ（多重）タスクで処理をするということになります。

マルチタスクの概念の具体的な説明は、いわゆる、ながら族の話です。テレビ見ながらご飯を食べる。ただ少し違うのは同時進行ではなく、ご飯を口に入れる時、又はそしゃくする瞬間はテレビを見ないという風に、コンピューターは瞬間瞬間は一つの事しか出来ないのです、下図の様に



仕事をしている時間はA, B共、断続的ですが、何か待つたびに、積極的に仕事を切替（スイッチング）て、処理すれば効率的です。プログラム上待つ事が生じる点に切替（スイッチング）を用いた命令を書く事により、マルチタスクが可能となります。待つ事がない連続的即時処理の場合、強制的に切替を入れなければなりません。誤解のないようにしなければならぬ事として、本マルチタスクは自動的になされるのではなく自分で切替を設定して構築する事です。もし仮に切替を用いなければ最初のタスクを延々と実行するシングルタスクの状態になります。しかしこれが意図的であれば一向に差支えはありません。

M u l t i E a s y P r o g r a m m i n g S y s t e m 略してMEPSは、I/Oコンパイラーを使用して、開発したソフトウェアを、MEPSコントローラーで、実行させるために、PCパソコンWindowsで、支援させるもので、総合システムです。

PCでは、MEPS支援ソフトは、記述選択したものを、I/Oコンパイラー命令に変換したり、命令から逆変換したり（コマンドジェネレーター）、I/Oコンパイラー言語の編集（エディター）、コンパイルして、コントローラーへ、RS232Cで、転送したりします。又、この状態で、デバッグ出来ます（オンラインデバッガー）。

コントローラーでは、デバッグ後、フラッシュメモリーへ、書き込みをしたり、I/Oチェックや、メモリー確認、通信パラメーターの設定、カレンダー時刻の設定、タイマー定数の設定等々、モニターメニューにもとずいて、ユーザーアプリケーションプログラムとは別にヘルププログラムが管理しています。

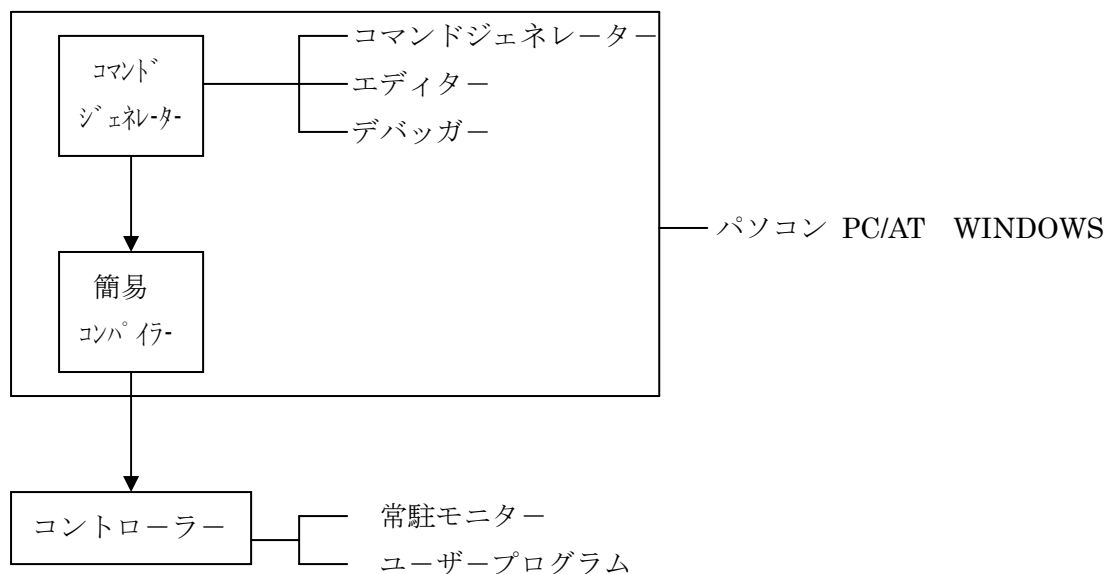
とにかく、MEPSは、命令をおぼえなくてもスピーディーかつ簡単にマルチ処理を作成させる事。プログラムの信頼度を向上させる事。MEPSコントローラーに手動調整画面を用意して、プログラマーの負担を軽減する事。これらを主旨として、長年の実践経験から製作したものです。以上、概要をのべましたが、以下、MEPSコマンドジェネレーター、MEPS I/Oコンパイラー、MEPSコントローラーと分けて説明します。

## 2. MEPS コマンドジェネレーター



上画面は開始画面です。

コマンドジェネレーターは図1の如く、コマンドジェネレーター、それに併う独自開発のエディター、デバッガーを含みます。コマンドジェネレーターはコマンドメニューにて実行します。エディターは編集メニュー、デバッガーはデバッグメニューにて実行します。エディターは、コピー、切り取り、貼り付け、検索、置換機能を持ちます。デバッガーは、デバッグモードでの実行（コントローラ-RAM上で実行）、カーソルの前まで実行（ブレイクポイント）、ステップ等の機能を持ちます。



【図 1】 MEPSシステムの構成図

コントローラー仕様：縦170mm × 横120mm × 高さ60mm

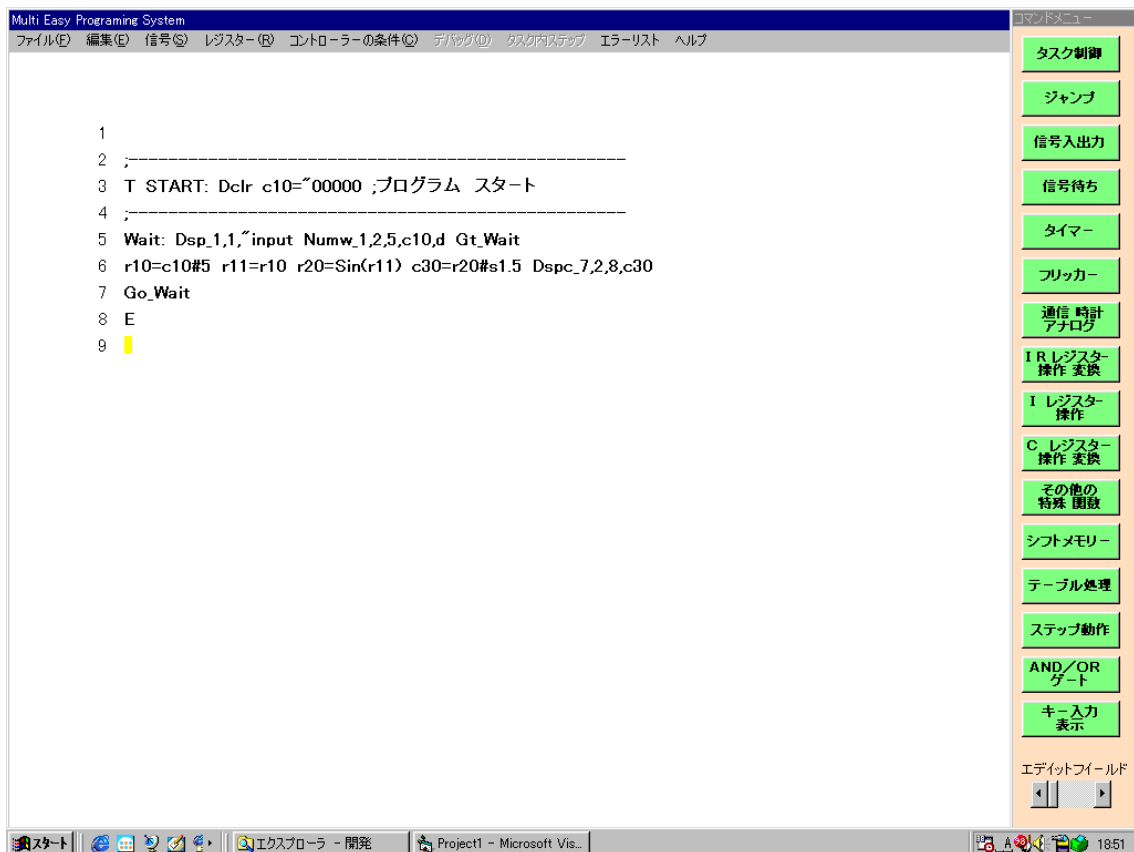
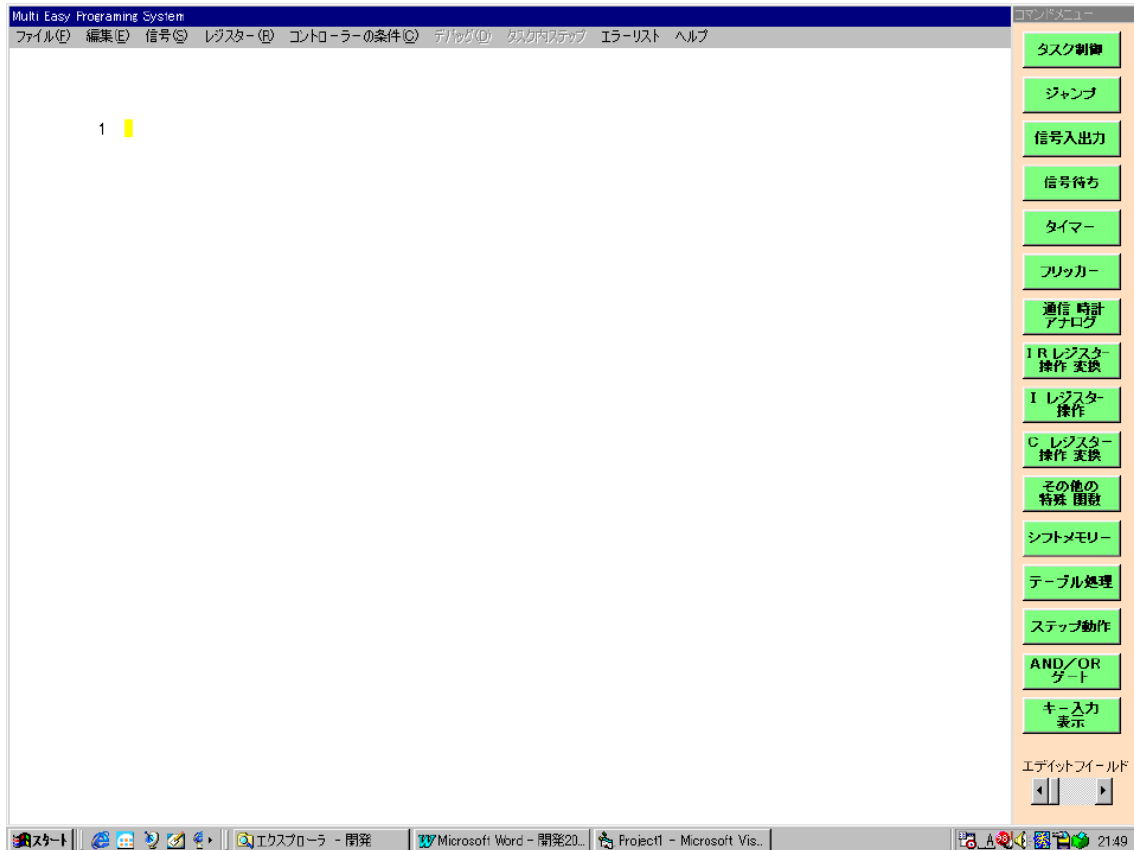
キー12入力、LCD6×16桁、I/O各32点フォトカプラー付き、カレンダー時計、RS232C 1CH、CPU Z80 10MHz、フラッシュメモリー512KB、スタティックRAM 512KB バッテリーバックアップ付き

オプションとしてA/D8CH、D/A2CH、RS422又はRS232C 1CH

もちろん簡易ロボット製作等の場合、CPU基板（PIO24ビット、RS232C 1CH）だけが最小システムで、切り離してロボットの中へ埋め込む事もできます。



## 2.1 コマンドメニューとエディット画面



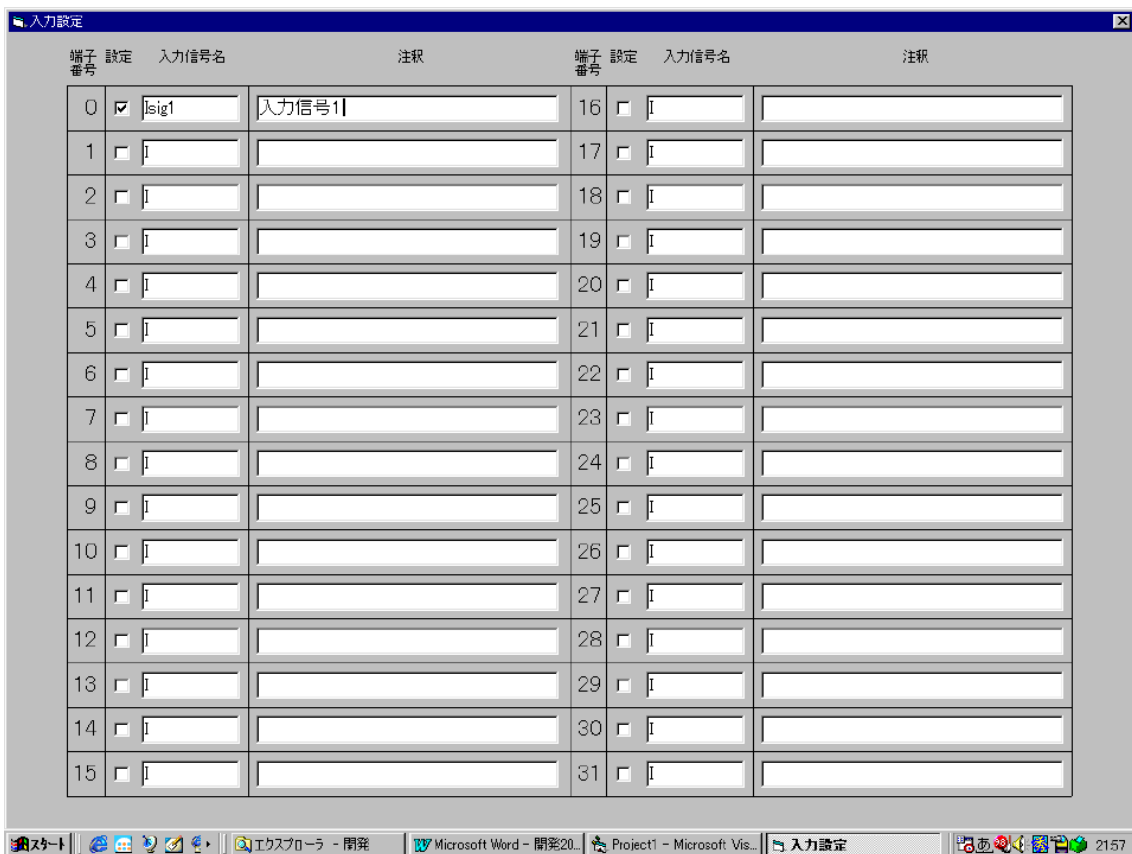
## Multi Easy Programming System

コマンド

| ファイル 編集 信号 レジスタ- コントローラの条件 デバッグ ステップ エラーリスト ヘルプ         | コマンド              |
|---|-------------------|
| エディット<br>フィールド  | タスク制御             |
|   | ジャンプ              |
|   | 信号入出力             |
|   | 信号待ち              |
|   | タイマー              |
|   | フリッカー             |
|   | 通信 時計<br>アナログ     |
|   | I R レジスタ<br>-操作変換 |
|   | I レジスタ<br>操作      |
|   | C レジスタ<br>操作変換    |
| コマンド選択された場合の各種画面<br>(エディットフィールドのカーソル位置によっては<br>上部に展開する) | その他の<br>特殊関数      |
|   | シフト<br>メモリー       |
|   | テーブル<br>処理        |
|   | ステップ<br>動作        |
|   | AND/OR<br>ゲート     |
|   | キー入力<br>表示        |

【図2】 コマンドジェネレーターの、メイン画面

図2は、コマンドジェネレーターの基本画面である。エディットフィールドにおける有効キーは半角英数字特殊記号で、注釈は全角漢字入力可能です。コントロールキーとして、スペースは**命令区切り**、DEL、TABは**1命令の削除**、BSは**1命令内のバックスキップ**、PgUp、PgDnは**ページ機能**。その他は無効です。まず、図中、メニューの、信号項目をクリックして入力、出力、内部信号の登録を行います。これは端子番号に名前と注釈を入れる事で、システムの設計と確認を容易にするためです。**マウスの変化で信号名や、注釈を表示します**。同様に使用レジスタNoにも、出来るだけ注釈を入れて下さい。



| 番号 | 設定                                  | 入力信号名 | 注釈 | 番号 | 設定                       | 入力信号名 | 注釈 |
|----|-------------------------------------|-------|----|----|--------------------------|-------|----|
| 0  | <input checked="" type="checkbox"/> | 入力信号1 |    | 16 | <input type="checkbox"/> |       |    |
| 1  | <input type="checkbox"/>            |       |    | 17 | <input type="checkbox"/> |       |    |
| 2  | <input type="checkbox"/>            |       |    | 18 | <input type="checkbox"/> |       |    |
| 3  | <input type="checkbox"/>            |       |    | 19 | <input type="checkbox"/> |       |    |
| 4  | <input type="checkbox"/>            |       |    | 20 | <input type="checkbox"/> |       |    |
| 5  | <input type="checkbox"/>            |       |    | 21 | <input type="checkbox"/> |       |    |
| 6  | <input type="checkbox"/>            |       |    | 22 | <input type="checkbox"/> |       |    |
| 7  | <input type="checkbox"/>            |       |    | 23 | <input type="checkbox"/> |       |    |
| 8  | <input type="checkbox"/>            |       |    | 24 | <input type="checkbox"/> |       |    |
| 9  | <input type="checkbox"/>            |       |    | 25 | <input type="checkbox"/> |       |    |
| 10 | <input type="checkbox"/>            |       |    | 26 | <input type="checkbox"/> |       |    |
| 11 | <input type="checkbox"/>            |       |    | 27 | <input type="checkbox"/> |       |    |
| 12 | <input type="checkbox"/>            |       |    | 28 | <input type="checkbox"/> |       |    |
| 13 | <input type="checkbox"/>            |       |    | 29 | <input type="checkbox"/> |       |    |
| 14 | <input type="checkbox"/>            |       |    | 30 | <input type="checkbox"/> |       |    |
| 15 | <input type="checkbox"/>            |       |    | 31 | <input type="checkbox"/> |       |    |

【図3】 入力信号登録画面

図3は入力信号の画面です。、以下出力、内部信号は同様です。

| 出力設定     |                                     |        |    |          |                          |       |
|----------|-------------------------------------|--------|----|----------|--------------------------|-------|
| 端子<br>番号 | 設定                                  | 出力信号名  | 注釈 | 端子<br>番号 | 設定                       | 出力信号名 |
| 0        | <input checked="" type="checkbox"/> | OLamp1 |    | 16       | <input type="checkbox"/> | O     |
| 1        | <input type="checkbox"/>            | O      |    | 17       | <input type="checkbox"/> | O     |

| 内部信号設定 |                                     |        |    |    |                          |       |
|--------|-------------------------------------|--------|----|----|--------------------------|-------|
| 番号     | 設定                                  | 内部信号名  | 注釈 | 番号 | 設定                       | 内部信号名 |
| 0      | <input checked="" type="checkbox"/> | VFlag1 |    | 16 | <input type="checkbox"/> | V     |
| 1      | <input type="checkbox"/>            | V      |    | 17 | <input type="checkbox"/> | V     |

入出力、内部信号とも、設定項目のレ点を入れないと登録していません。入出力点数は32点/32点ですが、内部信号は256点設定可能です。ページのup/downで対応しています。(大型システムでは、I/Oそれぞれ256点まで可能、又、簡易ロボット用のPIO24点は、出荷時、特注仕様として、コントローラーの変更をします。)

信号名は大文字、小文字の区別はありません。最大8文字までです。数字が頭文字でもかまいませんが、特殊記号には制約があります。(“=” 又は “\_” アンダーラインは使用不可です。)

同一の信号名のときは、小さい方の端子番号となりますので、注意して下さい。

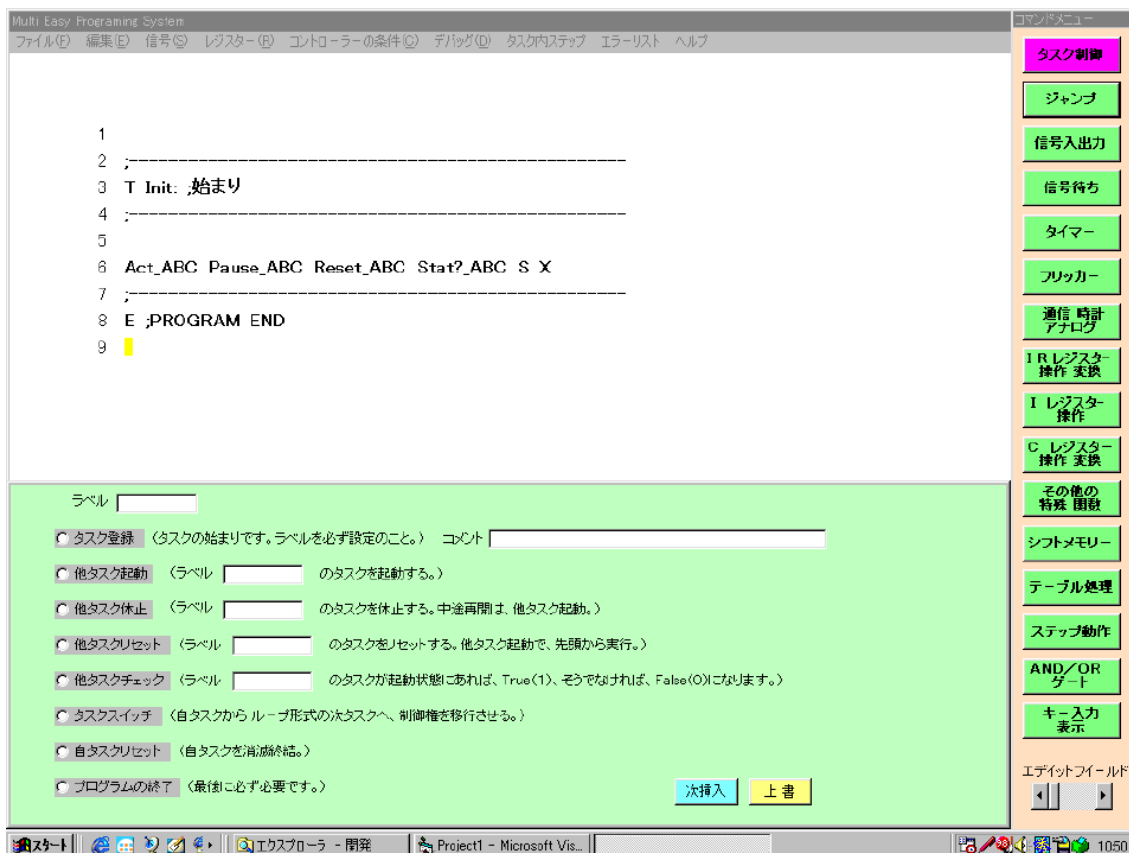
さて各コマンド選択画面を説明して行きますが、

選択画面に共通して、灰色の項目のスイッチを選択しますと (○はクリック選択スイッチ)、必要設定項目全て、フリッカーしますから、規定に基づき設定して、次挿入か、上書を押すことにより、命令を生成(ジェネレート)します。削除ボタンは、逆変換で、選択画面を出したときのみ存在し、受け付け可能です。又、逆変換の選択画面では、次挿入ボタンはありません。

レジスタ類NO、オーバータイマー値NO、データメモリーNOは、必ず10進2桁入力して下さい。定数、文字列入力は可変長です。

各コマンドの詳細は、MEPS I/O コンパイラーで説明しています。

## 2.2 タスク制御コマンド



ラベル  ①

- タスク登録** (タスクの始まりです。ラベルを必ず設定のこと。) コメント
- 他タスク起動** (ラベル  のタスクを起動する。)
- 他タスク休止** (ラベル  のタスクを休止する。中途再開は、他タスク起動。)
- 他タスクリセット** (ラベル  のタスクをリセットする。他タスク起動で、先頭から実行。)
- 他タスクチェック** (ラベル  のタスクが起動状態にあれば、True(1)、  
そうでなければ、False(0)になります。)
- タスクスイッチ** (自タスクからループ形式の次タスクへ、制御権を移行させる。)
- 自タスクリセット** (自タスクを消滅終結。)
- プログラムの終了** (最後に必ず必要です。)

②

|     |    |    |
|-----|----|----|
| 次挿入 | 上書 | 削除 |
|-----|----|----|

【図4】タスクコマンドの選択画面

- マルチタスク関連の命令は図4の如くまとめ、簡単にタスク間の連結を行います。但しデータのやりとりは行いません。本システムのデータ、レジスター、信号は全タスクから参照出来ます。
- とにかく、プログラムの開始は①タスク登録から始まります。ユーザープログラムで、MEPSコントローラーの電源ON時、最初の実行するのが、この最初に登録したタスクということで、初期タスクです。この初期タスクのみ、唯一、システムから起動されます。
- プログラムの終了には、②プログラムの終了スイッチを必ず選択して下さい。

## 2.3 ジャンプコマンド

Multi Easy Programing System  
 ファイル(F) 編集(E) 信号(S) レジスタ(R) コントローラーの条件(C) デバッグ(D) タスク内ステップ エラーリスト ヘルプ

コマンドメニュー

- タスク制御
- ジャンプ
- 信号入出力
- 信号待ち
- タイマー
- フリッカー
- 通信 時計 アナログ
- I/R レジスタ 操作 変換
- I レジスタ 操作
- C レジスタ 操作 変換
- その他の特殊 関数
- シフトメモリー
- テーブル処理
- ステップ動作
- AND/OR ゲート
- キー入力 表示
- エディットフィールド

```

1 Go_ABC Gf_ABC Gt_ABC If_i00>=i01,ABC If_i00<=i01,ABC If_i00>i01,ABC
2 If_i00<i01,ABC If_i00=i01,ABC If_i00>i01,ABC If_i00>=100,ABC If_i00<=100,ABC
3 If_i00>100,ABC If_i00<100,ABC If_i00=100,ABC If_i00<>100,ABC
4 If_r00>=r01,ABC If_r00<=r01,ABC If_r00>r01,ABC If_r00<r01,ABC If_r00=r01,ABC If_r00<>r01,ABC
5 Sel_c00,"A,Prog1,"B,Prog2,h'43,Prog3 Call_SUB1 R
  
```

ラベル

無条件に  ラベル  へジャンプします。  
 False(0)なら  
 True(0でない)なら  
 もしも (IF)

I  が  I  より  定数   
 R  が R  より

>= ならば  
 <= ならば  
 > ならば  
 < ならば  
 = ならば  
 <> ならば

ラベル  へジャンプします。それ以後は続行します。

ex. "A, Abc1, h'42, Abc2

選択 文字レジスター C  のコードによるジャンプ(コードとラベルをペアとして、カンマで区切ること。)   
 無条件に  ラベル  へコールします。  
 サブコールの終了。コールしたところの次のところへ戻る。

次挿入 上書

Project1 - Microsoft Vis... 11:09

ラベル

無条件に \_\_\_\_\_ ラベル  へジャンプします。  
 False(0)なら \_\_\_\_\_  
 True(0でない)なら \_\_\_\_\_

I  が  I  より  
 定数

R  が R  より

>=ならば  
 <=ならば  
 >ならば  
 <ならば  
 =ならば  
 <>ならば

\_\_\_\_\_ ラベル  へジャンプします。それ以外は続行します。

① 選択 文字レジスタ C  のコードによるジャンプ ex、"A",Abc1,h'42,Abc2  
 (コードとラベルをペアとして、カンマで区切ること。)

無条件に \_\_\_\_\_ ラベル  へコールします。  
 サブコールの終了。コールしたところの次のところへ戻る。

|     |    |    |
|-----|----|----|
| 次挿入 | 上書 | 削除 |
|-----|----|----|

【図5】ジャンプコマンドの選択画面

- ①選択ではCの内容で、ジャンプ先が分岐する形ですが、マッチングコード以外は次命令の続行となります。



## 2.4 信号入出力コマンド

Multi Easy Programing System

ファイル(F) 編集(E) 信号(S) レジスタ(R) コントローラの条件(C) デバッグ(D) タスク内ステップ エラーリスト ヘルプ

コマンドメニュー

- タスク制御
- ジャンプ
- 信号入出力
- 信号待ち
- タイマー
- フリッカー
- 通信 時計 アナログ
- IR レジスタ 操作 変換
- I レジスタ 操作
- C レジスタ 操作 変換
- その他の 特殊 関数
- シフトメモリー
- テーブル処理
- ステップ動作
- AND/OR ゲート
- キー入力 表示

```

1 I?_Isig1 Gf_ABC I?_Isig1 Gt_ABC V?_Vsig1 Gf_ABC V?_Vsig1 Gt_ABC
2 O?_Osig1 Gf_ABC O?_Osig1 Gt_ABC On_Osig1 Of_Osig1 Vn_Vsig1 Vf_Vsig1
3 Pi_0,i00&h'FF Po_0,i00&h'FF
  
```

ラベル

入力信号  を入力して

OFF なら

ON なら

内部信号  をチェックして

OFF なら

ON なら

出力信号  をチェックして

OFF なら

ON なら

出力信号  を

ON する

OFF する

内部信号  を

ON する

OFF する

ラベル  ヘジャンプし、それ以外は続行します

ポート入力  ポート入力に、16進2桁  と、ANDをとって、I  へ、ストアします。

ポート出力  ポートへ、I  の内容に、16進2桁  と、ANDをとって、出力します。ANDデータの、0、のビット相当端子の変化は無し

次挿入 上書

エディットフィールド

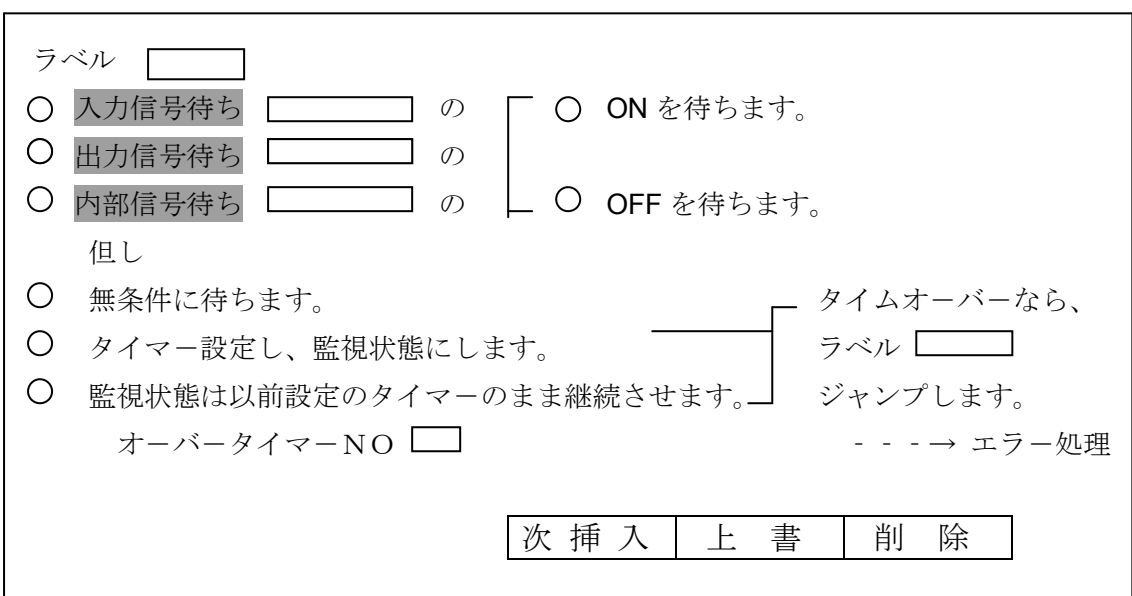
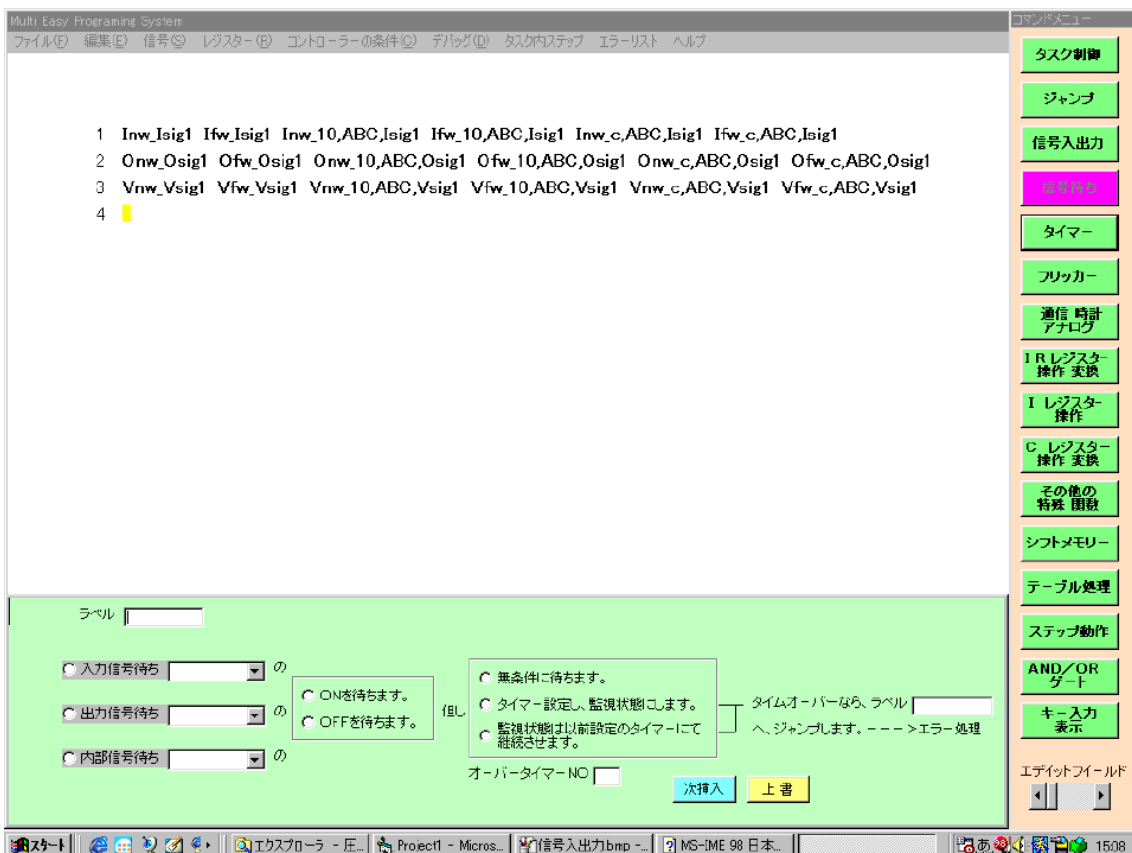
Windows 10 taskbar: エクスプローラ - 開発, Project1 - Microsoft Vis..., 1428

|  |                          |  |   |   |     |    |    |
|--|--------------------------|--|---|---|-----|----|----|
| ラベル  |                          |  |   |   |     |    |    |
| <input type="radio"/> 入力信号   | <input type="checkbox"/> | を入力して  | —   | <input type="radio"/> OFF なら                        |     |    |    |
|  |                          |  |   | <input type="radio"/> ON なら                         |     |    |    |
| <input type="radio"/> 内部信号   | <input type="checkbox"/> | をチェックして  | —   | <input type="radio"/> OFF なら                        |     |    |    |
|  |                          |  |   | <input type="radio"/> ON なら                         |     |    |    |
| <input type="radio"/> 出力信号   | <input type="checkbox"/> | をチェックして  | —   | <input type="radio"/> OFF なら                        |     |    |    |
|  |                          |  |   | <input type="radio"/> ON なら                         |     |    |    |
|  |                          |  |   | ラベル <input type="checkbox"/> ヘジヤンプし、<br>それ以外は続行します。 |     |    |    |
| <input type="radio"/> 出力信号   | <input type="checkbox"/> | を  | <input type="radio"/> ON する<br><input type="radio"/> OFF する |   |     |    |    |
| <input type="radio"/> 内部信号   | <input type="checkbox"/> | を  | <input type="radio"/> ON する<br><input type="radio"/> OFF する |   |     |    |    |
| <input type="radio"/> ポート入力  | <input type="checkbox"/> | ポート入力に、16進2桁 <input type="checkbox"/> と AND をとって、I <input type="checkbox"/> へ、ストアします。                            |   |   |     |    |    |
| <input type="radio"/> ポート出力  | <input type="checkbox"/> | ポートへ、I <input type="checkbox"/> の内容に、16進2桁 <input type="checkbox"/> と、AND をとって、出力します。<br>(AND をとった BIT 以外の変化はなし) |   |   |     |    |    |
| <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>次挿入</td> <td>上書</td> <td>削除</td> </tr> </table> |                          |  |   |   | 次挿入 | 上書 | 削除 |
| 次挿入  | 上書                       | 削除   |   |   |     |    |    |

【図6】 信号入出力コマンドの選択画面

- 信号は登録済みならカッコをクリックすると一覧が出ますから、選択クリックして下さい。
- ポートのNOは0から3までです。1ポートが8点で、 $8 \times 4 = 32$ 点 **信号と重複に注意**して下さい。

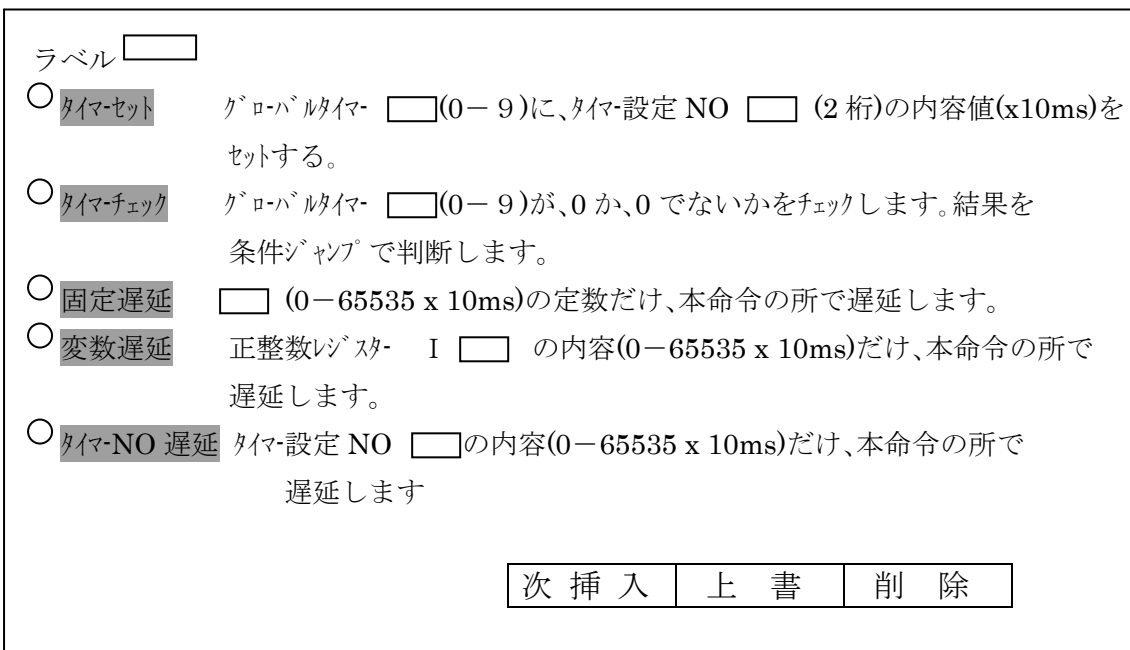
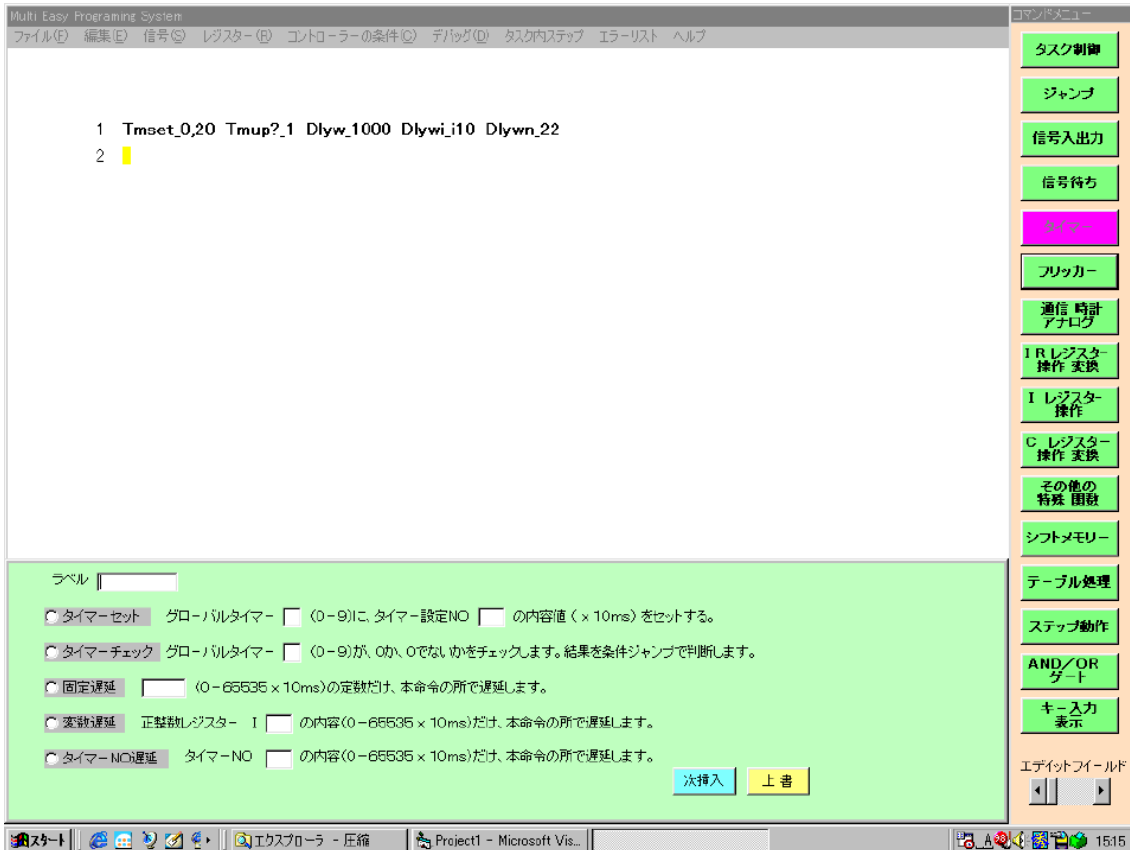
## 2.5 信号待ちコマンド



【図7】信号待ちコマンドの選択画面

- 監視状態は以前設定のタイマーのままの信号待ち、をいきなり選択して動作させるとタイマー値を設定していないのだから、間違いです。注意して下さい。

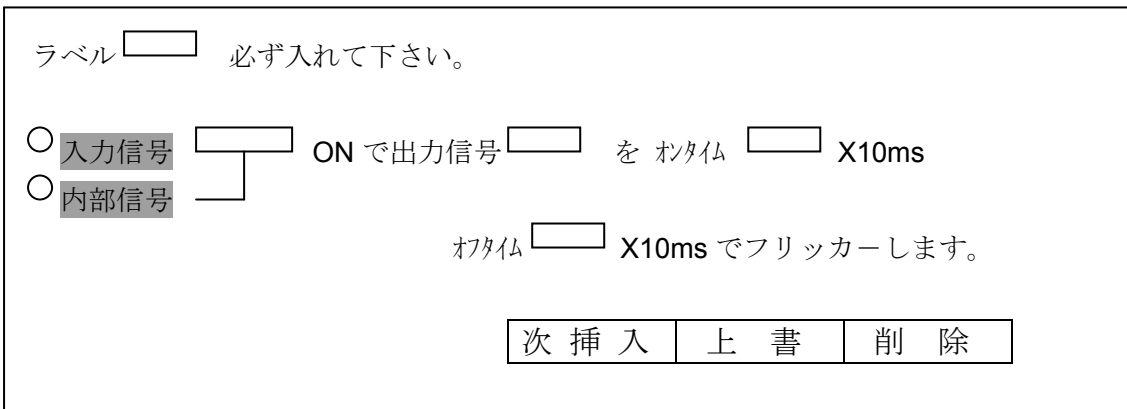
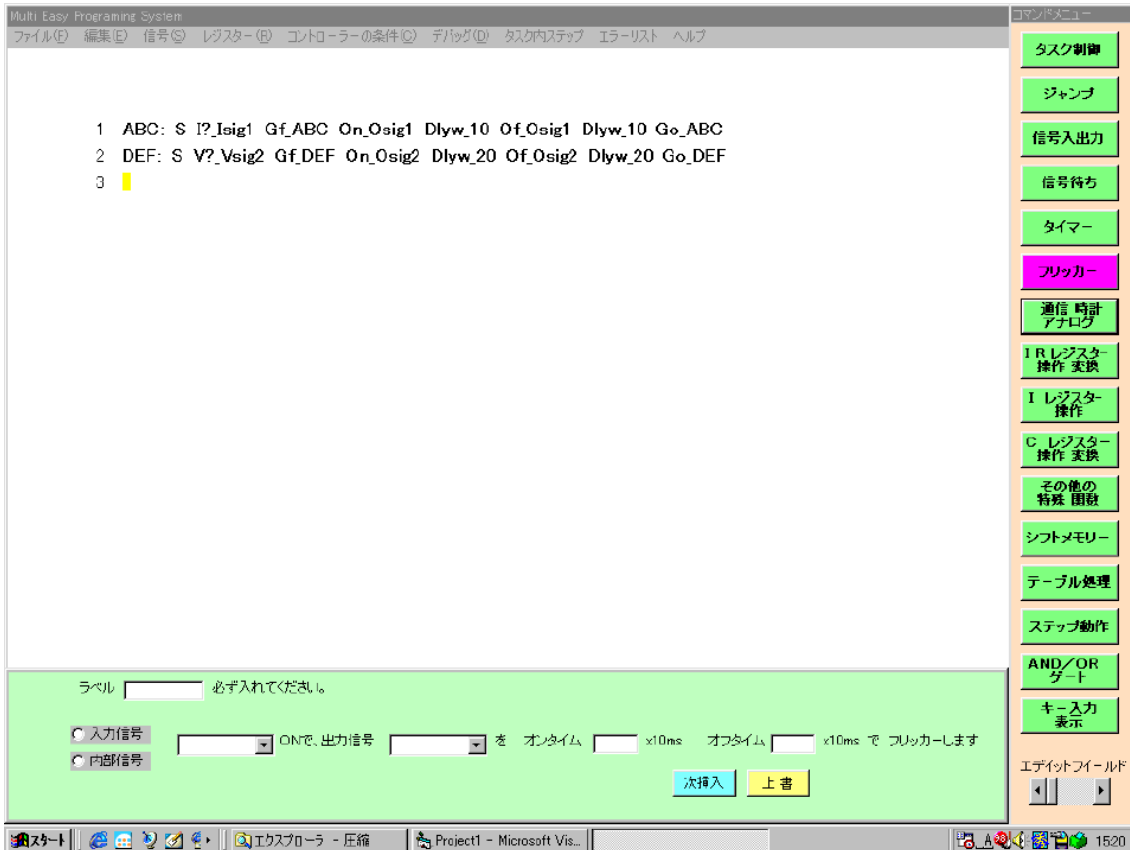
## 2.6 タイマーコマンド



【図8】タイマーコマンドの選択画面

- グローバルタイマーはタスク間の総合タイマーです。普通はタスク固有1個のタイマーで十分です。

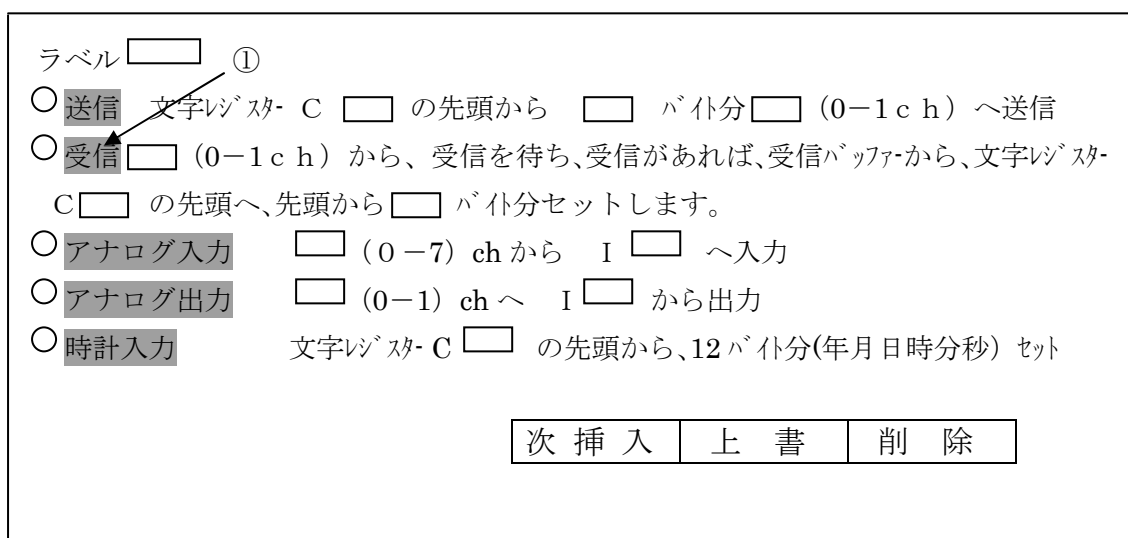
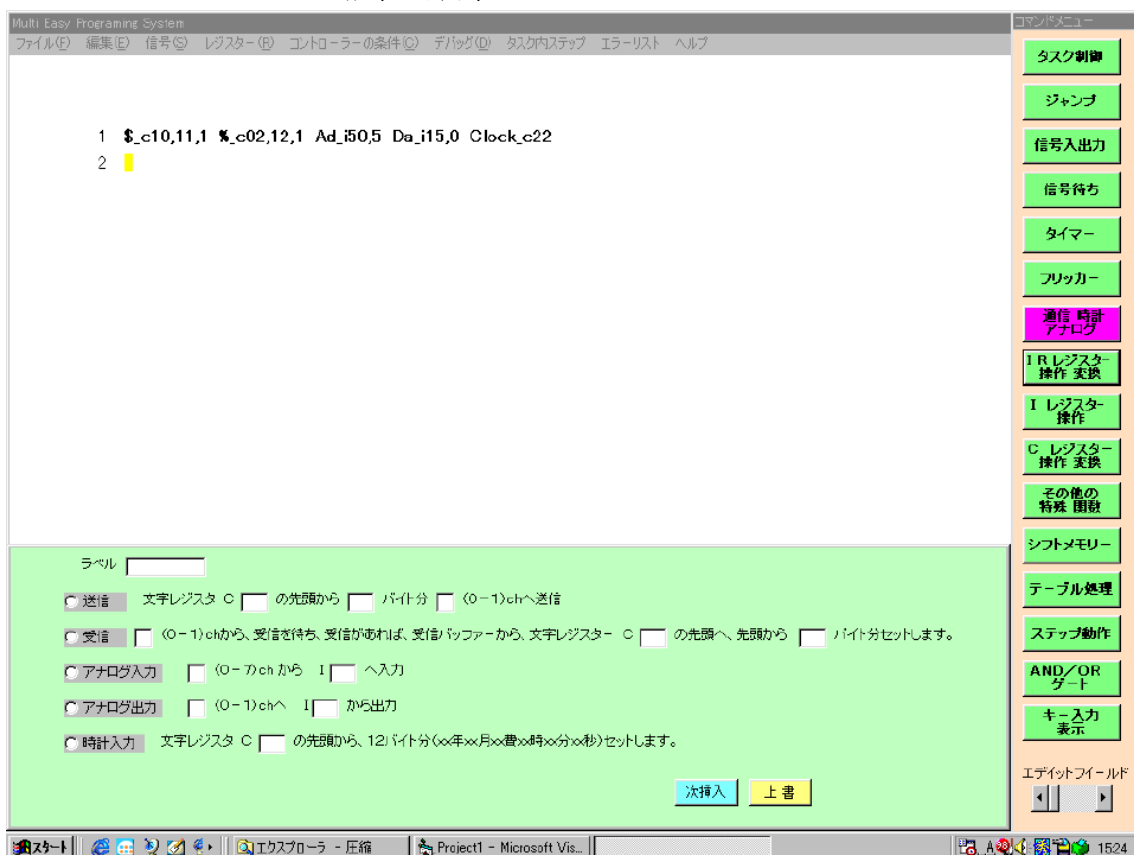
## 2.7 フリッカーコマンド



【図9】フリッカーコマンドの選択画面

- このプログラムのラベルをタスクのヘッドにすれば (T をつければ)、1つのタスクとなります。
- 同じフリッカータイムどうしのタスクが、ばらばらに動作する場合、乱雑な感じなので、内部信号を使用し、ONするときは、他の同じフリッカータイムのタスクをリセットして、起動すれば良いです。そうするとON/OFFが同期します。

## 2.8 通信、時計、アナログコマンド



【図10】通信、時計、アナログコマンドの選択画面

- ①受信命令で、受信がなければ、永遠に待ちつづける事をさけるため、別タスクで、タイマー監視を行ってください。(受信命令の直前に内部信号の一つを ON し、直後に OFF すれば、別タスクで、この内部信号の ON/OFF 待ちをタイマー監視命令でチェック出来ます。5、信号待ちコマンド参照の事。)

## 2.9 I、R、レジスタ操作、変換コマンド

Multi Easy Programming System  
 ファイル(F) 編集(E) 信号(S) レジスタ(R) コントローラの条件(C) デバッグ(D) タスク内ステップ エラーリスト ヘルプ

```

1 i00=i05+i10 i00=i05-i10 i00=i05*i10 i00=i05/i10
2 i11=i44+55 i11=i44-55 i11=i44*55 i11=i44/55 i22=i77 i60=1000
3 r15=r15+r30 r15=r15-r30 r15=r15*r30 r15=r15/r30 r33=r90
4 r10=i10 i30=r50
  
```

ラベル

I 正整数レジスタ同志の演算式  $I \square = I \square$   I 正整数レジスタと定数の演算式  $I \square = I \square$

I 正整数レジスタ同志の移動  $I \square = I \square$

R 実数レジスタ同志の演算式  $R \square = R \square$   I 正整数レジスタへの定数の代入  $I \square = \square$

R 実数レジスタ同志の移動  $R \square = R \square$

I 正整数レジスタから R 実数レジスタへの変換  $R \square = I \square$  (I は 0-9999 の範囲)

R 実数レジスタから I 正整数レジスタへの変換  $I \square = R \square$  (I は 0-9999 の範囲)

次挿入 上書

コマンドメニュー

- タスク制御
- ジャンプ
- 信号入出力
- 信号待ち
- タイマー
- フロッカー
- 通信 時計 アナログ
- I レジスタ同志の移動
- I レジスタ操作
- C レジスタ操作 変換
- その他の特殊 閉鎖
- シフトメモリー
- テーブル処理
- ステップ動作
- AND/OR ゲート
- キー入力表示

エディットフィールド

タスクバー: エクスプローラ - 圧縮 Project1 - Microsoft Vis... 15:36

ラベル

I 正整数レジスタ-同志の演算式  $I \text{  } = I \text{  } \begin{matrix} \text{○} + \\ \text{○} - \\ \text{○} * \\ \text{○} / \end{matrix} I \text{  }$

I 正整数レジスタ-と定数の演算式  $I \text{  } = I \text{  } \begin{matrix} \text{○} + \\ \text{○} - \\ \text{○} * \\ \text{○} / \end{matrix} \text{  } \text{①}$

I 正整数レジスタ-同志の移動  $I \text{  } = I \text{  }$

R 実数レジスタ-同志の演算式  $R \text{  } = R \text{  } \begin{matrix} \text{○} + \\ \text{○} - \\ \text{○} * \\ \text{○} / \end{matrix} R \text{  }$

I 正整数レジスタ-への定数の算入  $I \text{  } = \text{  }$

R 実数レジスタ-同志の移動  $R \text{  } = R \text{  }$

I 正整数レジスタ-から R 実数レジスタ-への変換  $R \text{  } = I \text{  } \text{②}$

R 実数レジスタ-から I 正整数レジスタ-への変換  $I \text{  } = R \text{  }$

【図11】 I、R、レジスタ操作、変換コマンドの選択画面

- I 正整数レジスタ-への定数①の算入は0から65535までです。
- ② I 正整数レジスタ-とR 実数レジスタ-間のデータ範囲は、0から9999までです。(C文字レジスタ-とR 実数レジスタ-間の変換、逆変換のほうが、直接、浮動小数点形式で出来るため、便利です。11、Cレジスタ-操作、変換コマンド参照の事。)



## 2.10 I、レジスタ操作コマンド

Multi Easy Programing System  
 ファイル(F) 編集(E) 信号(S) レジスタ(R) コントローラの条件(C) デバッグ(D) タスク内ステップ エラーリスト ヘルプ

```

1 i30=i40&i57 i30=i40@i57 i23=i36&h9999 i23=i36@h9999
2 i10=RL(i11,15) i10=RR(i11,15) i10=SL(i11,15) i10=SR(i11,15)
3 i50=D(i20) i50=B(i12) Mv_Data(i21),i29 Mv_i29,Data(i21)
4

```

ラベル

I 正整数レジスタ-同志の論理式  $I_{\square} = I_{\square}$ 
 AND  $I_{\square}$ 
 OR  $I_{\square}$ 
 I 正整数レジスタ-と定数の論理式  $I_{\square} = I_{\square}$ 
 AND  $I_{\square}$ 
 OR  $I_{\square}$

I 正整数レジスタの左回転  $I_{\square} \leftarrow I_{\square}$  (1-15)回 1ビットずつ左回転して、 $I_{\square}$  にセットする。

I 正整数レジスタの右回転  $I_{\square} \rightarrow I_{\square}$  (1-15)回 1ビットずつ右回転して、 $I_{\square}$  にセットする。

I 正整数レジスタの左シフト  $\leftarrow I_{\square}$  (1-15)回 1ビットずつ左シフトして、 $I_{\square}$  にセットする。

I 正整数レジスタの右シフト  $I_{\square} \rightarrow$  (1-15)回 1ビットずつ右シフトして、 $I_{\square}$  にセットする。

I 正整数レジスタの10進変換 2進数  $I_{\square}$  を10進数  $I_{\square}$  に変換する。但し0から9999の範囲。

I 正整数レジスタの2進変換 10進数  $I_{\square}$  を2進数  $I_{\square}$  に変換する。但し0から9999の範囲。

I 正整数レジスタをインデックスとした、データメモリーへの書き込み  $I_{\square}$  をインデックスとして、 $I_{\square}$  の内容を、データメモリーに書き込む。

I 正整数レジスタをインデックスとした、データメモリーからの読み込み  $I_{\square}$  をインデックスとして、 $I_{\square}$  へ、データメモリーから読み込む。

次挿入 上書

コマンドメニュー

- タスク制御
- ジャンプ
- 信号入出力
- 信号待ち
- タイマー
- フリッカー
- 通信 時計 アナログ
- IRレジスタ操作 変換
- Iレジスタ操作
- Cレジスタ操作 変換
- その他の特殊 関数
- シフトメモリー
- テーブル処理
- ステップ動作
- AND/ORゲート
- キー入力表示

エディットフィールド

Windows 1544

ラベル

○ I 正整数レジスタ-同志の論理式  $I \text{  } = I \text{  } \text{ --- } \begin{matrix} \text{AND} \\ \text{OR} \end{matrix} \text{ --- } I \text{  }$

○ I 正整数レジスタ-と定数の論理式  $I \text{  } = I \text{  } \text{ --- } \begin{matrix} \text{AND} \\ \text{OR} \end{matrix} \text{ --- } \text{}$  ①

○ 正整数レジスタ-の左回転  $I \text{  } \leftarrow \text{}$  (1-15)回 1ビットずつ左回転して、

○ I 正整数レジスタ-の右回転  $\text{} \rightarrow I \text{ }$  (1-15)回 1ビットずつ右回転して

○ I 正整数レジスタ-の左シフト  $\leftarrow I \text{ } \text{ --- } \text{}$  (1-15)回 1ビットずつ左シフトして

○ I 正整数レジスタ-の右シフト  $\text{--- } I \text{ } \rightarrow \text{}$  (1-15)回 1ビットずつ右シフトして

I  にセットする。

○ I 正整数レジスタ-の 10 進変換 2 進数 I  を 10 進数  $I \text{ }$  に変換する。

○ I 正整数レジスタ-の 2 進変換 10 進数 I  を 2 進数  $I \text{ }$  へ変換する。但し 0 から 9999 の範囲。

○ I 正整数レジスタ-をインデックスとした、データメモリーへの書き込み  
I  をインデックスとして、I  の内容を、データメモリーに書き込む。

○ I 正整数レジスタ-をインデックスとした、データメモリーからの読み込み  
I  をインデックスとして、I  へ、データメモリーから読み込む。

②

次 挿 入 | 上 書 | 削 除

【図 1 2】 I、レジスタ操作コマンドの選択画面

- I 正整数レジスタ-と定数の論理式における定数①は 16 進で入力して下さい。
- 2 進数と 16 進数の違いは、表記の違いだけで、同じ意味です。
- データメモリーとのやりとりに使用する I インデックス②は 0 から 9999 までです。

## 2.11 C、レジスター操作、変換コマンド

Multi Easy Programming System  
 ファイル(F) 編集(E) 信号(S) レジスタ(R) コントローラの案件(C) デバッグ(D) タスク内ステップ エラーリスト ヘルプ

```

1 c22="ABCDEF123 c22="313664 c22="h'23557980 c12="h'12345678ab
2 c44=i12#h c44=i12#z3 c12=i44#s3
3 c10=r11#z2.3 c10=r11#s2.3
4 i24=c11#h i24=c11#2 r89=c23#6
  
```

ラベル

文字レジスタへ直接入力  の先頭から  アスキーコード  10進コード  を文字列とする。

Iレジスタから文字レジスタへ入力 I  の内容、2進数
  16進で4桁  ゼロサプレス無しの10進で  桁として、C  の先頭からの文字列とする。
  ゼロサプレス有りの10進で

Rレジスタから文字レジスタへ入力 R  の内容
  ゼロサプレス無しの10進で 正数  桁、小数  桁を C  の先頭からの文字列とする。
  ゼロサプレス有りの10進で

文字レジスタからIレジスタへ入力 C  を先頭とした文字列
  16進で4桁  ゼロサプレス有り無し10進で  桁  を I  へ2進化入力する。

文字レジスタからRレジスタへ入力 C  を先頭とした文字列  桁を R  へ浮動小数点化入力する。

次挿入 上書

コマンドメニュー

- タスク制御
- ジャンプ
- 信号入出力
- 信号待ち
- タイマー
- フリッカー
- 通信 時計 アナログ
- IRレジスタ操作 変換
- Iレジスタ操作
- レジスタ間の変換
- その他の特殊 関数
- シフトメモリー
- テーブル処理
- ステップ動作
- AND/ORゲート
- キー入力表示

エディットフィールド

Windows taskbar: エクスプローラ - 圧縮 Project1 - Microsoft Vis... 1617

ラベル

**文字レジスタ-直接入力** C  の先頭から  アスキーコード  <sup>①</sup> を文字列とする。  
 16進コード

**Iレジスタ-から文字レジスタ-入力** I  の内容、2進数  16進で4桁  
 ゼロパディング無しでの10進で  桁  
 ゼロパディング有りの10進で  桁  
 として、C  の先頭からの文字列とする。

**Rレジスタ-から文字レジスタ-入力** R  の内容  ゼロパディング無しでの10進で  
 ゼロパディング有りの10進で  
 正数  桁、小数  桁を C  の先頭からの文字列とする。

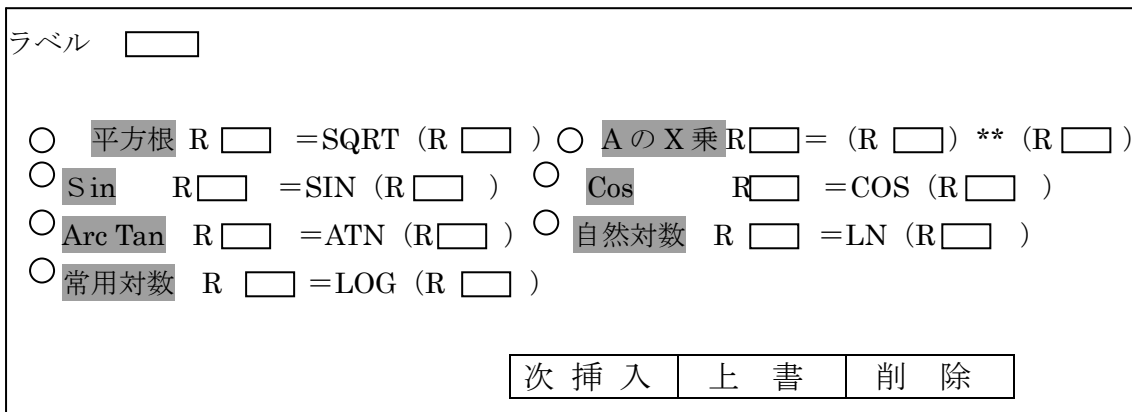
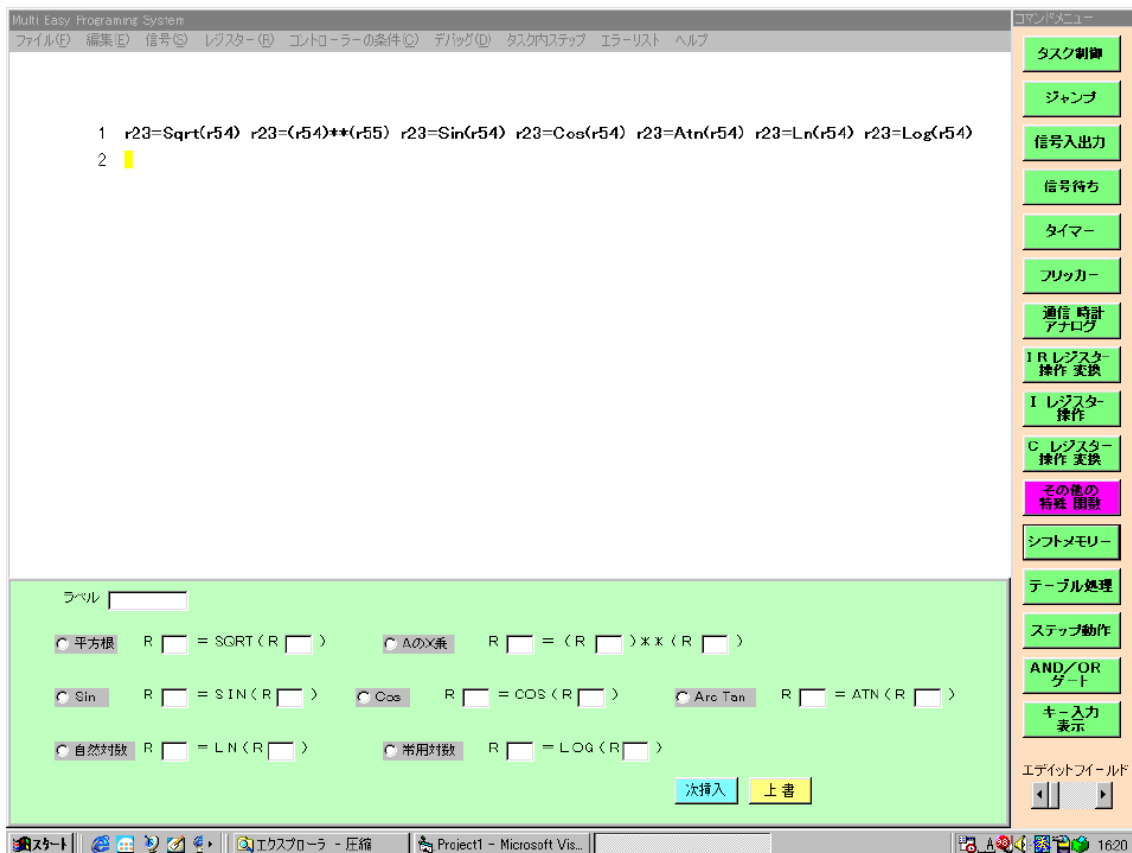
**文字レジスタ-からIレジスタ-入力** C  を先頭とした文字列 \_\_\_\_\_  
 \_\_\_\_\_  16進で4桁  
 ゼロパディング有り無しでの10進で  桁 を I  へ2進化入力する。

**文字レジスタ-からRレジスタ-入力** C  を先頭とした文字列  桁を R  へ  
 浮動小数点化入力する。

【図13】C、レジスタ操作、変換コマンドの選択画面

- 文字レジスタ-直接入力①での文字及び16進に、“や、h”は不要です。

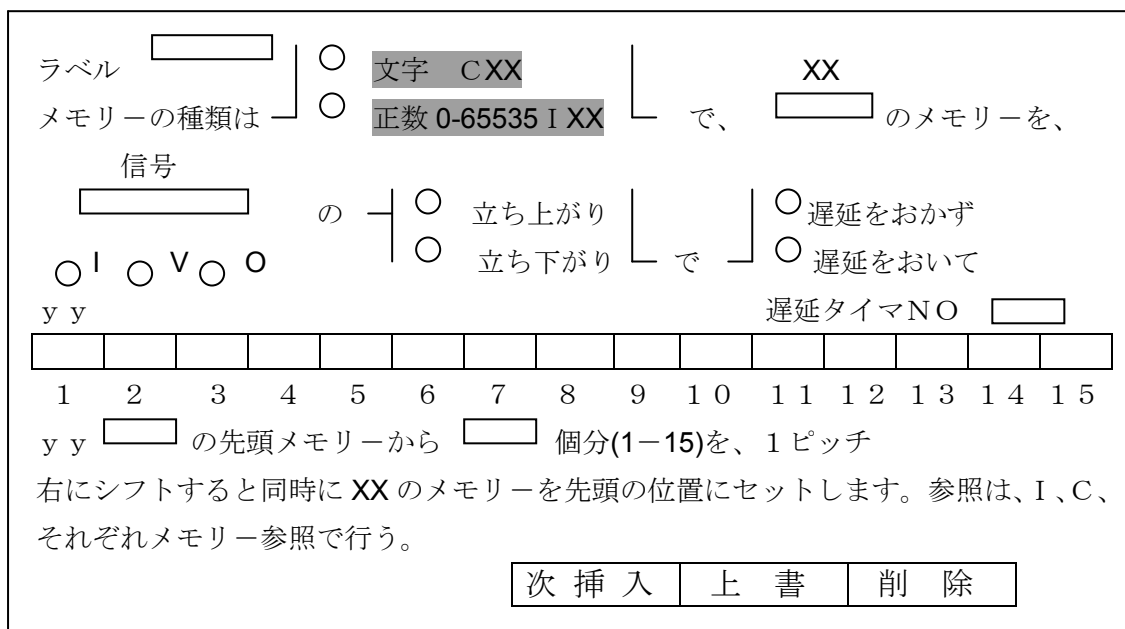
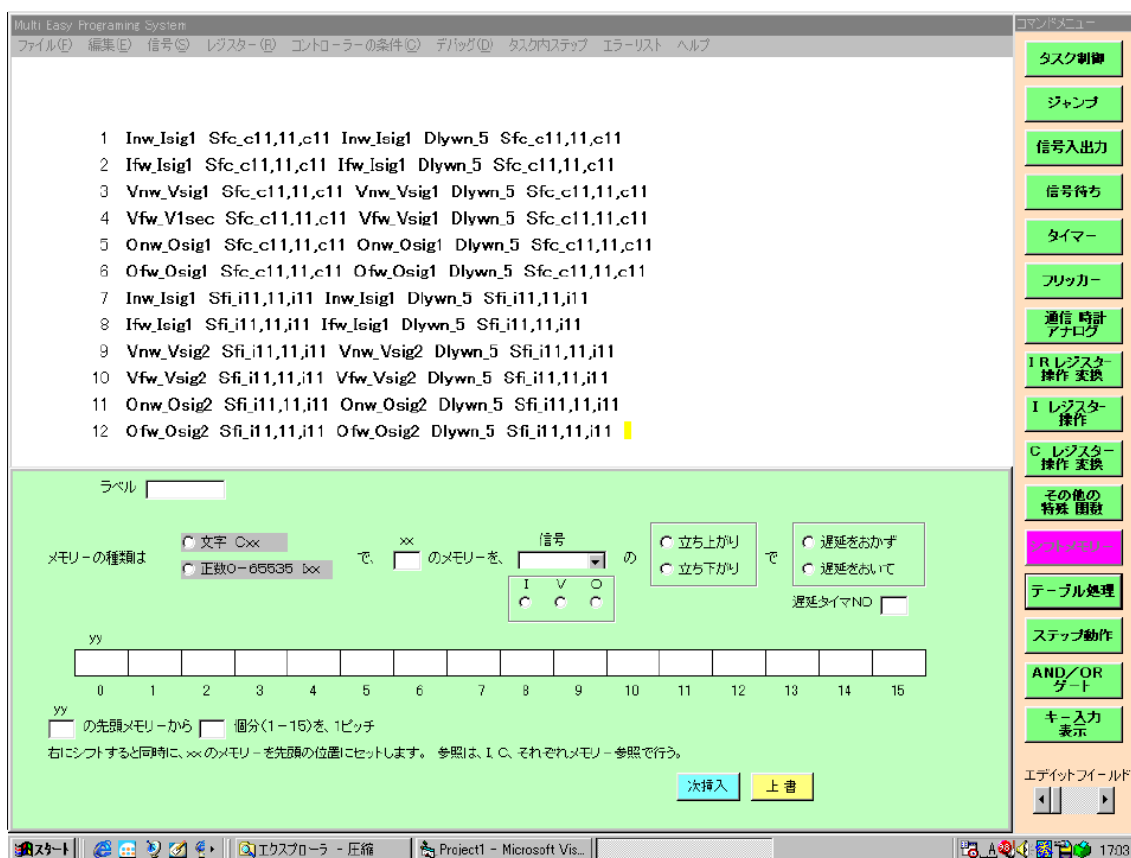
## 2.12 その他の特殊、関数コマンド



【図 1 4】 その他の特殊、関数コマンドの選択画面

- 三角関数はラジアンで処理します。

## 2.13 シフトメモリーコマンド



【図 15】シフトメモリーコマンドの選択画面

- 直列、又は回転パケットに情報をのせて、遅延処理させるものに対応した命令です。

## 2.14 テーブル処理コマンド

Multi Easy Programming System

ファイル(F) 編集(E) 信号(S) レジスタ(R) コントローラーの案件(C) デバッグ(D) タスク内ステップ エラーリスト ヘルプ

```

1 ABC: ;
2 ;; Db_1,2,3
3 ;; Db_4,5,6
4 AV: ;
5 ;; Dw_1,2,3,4
6 ;; Dw_5,6,7,8
7 Getb_ABC,c20,3,i10 Getw_AV,i0,4,i10
    
```

ラベル

テーブル作成  バイトテーブル  ワードテーブル

縦  × 横  のテーブルで、入力方法は、16進で入れて下さい。(最大 50 × 10)  
 テーブルのラベルは、必ず入れて下さい。

Iレジスタのポインタ

|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 |   |   |   |   |   |   |   |   |   |   |
| 1 |   |   |   |   |   |   |   |   |   |   |
| 2 |   |   |   |   |   |   |   |   |   |   |
| 3 |   |   |   |   |   |   |   |   |   |   |
| 4 |   |   |   |   |   |   |   |   |   |   |
| 5 |   |   |   |   |   |   |   |   |   |   |
| 6 |   |   |   |   |   |   |   |   |   |   |
| 7 |   |   |   |   |   |   |   |   |   |   |
| 8 |   |   |   |   |   |   |   |   |   |   |
| 9 |   |   |   |   |   |   |   |   |   |   |

ページ送り  
ページ戻し

バイトテーブル読み込み  ワードテーブル読み込み

I  で、ラベル  のテーブルから、横  の先頭からセットする。

テーブルの縦ポインタ      テーブルの横列と同じにする

バイト分、C   
 ワード分、I

上書   削除

コマンドメニュー

- タスク制御
- ジャンプ
- 信号入出力
- 信号待ち
- タイマー
- フリッカー
- 通信 時計 アナログ
- I R レジスタ 操作 変換
- I レジスタ 操作
- C レジスタ 操作 変換
- その他の 特殊 閉鎖
- シフトメモリー
- テーブル処理
- ステップ動作
- AND/OR ゲート
- キー入力 表示

エディットフィールド

エクスプローラ - 開発    Project1 - Microsoft Vis...    1740

ラベル

テーブル作成

バイトテーブル

ワードテーブル

縦  × 横  のテーブルで  
入力方法は、16進で入れて下さい。  
テーブルのラベルは、必ず入れて下さい。

Iレジスタ-のポインタ-

|   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|
|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 |   |   |   |   |   |   |   |   |   |   |
| 1 |   |   |   |   |   |   |   |   |   |   |
| 2 |   |   |   |   |   |   |   |   |   |   |
| 3 |   |   |   |   |   |   |   |   |   |   |
| 4 |   |   |   |   |   |   |   |   |   |   |
| 5 |   |   |   |   |   |   |   |   |   |   |
| 6 |   |   |   |   |   |   |   |   |   |   |
| 7 |   |   |   |   |   |   |   |   |   |   |
| 8 |   |   |   |   |   |   |   |   |   |   |
| 9 |   |   |   |   |   |   |   |   |   |   |

①

ページ送り

ページ戻り

バイトテーブル読み込み

ワードテーブル読み込み

I  で、ラベル  のテーブルから、横

|

テーブルの縦ポインタ

|

テーブルの横列と同じにする

バイト分、C

ワード分、I

の先頭からセットする。

|     |    |    |
|-----|----|----|
| 次挿入 | 上書 | 削除 |
|-----|----|----|

【図16】テーブル処理コマンドの選択画面

- テーブル最大は縦50×横10です。入力は16進です。縦10以上は①ページの送り、戻りで設定して下さい。
- 信号入力を縦ポインタとして、テーブルをルックアップして数値情報をひきだしたり、別信号に変換したりします。

32



## 2.15 ステップ動作コマンド

Multi Easy Programing System

ファイル(F) 編集(E) 信号(S) レジスタ(R) コントローラの条件(C) デバッグ(D) タスク内ステップ エラーリスト ヘルプ

コマンドメニュー

- タスク制御
- ジャンプ
- 信号入出力
- 信号待ち
- タイマー
- フリッカー
- 通信 時計 アナログ
- IR レジスタ 操作 変換
- I レジスタ 操作
- C レジスタ 操作 変換
- その他の特殊 閉鎖
- シフトメモリー
- テーブル処理
- ステップ動作
- AND/OR ゲート
- キー入力 表示

```

1 Inw_Isig1 On_Osig1 Inw_Isig1 Of_Osig1
2 Inw_Isig1 On_Osig1 Dlyw_12 Of_Osig1 Inw_Isig1 Of_osig1 Dlyw_2 On_osig1
3 Inw_Isig2 Dlyw_5 On_Osig2 Inw_Isig2 Dlyw_5 Of_Osig2
4 Inw_Isig2 Dlyw_5 On_Osig2 Dlyw_12 Of_Osig2 Inw_Isig2 Dlyw_5 Of_Osig2 Dlyw_12 On_Osig2
5 Ifw_Isig1 Vn_Vsig1 Ifw_Isig1 Vf_Vsig1
6 Ifw_Isig1 Vn_Vsig1 Dlyw_12 Vf_Vsig1 Vfw_Vsig1 Of_Osig1 Dlyw_12 On_Osig1
7 Vfw_Vsig2 Dlyw_5 Vn_Vsig2 Ifw_Isig2 Dlyw_5 Vf_Vsig2 Dlyw_12 Vn_Vsig2
8 Inw_12,as,Isig1 On_Osig1 Inw_12,as,Isig1 Vn_Vsig1 Dlyw_12 Vf_Vsig1
9 Vfw_12,as,Vsig1 Dlyw_5 Of_Osig1 Dlyw_12 On_Osig1
10 Inw_c,as,Isig1 Of_Osig1 Vnw_c,as,Vsig1! Dlyw_ Ifw_c,as,Isig1 Vf_Vsig1 Dlyw_12 Vn_Vsig1
11 Vfw_c,as,Vsig1 Dlyw_5 Of_Osig1 Dlyw_12 On_Osig1
    
```

ラベル

オーバertimeチェック無し  
 オーバertimeチェック有り  
 オーバertimeチェック継続

オーバータイマーNO

信号  の  I  V

立ち上がり  
 立ち下がり

で  遅延をおかず  遅延をおいて

信号  を  O  V

遅延  x10ms

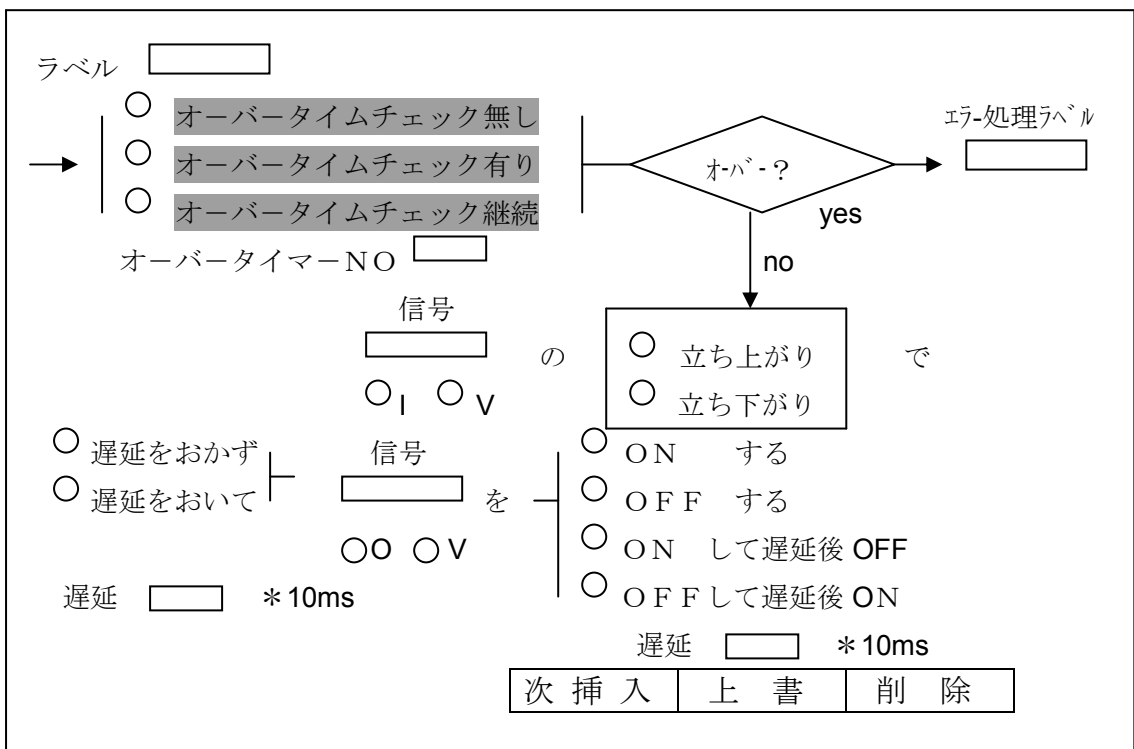
ON する  
 OFF する  
 ON して遅延後 OFF  
 OFF して遅延後 ON

遅延  x10ms

次挿入 上書

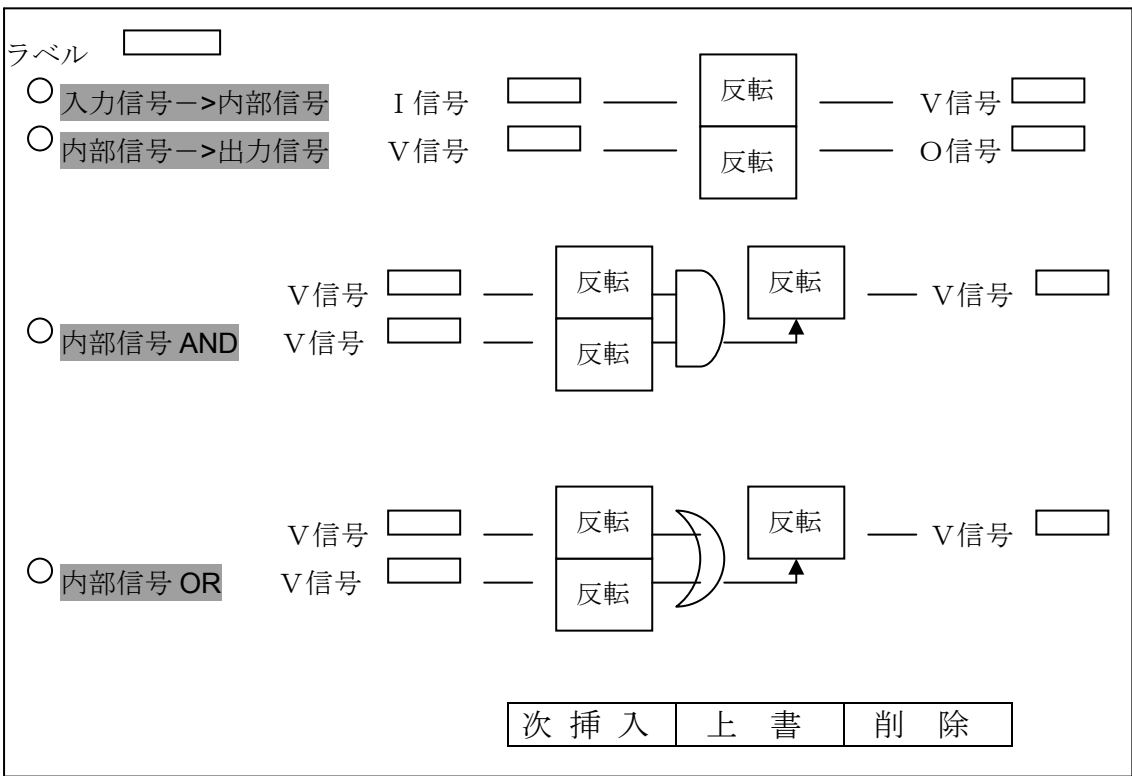
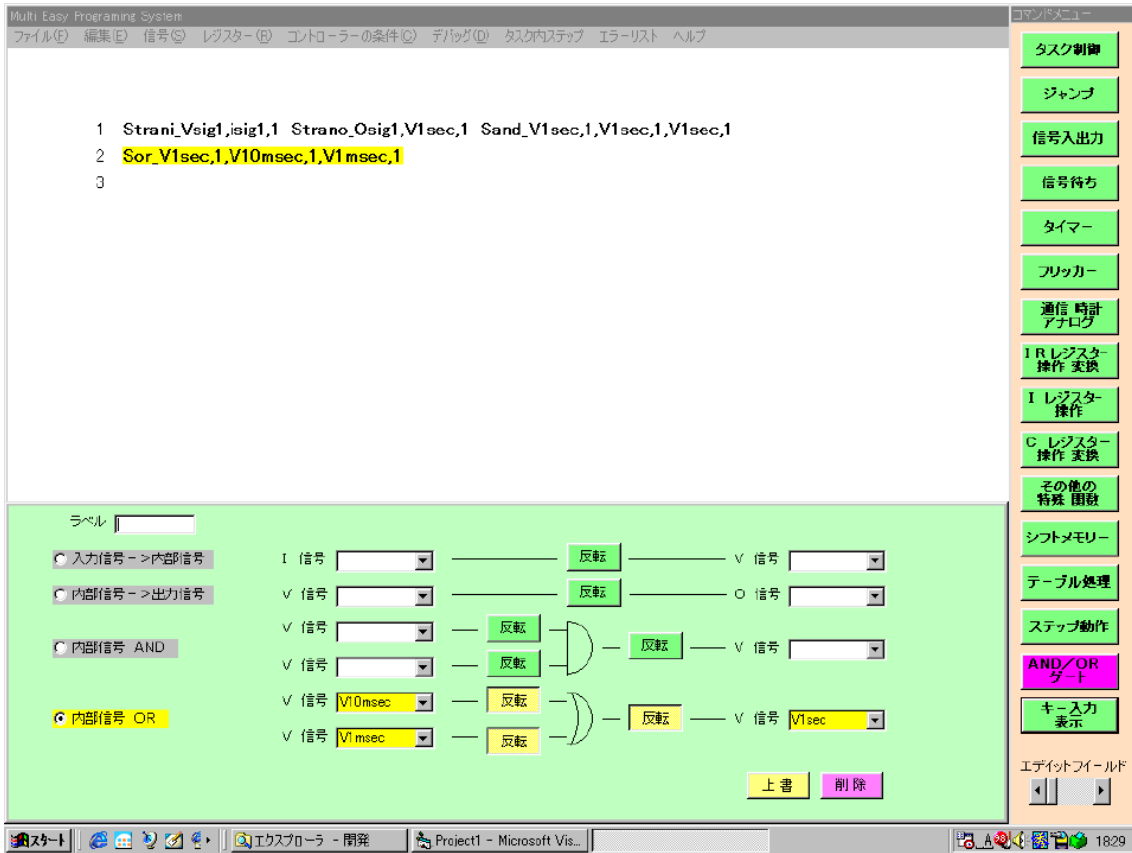
エディットフィールド

タスク1 | エクスプローラ - 開発 | Project1 - Microsoft Vis... | 1824



【図 17】 ステップ動作コマンドの選択画面

## 2.16 AND/ORゲートコマンド



【図18】AND/ORゲートコマンドの選択画面

## 2.17 キー入力表示コマンド

Multi Easy Programming System

ファイル(F) 編集(E) 信号(S) レジスタ(R) コントローラの条件(C) デバッグ(B) タスク内ステップ エラーリスト ヘルプ

コマンドメニュー

- タスク制御
- ジャンプ
- 信号入出力
- 信号待ち
- タイマー
- フリッカー
- 通信 時計 アナログ
- IR レジスタ 操作 変換
- I レジスタ 操作
- C レジスタ 操作 変換
- その他の特殊 関数
- シフトメモリー
- テーブル処理
- ステップ動作
- AND/OR ゲート
- キー入力表示

エディットフィールド

1 Ky\_i12 Kyw\_c20 Chk?\_c20,"a,"b,h'30,h'31,"A,"D Gt\_asss Dclr Dep\_2,2,"ABC Dspc\_2,2,10,c23  
 2 Drvs\_2,2,12 Numw\_2,2,12,c12,h Gt\_as Rvsset\_i11,2,2,4,3,9 **Rvsinc\_i11,2,2,4,3,9**

ラベル

キーステータス  へ即入力  キーコード入力待ち  へ押下後入力 ex. "0,"9 or h'30, h'39

コード範囲チェック  のコードがMIN<= <=MAXの範囲に無ければ、ラベル  ヘジャンプします。  ex. "0,"9 or h'30, h'39

画面クリア  固定メッセージ表示 ×  y  から、アスキーコード列  を表示します。 ex. 1,--,Space,ie,\_,Under\_Line

変数メッセージ表示 ×  y  から、変数  を先頭として  文字分、表示します。

反転表示 ×  y  から、  文字分、反転表示します。

数値入力表示 ×  y  から、  文字分、数値入力表示しながら、エントリーで、  を先頭として、メモリーセットします。 **16進入力**  
 クリアの場合は、true(not 0)となって、ラベル  ヘジャンプします。エントリーはジャンプしないで、続行します。 ex. 2, 3, 2, 5, 3, 2, 8, 3, 2

反転表示ポインターセット  I  で示す反転ポインターのセット。 各々 × y, length をセットし、最後はカンマは無し。

反転表示ポインターインクリメント  I **11** で示す反転ポインターのインクリメント。上記と同じデータをセットする。 **224339**

タスク - エクスプローラ - 開発 Project1 - Microsoft Vis... 2013

ラベル

○ キーステータス I  へ即入力      ○ キーコード入力待ち C  へ押下後入力

○ コード範囲チェック C  のコードが MIN<==MAX の範囲に無ければ、ラベル  へジャンプします。  
 ex, "0," "9" or h '30, h' 39

○ 画面クリア

○ 固定メッセージ表示 X  Y  から、アスキーコード列  を表示します。ex, "ABCDEFGHJKLM"

○ 反転表示 X  Y  から、 文字分、反転表示します。

○ 数値入力表示 X  Y  から、 文字分、数値入力表示しながら、エントリで、C  を先頭として、メモリセットします。クリアの場合は、true(not 0)となって、ラベル  へジャンプします。エントリはジャンプしないで、 16進入力  
 続行します。

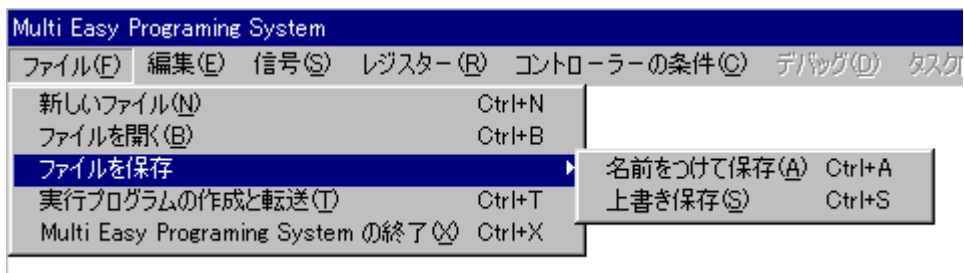
○ 反転表示ポインタセット I  で示す反転ポインタのセット。各々X,Y,length をセットし、最後コマは無し。  
 ex, 2, 3, 2, 5, 3, 2, 8, 3, 2

○ 反転表示ポインタインクリメント I で示す反転ポインタのインクリメント。上記と同じデータをセットする。

【図19】キー入力表示コマンドの選択画面

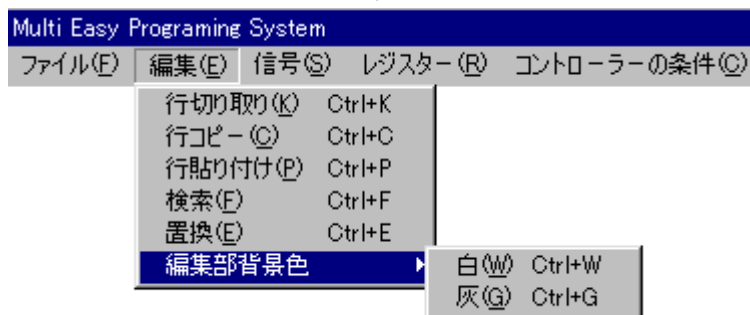
- キーコードについては、MEPS I/O コンパイラー 又は、MEPS コントローラー参照の事。
- ①コード範囲チェックの範囲データ入力で、“文字形式とh’の16進形式とmin/max 毎なら（ペアー毎なら）両方使用してかまいません。
- 数値入力表示で②の16進入力のスイッチがセットしているならば、0から9の次はA... Fとなります。詳細はMEPS I/O コンパイラーの12、LCD出力の中の Numw\_命令を参照の事。
- 反転表示ポインタセットと反転表示ポインタインクリメントとは、3バイトづつ同じデータ（位置と長さ）をセットし、インクリメント命令を実行する毎に、反転表示が変化します。

## 2.18 ファイルメニュー



ファイルの開閉、実行プログラムの作成と転送、MEPS の終了など、MEPS の操作コントロールに寄与するところが大です。

## 2.19 編集メニュー



行レベルでの編集と、各命令における検索、置換です。後述する編集機能もありミックスして使用してかまいません。編集部背景色は、デフォルトは白ですが、各命令毎ははっきり識別させるには、灰がいいです。

## 2.20 信号メニュー



各信号登録についての画面は、1 コマンドメニューとエディット画面の所のまず信号登録からというところで、説明済みです。

## 2.21 レジスターメニュー

| レジスター (R) |              | コントローラーの条件 (C) |   |
|-----------|--------------|----------------|---|
| C         | レジスター (Q)    | Ctrl           | Q |
| I         | レジスター (L)    | Ctrl           | L |
| R         | レジスター (H)    | Ctrl           | H |
|           | オーバータイマー (J) | Ctrl           | J |
|           | データメモリ (M)   | Ctrl           | M |

レジスターの内容はオンラインデバッグ時に確認できます。オーバータイマーやデータメモリの設定は、プログラム転送時、MEPS コントローラーに送信します。  
注釈は各命令選択画面において、レジスターNO におけるマウスの変化で表示します。

## 2.22 C文字レジスター (C00-C99)

| 文字レジスター |    |    |    |    |    |
|---------|----|----|----|----|----|
| 番号      | 内容 | 注釈 | 番号 | 内容 | 注釈 |
| 00      | 00 |    | 25 | 00 |    |
| 01      | 00 |    | 26 | 00 |    |
| 02      | 00 |    | 27 | 00 |    |
| 03      | 00 |    | 28 | 00 |    |
| 04      | 00 |    | 29 | 00 |    |

内容の左側は、16進表示2桁で、右側は、アスキー1文字表示です。

## 2.23 I正整数レジスター (I00-I99)

| 正整数レジスター |      |    |    |      |    |
|----------|------|----|----|------|----|
| 番号       | 内容   | 注釈 | 番号 | 内容   | 注釈 |
| 00       | 0000 | 0  | 25 | 0000 | 0  |
| 01       | 0000 | 0  | 26 | 0000 | 0  |
| 02       | 0000 | 0  | 27 | 0000 | 0  |
| 03       | 0000 | 0  | 28 | 0000 | 0  |
| 04       | 0000 | 0  | 29 | 0000 | 0  |

内容の左側は、16進4桁で、右側は、10進65535までの表示です。

## 2.24 R実数レジスタ（R00-R99）

| 実数レジスタ |             |    |    |             |    |
|--------|-------------|----|----|-------------|----|
| 番号     | 内容          | 注釈 | 番号 | 内容          | 注釈 |
| 00     | 0.000000+00 |    | 25 | 0.000000+00 |    |
| 01     | 0.000000+00 |    | 26 | 0.000000+00 |    |
| 02     | 0.000000+00 |    | 27 | 0.000000+00 |    |
| 03     | 0.000000+00 |    | 28 | 0.000000+00 |    |
| 04     | 0.000000+00 |    | 29 | 0.000000+00 |    |

内容は実数±0. DDDDDDE±NNの浮動小数点形式で、±0. DDDDDD (000000-999999) 有効数字10進6桁×±NN (+63-----64) 乗の範囲です。

## 2.25 オーバータイマー値（00-99）

| オーバータイム定数 |    |    |    |    |    |
|-----------|----|----|----|----|----|
| 番号        | 内容 | 注釈 | 番号 | 内容 | 注釈 |
| 00        | 0  |    | 25 | 0  |    |
| 01        | 0  |    | 26 | 0  |    |
| 02        | 0  |    | 27 | 0  |    |
| 03        | 0  |    | 28 | 0  |    |
| 04        | 0  |    | 29 | 0  |    |

内容は0から65535の10進数で、×10mSecのタイマー値となります。  
タイマーの確保ではありません。定数の種類が100とれるという事です。

## 2.26 データメモリー（D00-D99）

| データメモリー |    |    |    |    |    |
|---------|----|----|----|----|----|
| 番号      | 内容 | 注釈 | 番号 | 内容 | 注釈 |
| 00      | 0  |    | 25 | 0  |    |
| 01      | 0  |    | 26 | 0  |    |
| 02      | 0  |    | 27 | 0  |    |
| 03      | 0  |    | 28 | 0  |    |
| 04      | 0  |    | 29 | 0  |    |

内容は0から65535の10進数です。D00-D99はこのように設定できます。  
D100からD9999はデータロギング用です。

## 2.27 コントローラーの条件メニュー

The screenshot shows a software window titled 'コントローラーの条件' (Controller Conditions). It is divided into three main sections. The top section is for 'CH1 通信条件' (CH1 Communication Conditions), the middle for 'CH2 通信条件' (CH2 Communication Conditions), and the bottom for 'ADC/DAC' and '表示スピード' (Display Speed). Each communication condition section has identical settings: Baud Rate (4800, 9600, 19200, 38400), Data Bits (7, 8), Stop Bits (1, 2), Parity Check (None, Even, Odd), and Receive Character Interval Maximum Time (20 ms). The terminal emulation section has options for No, CR, LF, and ETX. The ADC section has options for No and 1-8. The DAC section has options for No and 1-2. The display speed section has options for Slow and Fast (but task cycle is slow).

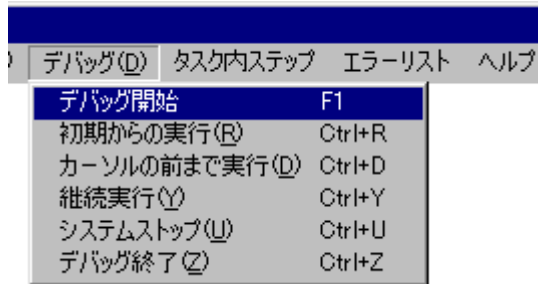
| Section  | Parameter              | Options                  | Selected |
|----------|------------------------|--------------------------|----------|
| CH1 通信条件 | ボーレート                  | 4800, 9600, 19200, 38400 | 38400    |
|          | データビット                 | 7, 8                     | 8        |
|          | ストップビット                | 1, 2                     | 1        |
|          | パリティチェック               | 無し, 偶数, 奇数               | 無し       |
|          | 受信キャラクター間最大タイム         | 20 ms                    | 20       |
| CH2 通信条件 | ボーレート                  | 4800, 9600, 19200, 38400 | 38400    |
|          | データビット                 | 7, 8                     | 8        |
|          | ストップビット                | 1, 2                     | 1        |
|          | パリティチェック               | 無し, 偶数, 奇数               | 無し       |
|          | 受信キャラクター間最大タイム         | 20 ms                    | 20       |
| ADC      | 無し                     | 無し                       | 無し       |
|          | 1, 2, 3, 4, 5, 6, 7, 8 | 1-8                      | 8        |
| DAC      | 無し                     | 無し                       | 無し       |
|          | 1, 2                   | 1-2                      | 2        |
| 表示スピード   | 遅い                     | 遅い, 早い(但し、タスク周期は遅い)      | 遅い       |
|          | 早い                     | 遅い, 早い(但し、タスク周期は遅い)      | 遅い       |

CH1、CH2の通信条件の設定とADC、DACのオプション設定、MEPSコントローラーの表示スピード設定です。オンラインデバッグ時は、上図CH1の設定をして下さい。

表示スピードは早くすると、表示プログラムの占有度が多くなり、他タスクのスキャン時間が遅くなります。(MEPSコントローラーのその他のチェック画面の、Max Cycle Tにて、ユーザープログラムのタスク1周時間がわかります。)

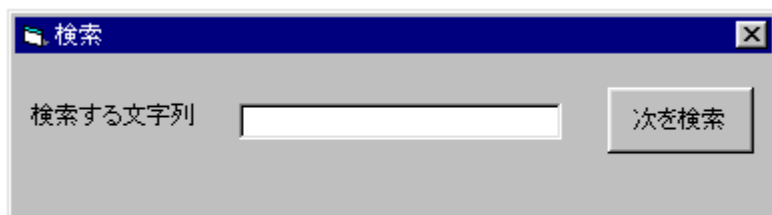


## 2.28 デバッグメニュー



- まず、プログラムのコンパイルでエラー無しを確認後、MEPSコントローラーとRS232Cケーブルでつなぎ、通信条件を合わせて、転送します。
- デバッグ開始を押しますと、デバッグモードになります。キー入力は出来ません。又、編集機能等は、出来ません。
- 初期からの実行を押しますとユーザープログラムの初期からの実行となります。
- カーソルの前まで実行は、黄色の命令の前まで、実行して、ブレークします。信号、レジスター等の内容を確認出来ます。次々と使用できますし、タスク内ステップにも移行できます。
- 継続実行は、現在の停止状態から、続行させます。ブレークはしません。
- システムストップは、強制的に停止状態になります。**タスク内で停止していないので黄色の命令の前まで実行したのではありません。**したがってタスク内ステップに移行出来ません。
- デバッグ終了は、停止状態であるならば、デバッグモードを終了させ、編集モードになります。
- タスク内ステップは、デバッグモード中で、初期からか、ブレーク後から操作可能で、タスク内のみ、ステップ動作をして、信号、レジスター等の内容を確認出来ます。したがって、別タスクへの移行は、その別タスクのカーソルの前まで実行させて、タスク内ステップへと移行させます。
- タスク内ステップも条件待ち命令等で、条件が整っていない時は停止しません。容易に条件が整わないときは、一旦システムストップで停止させて、原因を調べたほうが良いでしょう。
- カーソルの前まで実行や、タスク内ステップで、**LCD表示命令を実行させた直後は、まだ表示されていません。**理由としては、システムに表示タスクがあり、この表示タスクにて、表示をさせますが、ユーザープログラムを優先させていますので、遅くなります。100msぐらいの遅延後なら表示されます。

## 2.29 編集メニューからの検索、置換

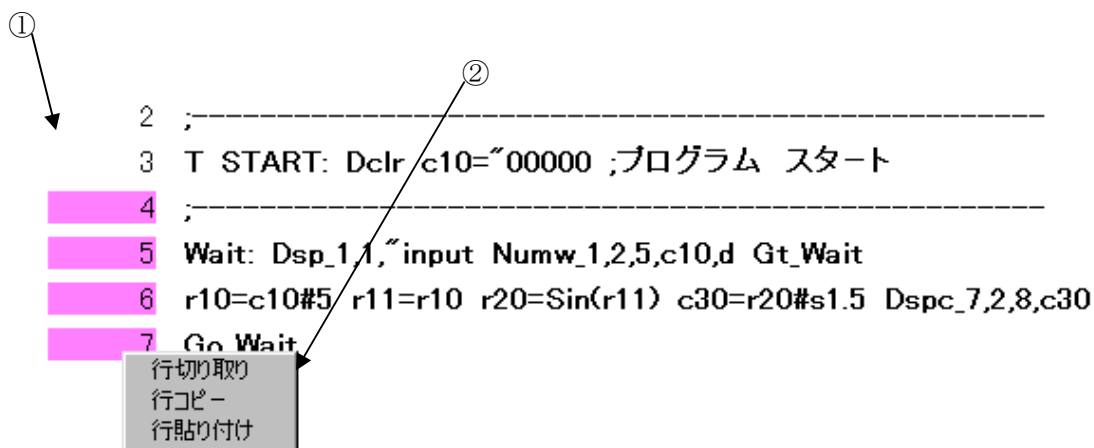


- エディットフィールドにおける検索文字列をサーチします。



- エディットフィールドにおける検索文字列を置換後の文字列に置換します。
- すべて置換で一斉に置換します。

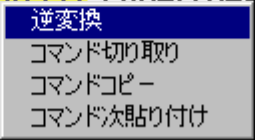
## 2.30 行表示からの編集



- エディットフィールドの左部分、行表示エリアの、ある行で一回マウス左クリックして下方ヘスライド（ドラッグではありません）すると、上図①の如くピンク色のエリアが出来ます。もう一度クリックして、ピンクエリアを確定すると、次は右クリックして、②の如き編集メニューを出して、左クリックで、行全体の切り取り、コピー、貼り付けを行う事が出来ます。これは19、編集メニューからの行全体の切り取り、コピー、貼り付けと同じです。

### 2.31 各命令からの編集

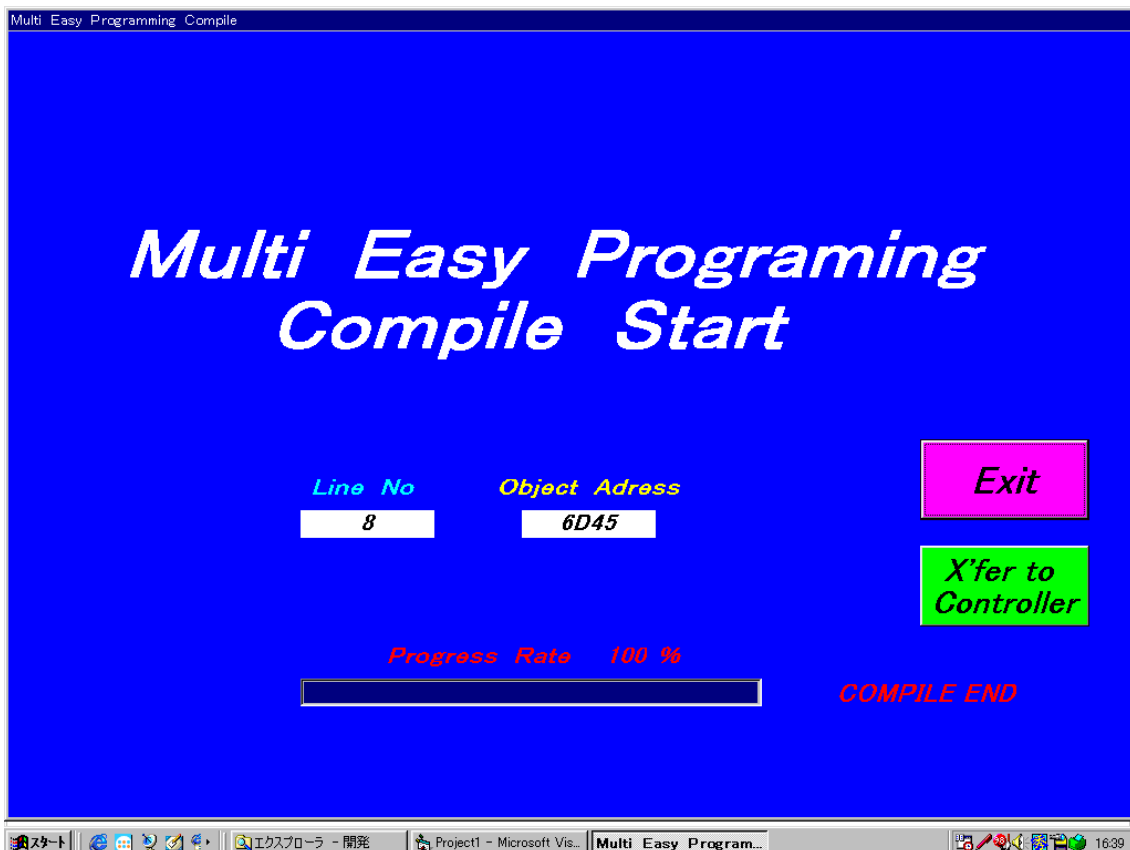
```
2 ;-----  
3 T START: Dclr c10="00000 ;プログラム スタート  
4 ;-----  
5 Wait: Dsp_1,1,"input Numw_1,2,5,c10,d Gt_Wait  
6 r10=c10#5 r11=r10 r20=Sin(r11) c30=r20#s1.5 Dspc_7,2,8,c30  
7 Go_Wait  
8 E  
9
```



A context menu is shown over line 6 of the code. The menu items are: 逆変換 (Inverse Conversion), コマンド切り取り (Cut Command), コマンドコピー (Copy Command), and コマンド次貼り付け (Paste Command Next). The '逆変換' option is highlighted in blue.

- 各命令の右クリックで、上図のメニューが出ます。左クリックで、コマンド切り取り、コピー、次貼り付けを実行出来ます。
- 逆変換では、各命令に対応する記述画面へ逆変換します。

### 2.32 実行プログラムの作成と転送



- コンパイル画面と、X'fer to Controller ボタンでME P Sコントローラーへの転送を行います。

### 3. I/Oコンパイラ命令

<本説明中使用記号>

\*\*\*\*\* @@@@@@ プログラムラベル名 (1から最大8文字)  
..... 信号名 (1から最大8文字) 入出力内部信号区別なく。ここで信号とリレーと同じ意味とします。  
I..... 入力信号名 (1から最大8文字)  
O..... 出力信号名 (1から最大8文字)  
V..... 内部信号名 (1から最大8文字)  
Vx..... 内部信号名 (1から最大8文字)  
Vy..... 内部信号名 (1から最大8文字)  
Vz..... 内部信号名 (1から最大8文字)  
XX YY ZZ TT No(各2文字固定)  
NNNN DDD HH 定数 (1から最大各文字数、HHは16進1バイトあらわすときは2文字固定)

i j k n 各パラメーター (1から最大各文字数)  
: は、\*\*\*\*\*: の形式でプログラムのラベル (ジャンプ先のアドレスを示す)。  
; は、その行の改行か、又は再度;の位置までの記述を、コメント (注釈) とし、プログラムに關与しません。

ex、緑部分はコメント

```
      ; a b c d e f g h i j k  
----- ; a b c ; ----
```

+、-、\*、/ は加減乗除演算子、&、@ はAND、OR演算子

= は左辺の式を右辺へ演算、又は変換をして代入します。

\_ は左部のコマンドに対して、右部の詳細仕様に基いたコマンドを実行します。

または、文字列中ではスペースを意味します。

# はパラメーター等を表現する補助の区切り記号です。

“ は次に続く文字が文字そのものを意味します。

h' は次に続く文字が16進コードを意味します。

<注意点>

- I/Oコンパイラの文字の大文字、小文字の識別はありません。但し、ジェネレーターで発生させる命令は、第1文字を大文字あと小文字と、本説明と一致させています。
- 命令の区切りはスペースか、改行です。

- ラベル名に “=” 又は “\_” アンダーラインは使用不可です。
- “?” の文字のコマンドに対する判定は `True` (1) か、`False` (0) で、チェックするものです。
- “S” “w” の文字のものはタスクを切替て (スイッチしながら) 全タスクを、スキャンして行きます。wはその所 (アドレス) でとどまってあたかも `Wait` (待っている) している様になります。
- それ以外 “\$” は送信を意味し送信完了まで `Wait` します。  
“%” は受信を意味し受信完了まで `Wait` します。

### 3.1 タスク命令

#### ① T

以下に示すプログラムが、1つの独立タスクである事を示します。

Tの後、1スペース以上あけて、タスクのエントリーラベル **\*\*\*\*\*** : を書かなければなりません。

プログラムは、まず、この T から、始まります。

```
e x、  
      T P 1 : -----  
                -----  
      T P 2 : -----  
                -----  
      T P 3 : -----  
                -----
```

T でプログラムを分け、それぞれをタスクと呼びます。タスクが2個以上の場合、マルチタスクと呼びます。

もちろん1個の場合もシングルタスクと呼び、動作します。

本コンパイラの特長は、各タスクは、フローチャートどおりに命令を書ける事です。しかし、各タスクは、並行処理をします。

したがって、タスクを分ける目安は、プログラムの流れが、なるべく独立していそうな事です。

完璧に独立しているという事は、少なく、作成中にも因果関係が増えていくものです。しかし、たとえば、ある信号を2個のタスクで使っていて、意図的ではないにしろ、片方ではONし、片方ではOFFしているというような事は、避けた方が良いでしょう。

メモリーも同じものでは、タスクの切り替え後は、こわれる事を前提に使用しなければなりません。

タスクの分け方は、経験上、上達するものです。しかし、これがプログラムの優劣を決め、しいては、システムの信頼性にも左右するという事です。

一般的にはタスクの記述の仕方に主に2通り有ります。一つ目は 起動されたタスクをループ構造にする事。二つ目は必要な仕事をした後、消滅する構造にする事。下記A、Bタスク

```
T  A : -----ループ
```

```
T  B : -----X
```

Aのタスクの場合 そのループ内に必ず S スイッチングか、wait系の命令 (wの文字が付いている命令) を含んでいなければならない。

<質問>、あるタスクから、ジャンプ命令などで、別のタスクに連結させた構造での動作は可能ですか？

可能です。筋書きどりの動作をします。プログラムは、サブルーチンではなく、共通ルーチン (C o - R o u t i n e) と変化しますが、タスクが1つになったのではなく各固有のタスクが、各固有のタイミングで動作するので、上述の如く信号、メモリーの重複 (オーバーラップ) に注意しなければなりません。

② A c t \_\*\*\*\*\*

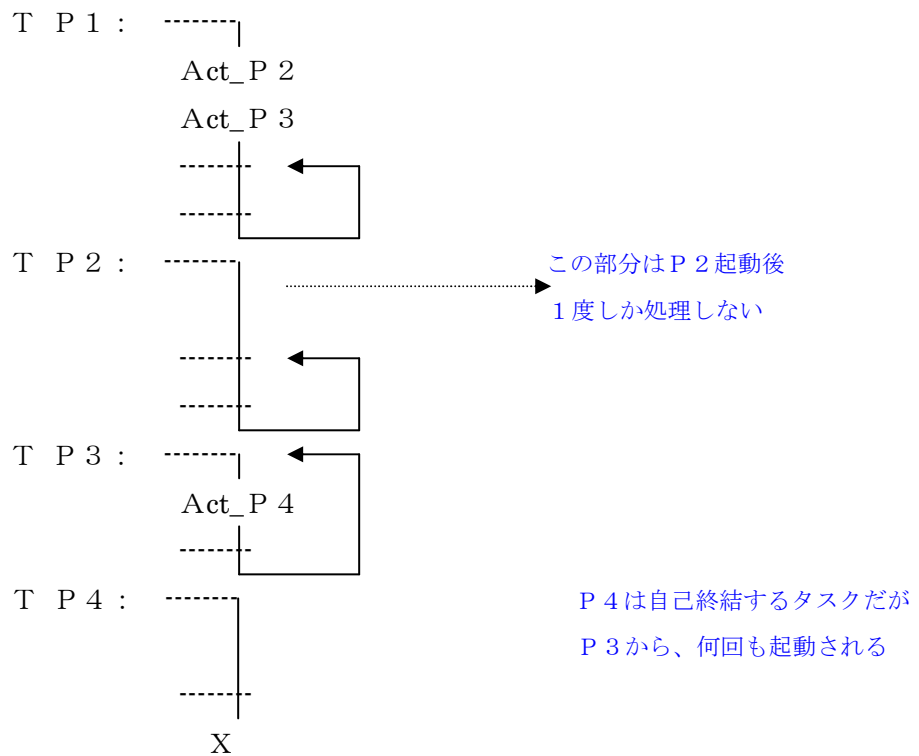
T \*\*\*\*\* : で始まるタスクの起動。又は再開 (P a u s e の後は再開、R e s e t の後、又は電 ON 後は、そのタスクの最初から起動)。

タスクが起動中ならば、無視されます。P a u s e の後、R e s e t の後か、タスクが起動中でない事を確認して、使用します。

プログラム最初の、T \*\*\*\*\* : は、初期タスクとして、M E P S コントローラシステムが、電 ON 後、唯一自動的に起動します。

他のタスクは、この初期タスクから、起動しておくか、ある時点で、起動するか、ですが、この初期タスクで一斉に起動するほうがわかり易いと思います。

e x、



→ は、ジャンプ命令等によるループ

上記 ex、は代表的な構造例ですが、この他にも色々な応用があります。

③ P a u s e \_\*\*\*\*\*

\*\*\*\*\* で始まるタスクを休止する。、 A c t \_\*\*\*\*\* で再開します。

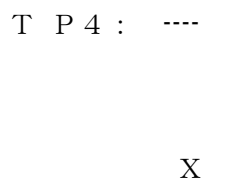
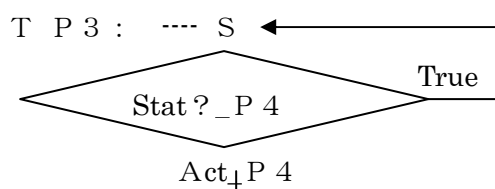
④ R e s e t \_\*\*\*\*\*

\*\*\*\*\*で始まるタスクをリセットする。 A c t \_\*\*\*\*\* でそのタスクの最初 (先頭) から処理します。

⑤ S t a t ? \_\*\*\*\*\*

\*\*\*\*\*で始まるタスクの状態が、起動中又は休止中 (True) か、リセット中 (False) かをチェックする。

ex、



P 4 が自己終結するまで、P 3 は待つ、P 4 が終結したら、再度、P 4 を起動する。

⑥ S

タスクのスイッチング (他タスクへ実行権を移します)。

w文字の付く命令も、w a i tをしながら他タスクへ実行権を移します。

もしも、ジャンプ命令などでループしているプログラム中、本命令か、w文字の付く命令がなかったら、そのループ中は、シングルタスクの様相を、呈します。

⑦ X

自己タスクを自分でリセットする (自己終結)。

C a l l されたサブルーチンの中で使用しても、そのサブルーチンを今コールしたタスクがリセットします。

⑧ E

プログラム全体の終了を示します。最後に必ず要ります。



## 3.2 ジャンプ命令

### ① G o \_\*\*\*\*\*

無条件ジャンプ

### ② G t \_\*\*\*\*\*

条件ジャンプ。0でない、つまり真偽値が真 True の時、ジャンプします。当然反対条件はこの命令の次へと続行します。

?文字を含む命令と N u m w 命令の直後に使用します。

信号チェック命令では、O N でジャンプします。

タイマーチェック命令 (T m u p ?) では、0でない状態でジャンプします。

コード範囲チェック命令 (C h k ?) では、範囲外なら、ジャンプします。

数値入力命令 (N u m w) では、クリアキー押下で、ジャンプします。エントリーキーでは、続行します。

タスク状態チェック命令 (S t a t ?) では、起動中又は休止中でジャンプします。

### ③ G f \_\*\*\*\*\*

条件ジャンプ。0、つまり真偽値が偽 False の時、ジャンプする。当然反対条件はこの命令の次へと続行します。

?文字を含む命令と N u m w 命令の直後に使用します。

信号チェック命令では、O F F でジャンプします。

タイマーチェック命令 (T m u p ?) では、0でジャンプします。

コード範囲チェック命令 (C h k ?) では、範囲内なら、ジャンプします。

数値入力命令 (N u m w) では、エントリーキー押下で、ジャンプします。クリアキーでは、続行します。

タスク状態チェック命令 (S t a t ?) では、リセット状態、又は終結状態でジャンプします。

### ④ C a l l \_\*\*\*\*\*

サブルーチンコール (ネスティング数最大32) R コマンドで元のルーチンに戻り続行します。

<質問>、サブルーチンの記述はどこに書けば良いのですか？

本コンパイラーのサブルーチンは全てグローバルなので、最初のタスクと、Eの間どこに記述しても良いです。まとめても良いし、ローカルなものは、タスク記述の側においても良いでしょう。

⑤ R

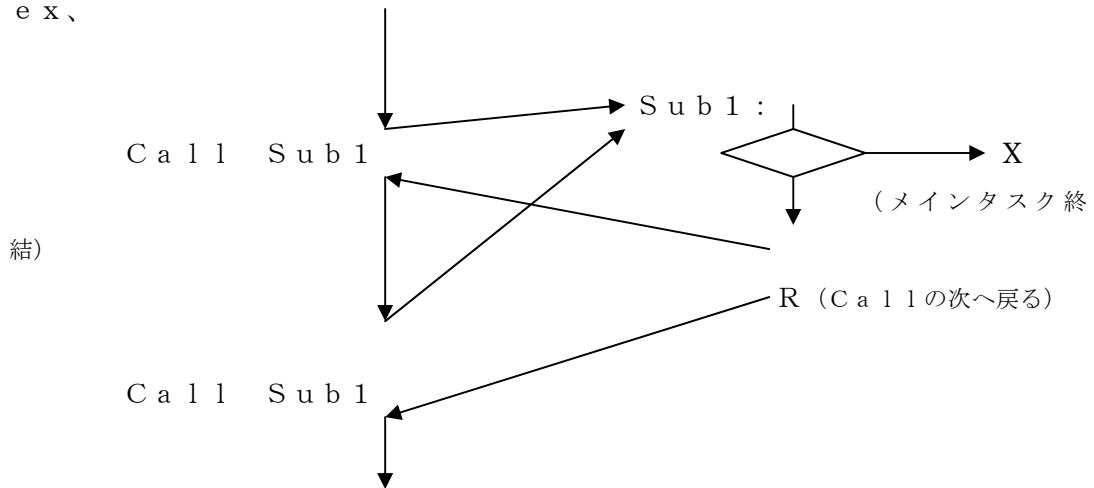
C a l l 命令でコールされたサブルーチンからのリターン。

サブルーチンからの出口に必要です。

タスクのプログラム中では、使用してはいけません。暴走原因となります。

必ず、C a l l 命令でコールされるサブルーチンの中だけ、使用します。

e x、



⑨ 正整数の I F 文

もしも、I X X 正整数、I Y Y 正整数、D D D D D (0-65535) の下記条件が満たされるなら、ラベル\*\*\*\*\*へジャンプし、満たされないなら、次の命令に続行します。

- |                                  |               |
|----------------------------------|---------------|
| I f _ i X X = i Y Y, *****       | 一致すれば、ジャンプ    |
| I f _ i X X = D D D D D, *****   |               |
| I f _ i X X < > i Y Y, *****     | 不一致ならば、ジャンプ   |
| I f _ i X X < > D D D D D, ***** |               |
| I f _ i X X > = i Y Y, *****     | より大イコールならジャンプ |
| I f _ i X X > = D D D D D, ***** |               |
| I f _ i X X < = i Y Y, *****     | より小イコールならジャンプ |
| I f _ i X X < = D D D D D, ***** |               |
| I f _ i X X > i Y Y, *****       | より大ならジャンプ     |
| I f _ i X X > D D D D D, *****   |               |
| I f _ i X X < i Y Y, *****       | より小ならジャンプ     |
| I f _ i X X < D D D D D, *****   |               |

⑪ 実数の I F 文

もしも、R X X 実数、R Y Y 実数の下記条件が満たされるなら、ラベル\*\*\*\*\*へジャンプし、満たされないなら、次の命令に続行します。

|                              |               |
|------------------------------|---------------|
| i F _ r X X = r Y Y, *****   | 一致すれば、ジャンプ    |
| i F _ r X X < > r Y Y, ***** | 不一致ならば、ジャンプ   |
| i F _ r X X > = r Y Y, ***** | より大イコールならジャンプ |
| i F _ r X X < = r Y Y, ***** | より小イコールならジャンプ |
| i F _ r X X > r Y Y, *****   | より大ならジャンプ     |
| i F _ r X X < r Y Y, *****   | より小ならジャンプ     |

⑫ S e l \_ c X X, "D, \*\*\*\*\*, -----, h'HH, @@@@, -----

選択ジャンプです。C X X 文字レジスターが、文字Dなら、ラベル\*\*\*\*\*へ、ジャンプし、16進でHHなら、ラベル@@@@@へ、ジャンプします。

“ と h' ” と混合して使用してもかまいません。それ以外は、次命令を続行します。  
文字又は16進コードとラベルはペアーです。

<質問>、どれだけの長さ書くことができますか？

制限はありません。

### 3.3 I/O命令

#### 3.3-1 )、 信号チェック命令

以下の命令の即、次の命令に条件ジャンプを使用します。

- ① I ?\_I..... (最大8文字)  
I..... の入力信号の ON (True), OFF (False) をチェックする。
  
- ② O ?\_O.....  
O..... の出力信号の ON (True), OFF (False) をチェックする。
  
- ③ V ?\_V.....  
V..... の内部信号の ON (True), OFF (False) をチェックする。

#### 3.3-2 )、 信号セット命令

- ④ O n \_O.....  
O..... の出力信号をONする。
  
- ⑤ O f \_O.....  
O..... の出力信号をOFFする。
  
- ⑥ V n \_V.....  
V..... の内部信号をONする。
  
- ⑦ V f \_V.....  
V..... の内部信号をOFFする。

#### 3.3-3 )、 信号待ち命令

待つといってもエッジを待つのではなく、あくまでもステータスHi、Loを待つのです。だから、条件がみたされていれば、即、続行します。

信号待ち命令には3パターン有ります。無条件に信号を待つのが1つ。それとタスクには、固有の1つのタイマーが有り、それを利用しての、オーバータイムチェック付きの待ち方に2つ有ります。以下、I n w命令で詳しく説明しますが、他の命令も同様です。

- ① I n w \_I.....  
I..... の入力信号のONを待ちます。

待つというのは他タスクの処理をしながら待つという事です。条件が整えば次の命令へと続行します。

`Inw_I.....` は、無条件にONを待ちます。

`Inw_TT, *****`, `I.....` この型は `I.....` の入力信号がONになるのを待つのですが、`TT`で示されたタイマー定数`No` (00 から 99 までの 100 個の定数の内の1つ) に設定された内容値をタスクタイマーにセットし、`値×10ms`の時間が過ぎると、この命令を中止して、`*****`のラベル名へジャンプします。時間内に信号が、ONすれば、次の命令へと続行します。

`Inw_c, *****`, `I.....` この型は、以前に`??w_TT, *****`,.....で設定されたタスクタイマーをそのまま続けて使用します。`c`はContinueの意味です。つまり同一タスク内、いくつかの信号待ちを最初にTTで設定したタスクタイマーで一括してチェックするというものです。

`TT`の値が0の時は即チェックをしてエラージャンプか、次の命令を続行か、します。信号チェック命令に相当します。

ex、

```
----- On_Orelay1 Inw_98,Error1,Isens1 Ifw_c,Error1,Isens1
Inw_c,Error1,Isens2 Of_Orelay1 -----
Error1: Of_Orelay1 -----
```

出力 `Orelay1` を ON してタイマー定数 `No98` が 100 なら 1 秒間、入力 `Isens1` の ON OFF をチェックし、そのあと入力 `Isens2` の ON をチェックし、OK なら、出力 `Orelay1` を OFF して、プログラムは続きます。1 秒オーバーすれば、`Error1` へジャンプして、出力 `Orelay1` を OFF して、エラー処理としてのプログラムは続きます。

<質問>、`TT`のタイマー定数の設定が、なぜ間接的なのですか？

経験上、直接だと調整段階でプログラムを何回も書きかえることになり、間接だとコントローラだけの調整ですませることが出来ます。

## ② `Ifw_I.....`

`I.....` の入力信号のOFFを待ちます。

`Ifw_I.....` は、無条件にOFFを待ちます。

`Ifw_TT, *****`, `I.....` タイマー条件付きで、OFFを待ちます。タイムオーバーで`*****`へジャンプします。

`Ifw_c, *****`, `I.....` タイマー継続条件付きで、OFFを待ちます。

タイムオーバーで\*\*\*\*\*へジャンプします。

③ V n w \_ V . . . . .

V . . . . . の内部信号のONを待ちます。

V n w \_ V . . . . . は、無条件にONを待ちます。

V n w \_ T T , \* \* \* \* \* , V . . . . . タイマー条件付きで、ONを待ちます。

タイムオーバーで\*\*\*\*\*へジャンプします。

V n w \_ c , \* \* \* \* \* , V . . . . . タイマー継続条件付きで、ONを待ちます。

タイムオーバーで\*\*\*\*\*へジャンプします。

④ V f w \_ V . . . . .

V . . . . . の仮想信号のOFFを待ちます。

V f w \_ V . . . . . は、無条件にOFFを待ちます。

V f w \_ T T , \* \* \* \* \* , V . . . . . タイマー条件付きで、OFFを待ちます。

タイムオーバーで\*\*\*\*\*へジャンプします。

V f w \_ c , \* \* \* \* \* , V . . . . . タイマー継続条件付きで、OFFを待ちます。

タイムオーバーで\*\*\*\*\*へジャンプします。

⑤ O n w \_ O . . . . .

O . . . . . の出力信号のONを待ちます。

O n w \_ O . . . . . は、無条件にONを待ちます。

O n w \_ T T , \* \* \* \* \* , O . . . . . タイマー条件付きで、ONを待ちます。

タイムオーバーで\*\*\*\*\*へジャンプします。

O n w \_ c , \* \* \* \* \* , O . . . . . タイマー継続条件付きで、ONを待ちます。

タイムオーバーで\*\*\*\*\*へジャンプします。

⑥ O f w \_ O . . . . .

O . . . . . の出力信号のOFFを待ちます。

O f w \_ O . . . . . は、無条件にOFFを待ちます。

O f w \_ T T , \* \* \* \* \* , O . . . . . タイマー条件付きで、OFFを待ちます。

タイムオーバーで\*\*\*\*\*へジャンプします。

O f w \_ c , \* \* \* \* \* , O . . . . . タイマー継続条件付きで、OFFを待ちます。

タイムオーバーで\*\*\*\*\*へジャンプします。

### 3.3-4)、パラレルI/O命令

#### ① P i \_n, i X X & h ' H H

入力ポート n (0-3) から入力した 8 ビットに H H の 16 進データと AND をとり i X X レジスタの下位へストアします。(上位は 0 になる)

#### ② P o \_n, i X X & h ' H H

出力ポート n (0-3) へ i X X レジスタの下位 8 ビットに H H の 16 進データと AND をとり出力する。H H のマスクビット以外のビット出力 (0 となるビット出力) には変動させません。つまり H H はポート出力の本命令の有効ビットを指定し、H H の 0 となるビット出力には関係ないように出力します。

### 3.3-5)、ゲート命令

#### ① S t r a n i \_V....., I....., n

I..... 入力信号を V..... 内部信号へ、n=0 なら、そのまま、n=1 なら反転して、伝達します。

#### ② S t r a n o \_O....., V....., n

V..... 内部信号を O..... 出力信号へ、n=0 なら、そのまま、n=1 なら反転して、伝達します。

#### ③ S a n d \_V z....., k, V x....., i, V y....., j

V x..... 内部信号を i=0 なら、そのまま、i=1 なら反転して、  
V y..... 内部信号を j=0 なら、そのまま、j=1 なら反転して、  
上記 2 信号の AND をとって、k=0 なら、そのまま、k=1 なら反転して、  
V z..... 内部信号へ伝達します。

#### ④ S o r \_V z....., k, V x....., i, V y....., j

V x..... 内部信号を i=0 なら、そのまま、i=1 なら反転して、  
V y..... 内部信号を j=0 なら、そのまま、j=1 なら反転して、  
上記 2 信号の OR をとって、k=0 なら、そのまま、k=1 なら反転して、  
V z..... 内部信号へ伝達します。

### 3.4 タイマー命令

- ① `Tmset__n, TT`  
n (0~9) でグローバルタイマー10個の内、1個選択し。TT (00-99) でしめされたタイマー定数Noの内容値をセットします。単位は10ms。
- ② `Tmup?__n`  
n (0~9) のグローバルタイマーがタイムアップかどうかのチェックをし、タイムアップしていれば真偽値が0 (False)、していなければ0以外 (True) です。  
直後に条件ジャンプ命令が必要です。  
Tmset と Tmup? は、別タスク間のグローバルなタイマーとして使用し、たいていは、タスク固有1個のタイマーを使用する以下の命令を多用します。
- ③ `Dlyw__NNNNN`  
タスク固有タイマーを使用してNNNNN (0-65535) × 10ms の遅延を行います。  
Nが0なら遅延は有りません。
- ④ `Dlywi__iXX`  
タスク固有タイマーを使用してiXX整数レジスターの内容値で遅延を行う。  
内容が0なら遅延は有りません。
- ⑤ `Dlywn__TT`  
タスク固有タイマーを使用してTT (00-99) でしめされたタイマー定数Noの内容値で遅延を行います。内容が0なら遅延は有りません。



### 3.5 通信

- ① \$\_\_cXX, NN, n  
n (0-1) chへ、cXXからNN文字送信する。送信完了までWAITする。
- ② %\_\_cXX, NN, n  
n (0-1) chから、受信を待ち、受信完了したらNN文字受信した文字列をcXXからストアする。

### 3.6 アナログ入出力

- ① Ad\_\_iXX, n  
n (0-7) chから、iXXへ、0-4095 (0-4.095V) 入力します。
- ② Da\_\_iXX, n  
n (0-1) chへ、iXXから、0-4095 (0-5V) 出力します。

### 3.7 時計入力

- ① Clock\_\_cXX  
カレンダークロックから読み出し、cXXを先頭として、cXX+11まで、年月日時分秒12文字セットします。

### 3.8 演算

#### ① 正整数レジスタの演算

正整数レジスタは、0-65535までの正整数です。

##### a)、四則演算

|                               |                   |
|-------------------------------|-------------------|
| $iZZ = NNNNN$ (0-65535)       | データセット            |
| $iZZ = iXX$                   | コピー               |
| $iZZ = iXX + NNNNN$ (0-65535) | $iZZ = iXX + iYY$ |
| $iZZ = iXX - NNNNN$ (0-65535) | $iZZ = iXX - iYY$ |
| $iZZ = iXX * NNNNN$ (0-65535) | $iZZ = iXX * iYY$ |
| $iZZ = iXX / NNNNN$ (1-65535) | $iZZ = iXX / iYY$ |

- 掛け算では、 $iZZ + 1$ にも上位の答えが入ります。
- 割り算では、被除数  $iXX + 1$  (上位)  $iXX$  (下位) を必ずセットの事。余りが  $iZZ + 2$ 、商が  $iZZ + 1$  (上位)  $iZZ$  (下位) となります。
- 割り算で分母が0の時、答えは0になります。

65535に+1すると、65536で0になります。又、0から-1すると

65535になります。つまり32768以上を-とすると、

+32767-----32768の範囲なら、正負数の+-演算が出来ます。(\* /は不可)

数値の対応は下記の様になります。

32767, 32766, -----, 1, 0, -1, -2, -----, -32767, -32768

32767, 32766, -----, 1, 0, 65535, 65534, -----, 32769, 32768

このとき、演算前に例えば、 $i01 = -2$ は出来ないので、 $i01=0$   $i01=i01-2$

として下さい。又結果も、 $i01$ なら **If\_i01>=32768,Minus i00=1 -----**

**Minus : i00=0 i01=i00-i01 -----** の如く

$i01$ を32768と比較して以上なら、Minusへジャンプして、負の意味で*i00*に

0をセットし、 $i01$ を絶対値変換します。32767以下なら、 $i01$ はそのまま

正の意味で*i00*に1をセットします。

##### b)、論理演算

I 正整数レジスタ0-65535を0-FFFFの16進数とみなします。

AND

$iZZ = iXX \& iYY$

$iZZ = iXX \& h'HHHH$

OR

$iZZ = iXX @ iYY$

$iZZ = iXX @ h'HHHH$

HHHHは最大4桁の16進数 (0-FFFF)

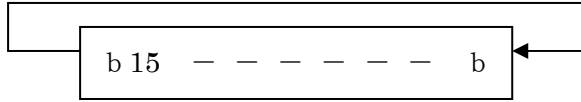
c)、ローテイトシフト

I 正整数レジスタは、16ビットのレジスタで2進数とみなします。

$n$  は回数で1~15

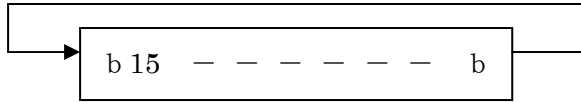
$$i Z Z = R L ( i X X, n )$$

RLはローテイトレフト



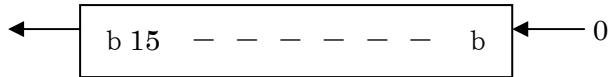
$$i Z Z = R R ( i X X, n )$$

RRはローテイトライト



$$i Z Z = S L ( i X X, n )$$

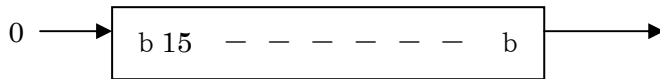
SLはシフトレフト



シフトするたびにオーバーフローしなければ、2倍、4倍、8倍、-----となります。続けるとオーバーフローしてゆき最後は0になります。

$$i Z Z = S R ( i X X, n )$$

SRはシフトライト



シフトするたびにアンダーフローしなければ、1/2倍、1/4倍、1/8倍、-----となります。但し、小数点以下はないので、切り捨てになり続けるとアンダーフローしてゆき最後は0になります。

d)、10進2進変換

0~9999までの数

$$i Z Z = D ( i X X )$$

2進から10進変換

$$i Z Z = B ( i X X )$$

10進から2進変換

I 正整数レジスタは基本的に2進数(16進数)であるが、C文字レジスタ列の様に、10進形式も例外的に必要で、そのとき4桁9999までです。

② 実数レジスタ同志の演算

実数は±0. DDDDDDE±NNの浮動小数点形式で、±0. DDDDDD

(000000-999999) 有効数字10進6桁×±NN (+63-----64) 乗の範囲です。

$$r\ Z\ Z = r\ X\ X \quad \text{コピー}$$

$$r\ Z\ Z = r\ X\ X + r\ Y\ Y$$

$$r\ Z\ Z = r\ X\ X - r\ Y\ Y$$

$$r\ Z\ Z = r\ X\ X * r\ Y\ Y$$

$$r\ Z\ Z = r\ X\ X / r\ Y\ Y$$

直接、実数レジスターにデータをセットできないので、C文字レジスター、又はI正整数レジスターから変換してください。I正整数レジスターからは、負数をセットできないので、例えば、-2.0をセットしたいときは

-----i00=2 r00=i00 r00=r99-r00 -----のように変換してください。

(r99はコントローラーの電源ONで0.0にデフォルトセットしています。コントローラー仕様参照の事)

### ③ 文字レジスターセット

C文字レジスターを先頭として、文字列セットします。下記の例の如く2形式有りますが、1命令中、**混合しての使用は出来ません。**

e x、

$$c\ X\ X = "ABC" \quad c\ X\ X = "A" \quad c\ X\ X + 1 = "B" \quad c\ X\ X + 2 = "C" \quad \text{と}$$

なります。アスキー文字セットです。

$$c\ X\ X = h'4\ 1\ 4\ 2\ 4\ 3 \quad c\ X\ X = h'4\ 1 \quad c\ X\ X + 1 = h'4\ 2$$

$$c\ X\ X = h'4\ 3 \quad \text{と16進コードセットです。}$$

### ④ 各レジスター間の変換

I R間 0～9999までの数

$$r\ Z\ Z = i\ X\ X \quad I\ X\ X \text{の整数を} R\ Z\ Z \text{の実数に変換}$$

$$i\ Z\ Z = r\ X\ X \quad R\ X\ X \text{の実数を} I\ Z\ Z \text{の正整数に変換}$$

I C間 0～65535までの数

$$i\ Z\ Z = c\ X\ X\#n \quad n \text{は} 1 \sim 5 \quad C\ X\ X \text{を先頭に} n \text{文字分} I\ Z\ Z \text{に2進数としてセットしまう。}$$

$$c\ Z\ Z = i\ X\ X\#z\ n \quad n \text{は} 1 \sim 5 \quad I\ X\ X \text{の2進数をゼロサプレス無しの10進数として、} C\ Z\ Z \text{を先頭に} n \text{文字分セットします。}$$

$$c\ Z\ Z = i\ X\ X\#s\ n \quad n \text{は} 1 \sim 5 \quad I\ X\ X \text{の2進数をゼロサプレス有りの10進数として、} C\ Z\ Z \text{を先頭に} n \text{文字分セットします。}$$

I C間 0～FFFFまでの16進数

$$c\ Z\ Z = i\ X\ X\#h \quad I\ X\ X \text{の16進数をゼロサプレス}$$

無しの16進数として、CZZを先頭に  
4文字分セットします。

i Z Z = c X X # h

CXXを先頭として、16進数4文字をI ZZ  
にセットします。

R C間 **実数 (浮動小数点形式)**

r Z Z = c X X # n

nは1~6 CXXの先頭からn文字をR ZZ  
に変換します。

+-. を含んでいてもかまいません。

e x、

c10=" -32.5 r20=c10#5 ----- ; R20に-32.5が入る

c Z Z = r X X # z m . n

mは1~6整数桁、nは1~6小数桁

zはゼロの意味でゼロサプレス無し

但し  $1 \leq m + n \leq 6$

RXXが負ならCZZ先頭はマイナスをセット  
し、CZZ+1からフォーマットします。

但し、小数点以外はm+n桁でカットします。

e x、

c10=" -32.5 r20=c10#5 ----- ; R20に-32.5が入る

c20=r20# z 6.0 ; C20から -00003 +32.5なら 000032

c20=r20# z 5.1 ; C20から -00032. +32.5なら 00032.5

ゼロサプレス無しで、+-有る場合は、桁をそろえるため、下記の例が  
良いと思います。

e x、

c10=" -32.5 r20=c10#5 ----- ; R20に-32.5が入る

I f\_r20<r99,Minus c20=r20# z 5.1 ----- ; R20が負なら、Minusへ

; +32.5正なら C20から 00032.5

Minus : c20=r20# z 4.2 ----- ; C20から-0032.5

c Z Z = r X X # s m . n

sはスペースの意味でゼロサプレス付

但し  $1 \leq m + n \leq 6$

RXXが負でCZZ先頭がスペースならマイナ  
スをセットするだけですが、スペースでないな  
ら、マイナスをセット後、CZZ+1からフォ

フォーマットします。但し、小数点以外はm+n桁でカットします。

ex、

c10=" -32.5 r20=c10#5 ; R20 に-32.5 が入る

c20=r20# s 6.0 ; C20 から -。。。 32 +32.5 なら 。。。 32

c20=r20# s 5.1 ; C20 から -。。 32.5 +32.5 なら 。。。 32.5

c20=r20# s 3.1 ; C20 から -32.5 +32.5 なら 。 32.5

c20=r20# s 2.2 ; C20 から -32.5 +32.5 なら 32.50

c20=r20# s 4.2 ; C20 から -。 32.50 +32.5 なら 。。 32.50

。はスペースです。桁をそろえるなら、3番目と最後の例が最良だと思います。

### 3.9 特殊関数

① 平方根

$$r Z Z = S q r t ( r X X ) \quad r X X > 0. 0$$

② AのX乗

$$r Z Z = ( r X X ) ** ( r Y Y ) \quad r X X > 0. 0$$

③ S I N

$$r Z Z = S i n ( r X X ) \quad r X X \text{はラジアン}$$

$2\pi$ で周期であるから、入力値を極力  
1周期以内で使用すれば精度も良い・

④ C O S

$$r Z Z = C o s ( r X X ) \quad r X X \text{はラジアン}$$

$2\pi$ で周期であるから、入力値を極力  
1周期以内で使用すれば精度も良い・

⑤ A R C T A N

$$r Z Z = A t n ( r X X ) \quad r X X \text{はラジアン}$$

$2\pi$ 周期であるから、入力値を極力  
1周期以内で使用すれば精度も良い・

⑥ 自然対数

$$r Z Z = L n ( r X X ) \quad r X X > 0. 0$$

⑦ 常用対数

$$r Z Z = L o g ( r X X ) \quad r X X > 0. 0$$

特殊関数は近似式で処理するため入力領域によっては精度が均一でない。また関数によっては精度の低いものも有ります。出来るだけ答えの有効数字4桁は正しくするようにしたつもりですが、事前に単独で入力領域と出力精度を確認したほうが良いと思います。又、続けて使用する場合の累積誤差にも注意してください。続けて使用する場合、処理時間がかかるため、信号処理のスピードを優先するならば、タスクを分けて特殊関数と特殊関数の間に、S（タスクのスイッチング）命令をはさんで使用してください。その場合、同一タイミングでの他タスクとのR実数レジスタの重複は避けてください。

### 3.10 メモリー操作

① S f c \_ c Z Z , n , c X X

文字レジスタのシフト

C Z Z から  $n$  ( 1 - 1 5 ) 文字分 C Z Z +  $n$  - 1 まで、C Z Z + 1 からにシフトして、C Z Z + 1 から C Z Z +  $n$  までストアすると同じに、元もとの C X X を C Z Z にセットします。C X X が C Z Z +  $n$  なら、回転する形になります。

② S f i \_ i Z Z , n , i X X

正整数レジスタのシフト

I Z Z から  $n$  ( 1 - 1 5 ) レジスタ分 I Z Z +  $n$  - 1 まで、I Z Z + 1 からにシフトして、I Z Z + 1 から I Z Z +  $n$  までストアすると同じに、元もとの I X X を I Z Z にセットします。I X X が I Z Z +  $n$  なら、回転する形になります。

e x、

```
i20=0 i21=0 i22=0
```

```
Loop : Inw_Isig1 Pi_0,i10&h'FF Sfi_i20,3,i10 Po_3,i23&h'FF
```

```
Ifw_Isig1 Go_Loop
```

I sig1 の立ち上がりで 0 ポートから入力した、データを 4 ピッチずらして 3 ポートへ出力します。

③ M v \_ i Y Y , D a t a ( i X X )

I X X の内容をインデックスとして、D a t a メモリーから I Y Y へセットします。

I X X の内容は、0 - 9 9 9 9 までとし、それ以外は何もしません。

④ M v \_ D a t a ( i X X ) , i Y Y

I X X の内容をインデックスとして、I Y Y から D a t a メモリーへセットします。

I X X の内容は、0 - 9 9 9 9 までとし、それ以外は何もしません。

MEPS コマンドジェネレーター及び、コントローラーで、D 0 0 - D 9 9 は、I X X の 0 - 9 9 に対応する D A T A メモリーですが、コントローラーのサービスメニューで `C o n t r o l - F l a s h W r i t e - S e t D a t a` の書きこみで、スタティック R A M が消えた時の、バックアップデフォルト値として登録出来ます。D 0 0 - D 9 9 は、プログラムの切り替え用等のリードオンリーメモリーとすれば便利です。

I X X の 1 0 0 - 9 9 9 9 に対応する D A T A メモリーはデータロギング用です。



⑤ Db\_\_HH, HH, -----

16進で2桁ずつバイトテーブルを設定。文字とはかぎりません。機械語でもかまいません。この場合、Ca11命令でテーブルのラベルをコールして、機械語の最後にリターンの機械語をセットして、サブルーチンの形が良いと思います。他のコンパイラーで、機械語を作成するのもいいですが、暗黙の安全スタック領域を1タスク64バイト（実際は80バイト）と考えてください。たくさん必要とすれば、暴走の原因になります。つまりあまり多くのサブルーチンのネスティングをしてはいけません。又、メモリとして絶対アドレスで確保できる領域は、BBFFHまで、プログラムの最終から使用できます。最終は、コンパイル時の最終アドレスで確認出来ます。BC00Hを越すと暴走します。

これらは、MEPSと他機能とのソフトウェアインターフェースとしての可能性を残すものとしています。

テーブルには、先頭にラベルをつけてください。

⑥ Dw\_HHHH, -----

16進で4桁ずつワードテーブルを設定。Hi-Loの順にセットしてください。テーブルには、先頭にラベルをつけてください。

⑦ Getb\_@@@@@@, cYY, nn, iXX

横nn(1-10)バイト毎の、ラベル@@@@@@のバイトテーブルのiXXの内容値をインデックスとした横1列nnバイトをcYYからからnnバイト分セットします。

⑧ Getw\_@@@@@@, iYY, nn, iXX

横nn(1-10)ワード毎の、ラベル@@@@@@のワードテーブルのiXXの内容値をインデックスとした横1列nnワードをiYYからからnnワード分セットします。

### 3.11 キー入力

①  $Ky\_iXX$

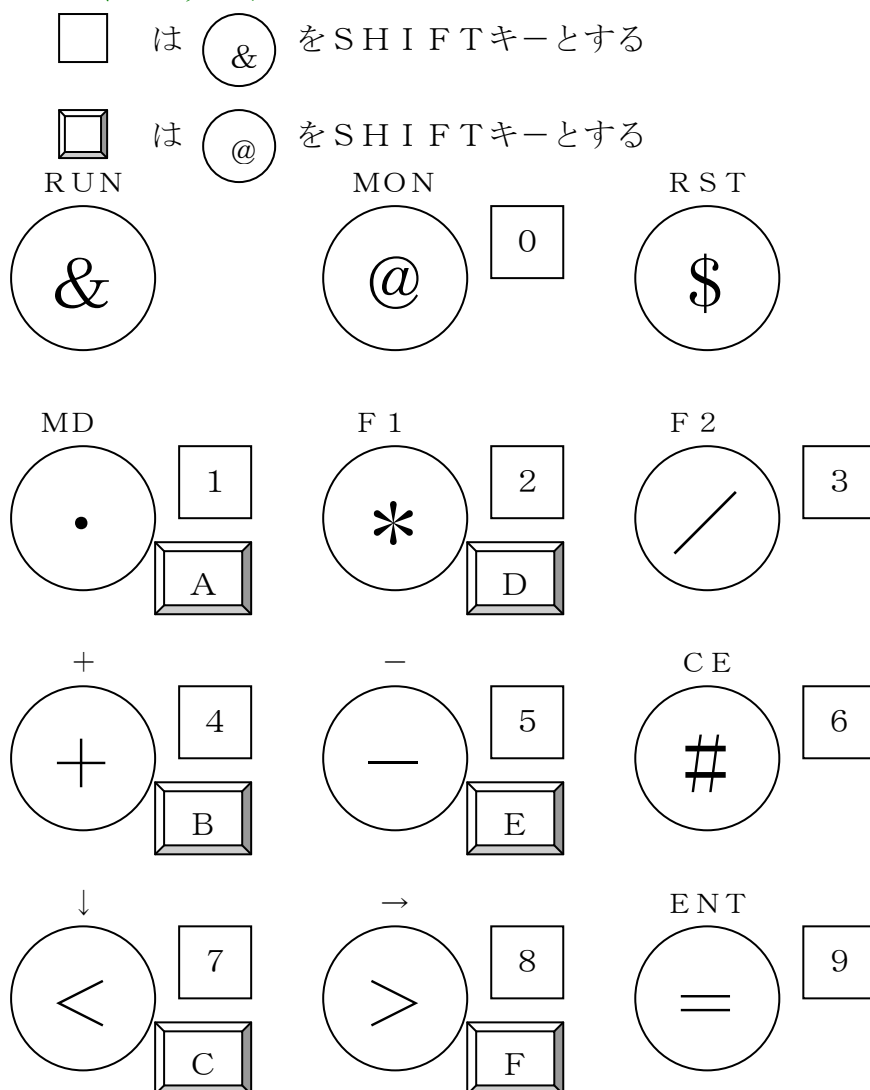
キー入力を即実行し、 $iXX$ に12ビットセットします。

|     |     |    |     |    |    |    |     |     |    |    |     |
|-----|-----|----|-----|----|----|----|-----|-----|----|----|-----|
| b11 | b10 | b9 | b8  | b7 | b6 | b5 | b4  | b3  | b2 | b1 | b0  |
| ↓   | +   | MD | RUN | →  | -  | F1 | MON | ENT | CE | F2 | RST |

②  $Kyw\_cXX$

キー入力をウェイトし、押されれば、その変換キーコードを $cXX$ にセットします。

<Me p sコントローラのキー>



③  $Chk?_cXX$ , "0", "9", "----", h'41, h'41

$cXX$ の内容を、ペアづつコードチェックし、 $min \leq cXX \leq max$ なら False (0)です。有効キーのチェックに使用します。

### 3.12 LCD表示

① DCLR

画面をクリアします。

② Disp\_\_i i, j, "ABC- - -

画面 i i (0-15) j (0-5) 座標から、ABC - - - の文字列を出力します。

③ Dspc\_\_i i, j, n n, c Z Z

C Z Z のレジスタから n n 文字分 (i i + n n < 16) C Z Z + n n - 1 まで 画面 i i (0-15) j (0-5) 座標から出力します。

④ Drvs\_\_i i, j, n n

画面 i i (0-15) j (0-5) 座標から横列 n n 文字分 (i i + n n < 16) 反転文字にします

⑤ Numw\_\_i i, j, n n, c Z Z, d

10進入力

画面 i i (0-15) j (0-5) 座標から、数値 0-9 キーで入力表示しながら、→キーで桁送りしながら、n n 文字分 (i i + n n < 16) 入力して、ENTキー押下 (False (0))、CE(Clear Entry)キー押下 (True (1)) のどちらかで、次命令の続行となります。

CE キーの場合は、C Z Z から、n n 文字分元のデータになります。

ENT キーの場合は、C Z Z から n n 文字分、数値セットします。数値の入力方法に 2 とおりあり、キーマット図の如く、シフトキーを併用して直接数値を入力する方法と、+ キーで 0 1 2 3 4 5 6 7 8 9 . をローテーションさせて設定する方法とが有ります。

パラメータ d は 10 進入力の意味で実数入力も出来るように - と . はローテーションの中にあります。

C Z Z から C Z Z + n n - 1 をこの命令以前にクリアすべきです。

⑥ Numw\_\_i i, j, n n, c Z Z, h

16進入力

画面 i i (0-15) j (0-5) 座標から、数値 0-9、A-F キーで入力表示しながら、→キーで桁送りしながら、n n 文字分 (i i + n n < 16) 入力して、ENT キー押下 (False (0))、CE(Clear Entry)キー押下 (True (1)) のどちらかで、次命

令の続行となります。

CE キーの場合は、C Z Z から、n n 文字分元のデータになります。

ENT キーの場合は、C Z Z から n n 文字分、数値セットします。数値の入力方法に 2 とおりあり、キーマツ図の如く、シフトキーを併用して直接数値を入力する方法と、+ キーで 0 1 2 3 4 5 6 7 8 9 A B C D E F をローテーションさせて設定する方法とが有ります。

パラメーター h は 1 6 進入力の意味です。

C Z Z から C Z Z + n n - 1 をこの命令以前にクリアすべきです。

⑦ R v s s e t \_ i Z Z , i i , j , n n , i i 1 , j 1 , n n 1 -----

I Z Z を、以下に続く i i ( 0 - 1 5 ) , j ( 0 - 5 ) の位置から n n 文字分 ( i i + n n < 1 6 ) の 繰り返しテーブルのポインターとして、反転をループ状に行う初期セット命令です。

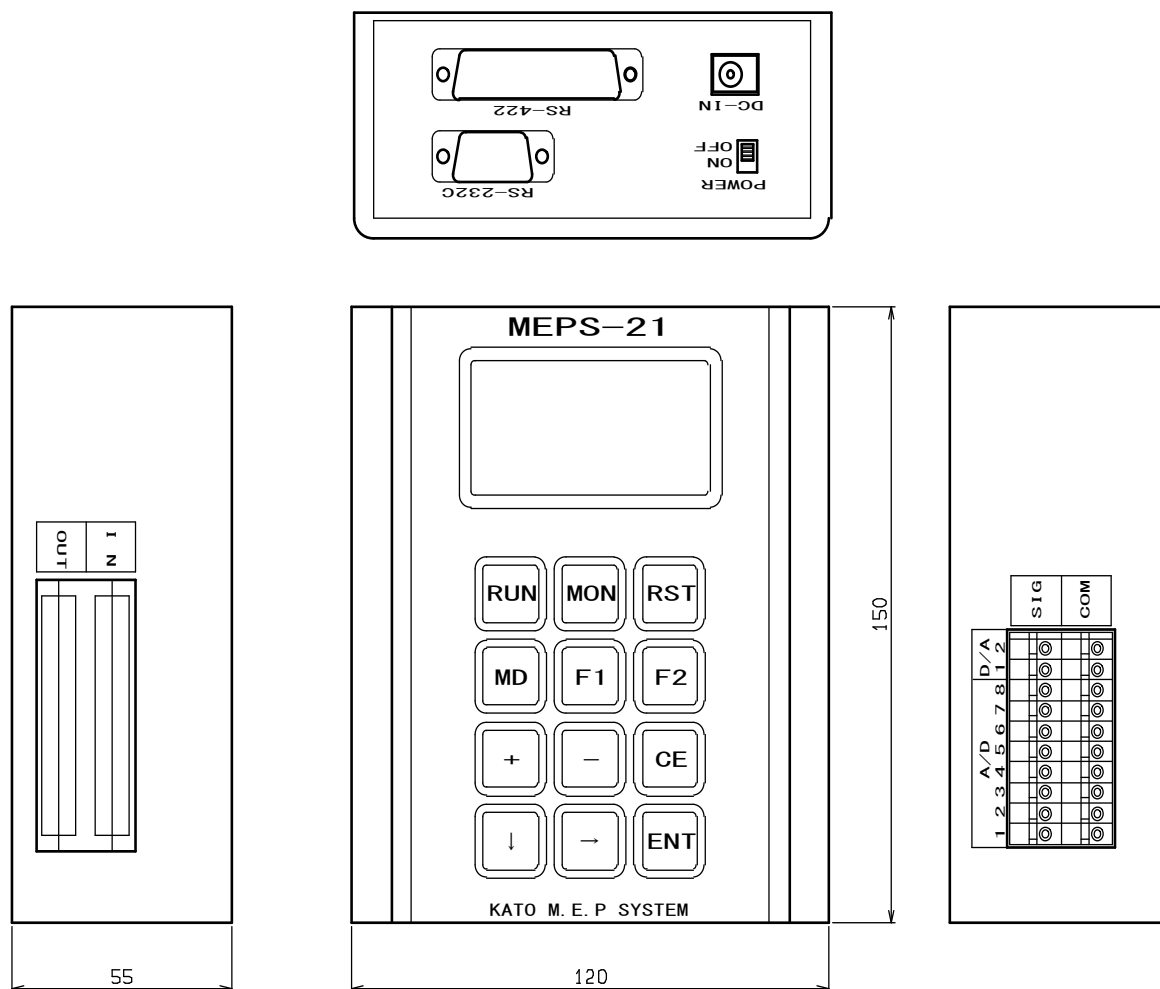
⑧ R v s i n c \_ i Z Z , i i , j , n n , i i 1 , j 1 , n n 1 -----

R v s s e t と同じパラメーターにして下さい。

この命令を実行するたびに、反転をループ状に行います。

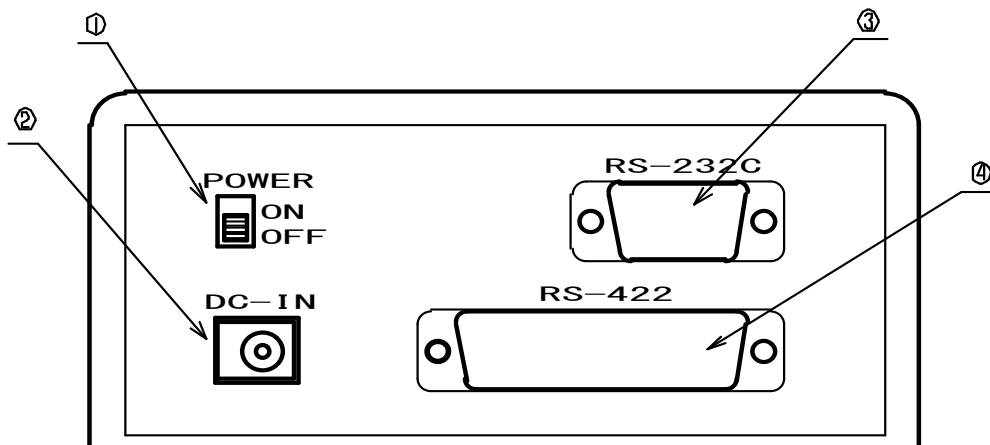
## 4. MEPS コントローラー

### 4.1 外観図



<外観図>

## 4.2 コネクター



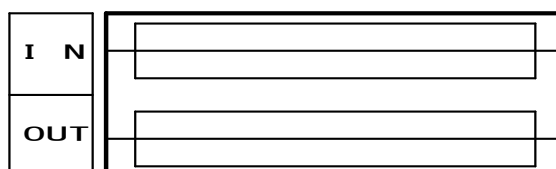
① 電源スイッチ

② ACアダプター入力

③ RS-232C

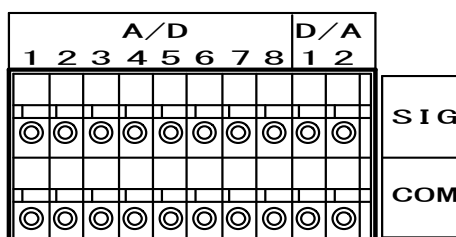
④ RS-422 (オプション)

### デジタル入出力コネクタ



### A/D (8チャンネル)、D/A (2チャンネル) 入出力端子

(オプション)



## 4.3 コネクターピン

### 4.3.1 通信

#### RS-232C コネクタ (D-sub 9ピン ピンプラグ)

|   |     |   |     |
|---|-----|---|-----|
| 1 | CD  | 2 | RXD |
| 3 | TXD | 4 | DTR |
| 5 | GND | 6 | DSR |
| 7 | RTS | 8 | CTS |
| 9 | RI  |   |     |

#### RS-422 コネクタ (D-sub 25ピン ピンプラグ)

##### オプション

|    |      |    |      |
|----|------|----|------|
| 1  | NC   | 2  | TX+  |
| 3  | RX+  | 4  | RTS+ |
| 5  | CTS+ | 6  | RTS- |
| 7  | GND  | 8  | TX-  |
| 9  | NC   | 10 | NC   |
| 11 | NC   | 12 | NC   |
| 13 | NC   | 14 | NC   |
| 15 | NC   | 16 | NC   |
| 17 | NC   | 18 | NC   |
| 19 | NC   | 20 | RX-  |
| 21 | NC   | 22 | CTS- |
| 23 | NC   | 24 | NC   |
| 25 | NC   |    |      |

#### 4.3.2 デジタル入出力

入力コネクタ (適応コネクタ XG4M-4030(オムロン)又は相当品)

|    |     |    |     |
|----|-----|----|-----|
| 1  | +V0 | 2  | D00 |
| 3  | D01 | 4  | D02 |
| 5  | D03 | 6  | D04 |
| 7  | D05 | 8  | D06 |
| 9  | D07 | 10 | NC  |
| 11 | +V1 | 12 | D10 |
| 13 | D11 | 14 | D12 |
| 15 | D13 | 16 | D14 |
| 17 | D15 | 18 | D16 |
| 19 | D17 | 20 | NC  |
| 21 | +V2 | 22 | D20 |
| 23 | D21 | 24 | D22 |
| 25 | D23 | 26 | D24 |
| 27 | D25 | 28 | D26 |
| 29 | D27 | 30 | NC  |
| 31 | +V3 | 32 | D30 |
| 33 | D31 | 34 | D32 |
| 35 | D33 | 36 | D34 |
| 37 | D35 | 38 | D36 |
| 39 | D37 | 40 | NC  |

+V0~+V3 は外部電源入力

入力点数：32点 入力抵抗：1.2V/2.4V 入力4.7K 5V/1.2V

入力2.2K 消費電流：80mA (無負荷動作時)



出力コネクタ (適応コネクタ XG4M-4030(オムロン)又は相当品)

|    |     |    |      |
|----|-----|----|------|
| 1  | +V0 | 2  | D00  |
| 3  | D01 | 4  | D02  |
| 5  | D03 | 6  | D04  |
| 7  | D05 | 8  | D06  |
| 9  | D07 | 10 | GND0 |
| 11 | +V1 | 12 | D10  |
| 13 | D11 | 14 | D12  |
| 15 | D13 | 16 | D14  |
| 17 | D15 | 18 | D16  |
| 19 | D17 | 20 | GND1 |
| 21 | +V2 | 22 | D20  |
| 23 | D21 | 24 | D22  |
| 25 | D23 | 26 | D24  |
| 27 | D25 | 28 | D26  |
| 29 | D27 | 30 | GND2 |
| 31 | +V3 | 32 | D30  |
| 33 | D31 | 34 | D32  |
| 35 | D33 | 36 | D34  |
| 37 | D35 | 38 | D36  |
| 39 | D37 | 40 | GND3 |

+V0~+V3、GND0~GND3 は外部電源入力

出力点数：32点 出力電圧：30V (最大) 出力電流：500mA (最大)

消費電流：80mA (無負荷動作時)

#### 4.3.3 A/D、D/Aコンバータ

端子側面上段

|     |     |     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  |
| AI0 | AI1 | AI2 | AI3 | AI4 | AI5 | AI6 | AI7 | AO0 | AO1 |

端子側面下段

1———8 GND1 9-10 GND2

A/D 12bit × 8ch 0-4.095V

D/A 12bit × 2ch 0-5V

## 4.4 M e p s コントローラーの仕様

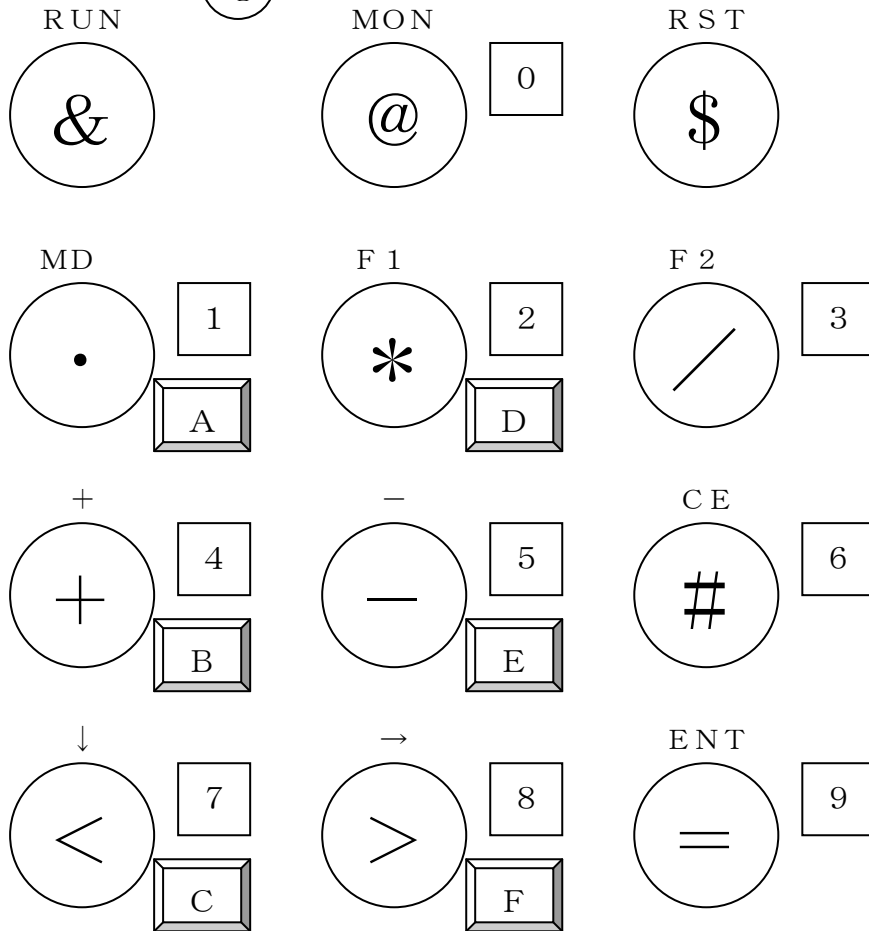
- 1、 I/O フォトカプラー付き 32点/32点
- 2、 A/D 12bit × 8ch 0-4.095V (オプション)
- 3、 D/A 12bit × 2ch 0-5V (オプション)
- 4、 カレンダー時計 (年月日時分秒)
- 5、 通信 RS232C 1ch RS422 1ch (オプション)
- 6、 128 × 64 LCD (横16桁 × 縦6行 表示器)
- 7、 12個のキー
- 8、 256点の内部接点信号
- 9、 100個の整数16bitレジスター
- 10、 10000個の16bit定数及びデータ格納レジスター (バッテリー記憶)
- 11、 100個の16bitタイマー定数レジスター (バッテリー記憶)
- 12、 100個の実数32bitレジスター
- 13、 100個の文字8bitレジスター
- 14、 10個の外部タイマー (基準10ms × 最大65535)
- 15、 各タスク1個の固有タイマー (基準10ms × 最大65535)
- 16、 最大タスク数120
- 17、 アプリケーションプログラム最大1000行 (1行平均5命令)
- 18、 システムサポート
  - 1)、 内部接点V255: 1ms duty clock サービス
  - 2)、 内部接点V254: 10ms duty clock サービス
  - 3)、 内部接点V253: 1sec duty clock サービス
  - 4)、 整数レジスターI99: msカウンター (0-999)

**I98: 0-65535 擬似一様乱数 (10ms 毎)**  
これは冗長性のあるロボットなどへの応用として有用  
もちろん正規分布等、関数式で変換できます。
- 5)、 電源ONのスタート時のみ、デフォルト値として下記の如くセット  
実数レジスター R99: 0.0                    R98: 1.0  
R97: 10.0                    R96: 100.0  
R95: 1000.0                    R94: 4096.0  
R93: 3.14159                    R92: 360.0  
R91: 273.0                    R90: 2.71828

19、Meps コントローラのキー

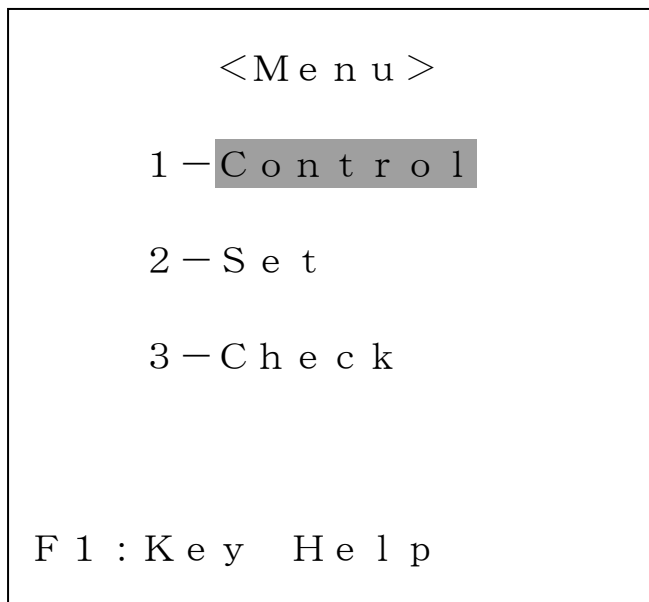
□ は ( & ) をSHIFTキーとする

▣ は ( @ ) をSHIFTキーとする



## 5. Meps Controller Help (MCH) 画面

### 5.1 メニュー画面



大別して1、コントロール

2、設定

3、チェック

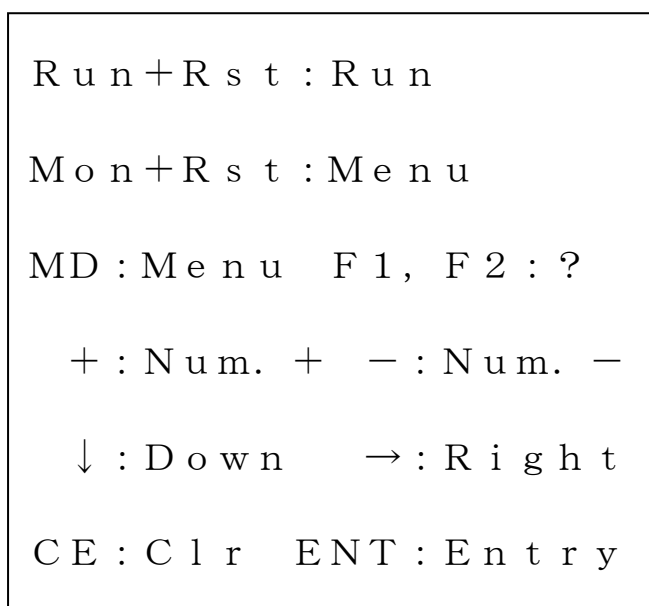
があり、↓キーの選択と

ENTキーにて、各画面へ

と推移します。

F1キーではキーヘルプ画面

### 5.2 キーヘルプ画面



1、RunキーとRstキー

又は、電源ON時にRun

キーで、ユーザープログラム

が開始します。但し、まだ

プログラムが入ってない時

は、Meps Controller Help (MC

H) が開始します。

2、MonキーとRstキー

又は、電源ON時にMon

キーで、MCHが開始します。

3、MDキーは各画面から、メニ

ュー画面へ推移します。

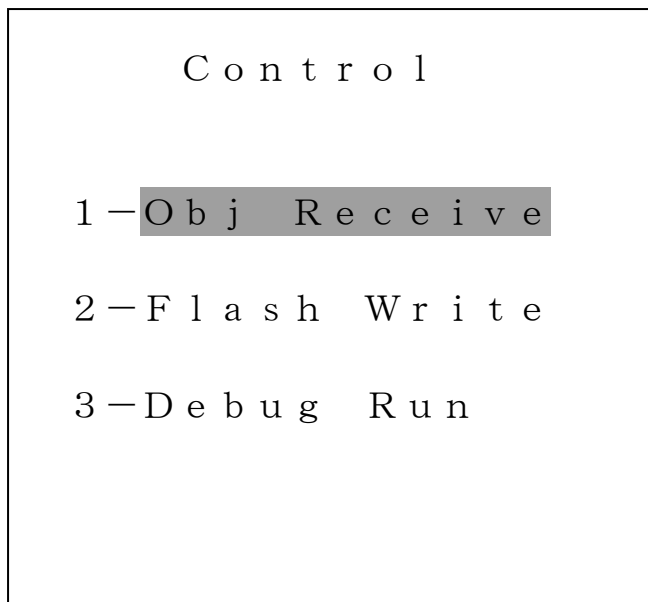
4、F1、F2キーは各画面にて

それぞれの意味づけをして

下さい

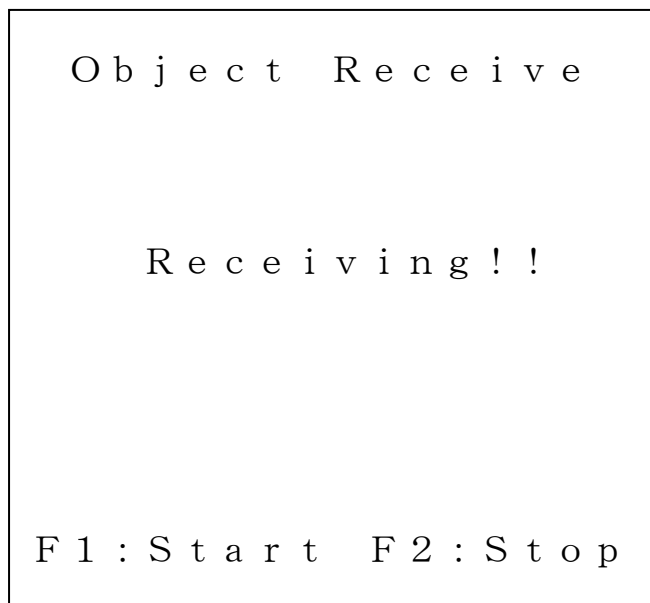
- 5、+、- キーは数値入力のUP/DOWNに使用します。
- 6、↓、→ キーは、項目の行、列、指定に使用します。
- 7、CEキーは、クリアで、元のデータに戻ります。
- 8、ENTキーは確定及び設定をします。

### 5.3 コントロール画面



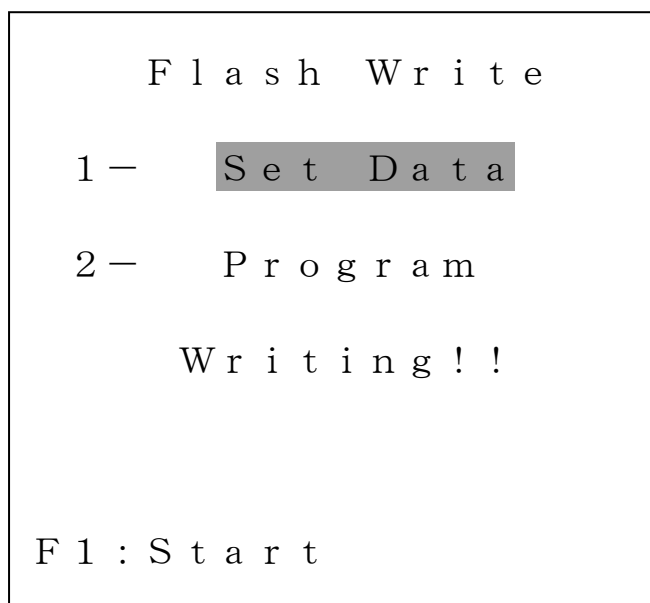
- 1、オブジェクトコード受信
- 2、フラッシュメモリー書きこみ
- 3、デバッグ開始  
↓キーの選択とENTキーにて、各画面へと推移します。

## 5.4 オブジェクトコード受信画面



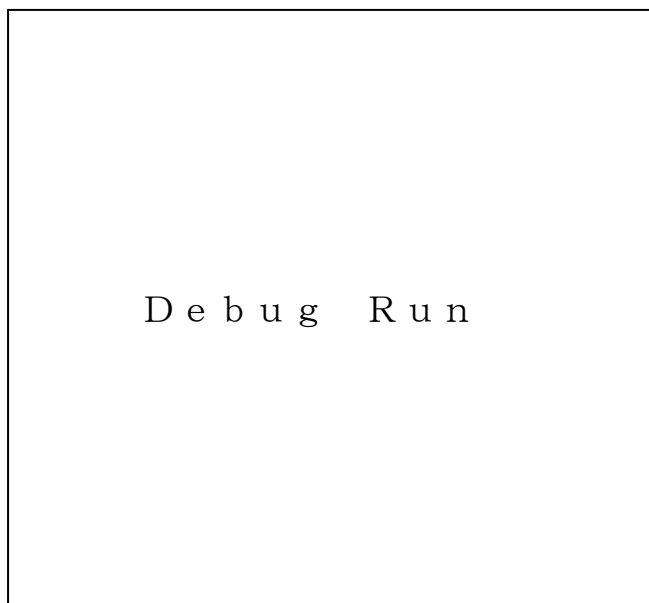
- 1、M e p s コマンドジェネレーターによって、PCからオブジェクトコードを受信します。
- 2、F 1 キーを押しますと、真ん中に R e c e i v i n g !! を表示します。PCからの転送を開始してください。終了、又はF 2 キーで中断すれば、R e c e i v e e n d を表示して、終了します。
- 3、MDキーでメニュー画面です。

## 5.5 フラッシュメモリー書き込み画面



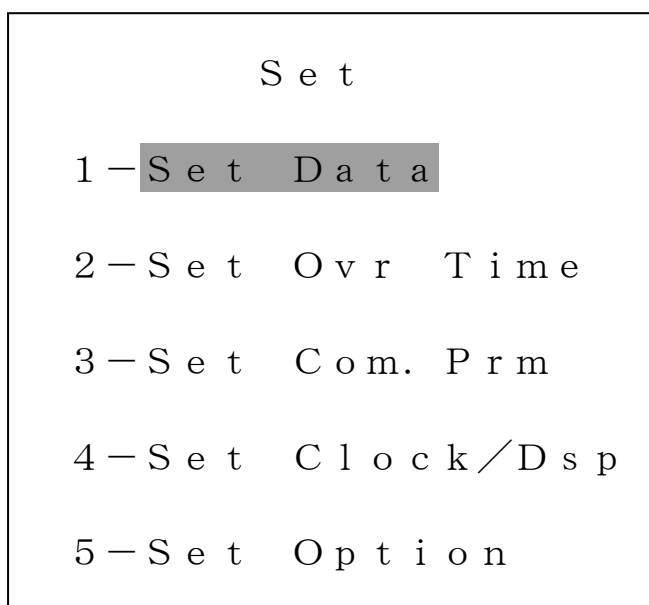
- 1、設定データ書き込みは、データメモリー00-99とタイマー定数No00-99の値、各種コントローラー条件を書き込みます。
- 2、プログラム書き込みは、PCからのオブジェクトコード受信後、書き込みます。
- 3、↓キーで項目指定し、F 1 キーで書き込み開始し、W r i t i n g !! を表示します。
- 4、MDキーでメニュー画面です。

## 5.6 デバッグ画面



- 1、オブジェクトコード受信後直後、この画面がひらけます。
- 2、この画面から、MDキーでもメニュー画面には戻りません。MCHの管理下ではありません。電源を入れなおしてください。
- 3、Debug Runの表示は、ユーザープログラムで表示するまでの表示です。

## 5.7 設定画面



- 1、D00-99のデータメモリの設定
  - 2、タイマー定数の設定
  - 3、通信パラメーターの設定
  - 4、カレンダークロックの設定
  - 5、AD/DAのch予約表示スピードの選択
- ↓キーの選択とENTキーにて、各画面へと推移します。

## 5.8 データ設定画面

|                           |               |
|---------------------------|---------------|
| S e t   D a t a           |               |
| No                        | D a t a       |
| D 0 0                     | :   0 0 0 0 0 |
| F 1 : N o   F 2 : D a t a |               |

- 1、F1キーでNo設定モード
- 2、F2キーでデータ定数設定モード
- 3、設定モードでは、数値が点滅→キーで、桁送り
- 4、ENTキーは確定、CEキーでは元のデータにもどります。
- 5、MDキーでメニュー画面です。

## 5.9 タイマー値設定画面

|                           |                     |
|---------------------------|---------------------|
| S e t   O v e r   T i m e |                     |
| No                        | D a t a             |
| 0 0                       | :   0 0 0 0 0 0 m s |
| F 1 : N o   F 2 : D a t a |                     |

- 1、F1キーでNo設定モード
- 2、F2キーでタイマー定数設定モード
- 3、設定モードでは、数値が点滅→キーで、桁送り
- 4、ENTキーは確定、CEキーでは元のデータにもどります。
- 5、MDキーでメニュー画面です。



## 5.10 通信パラメーター設定画面

### 0 c h

```
Com0 Parameter
Baud Rate : 38400
(48, 96, 192, 384)
Bit       : 8 (7, 8)
Stop      : 1 (1, 2)
Check     : N (N, E, O)
```

- 1、↓キーでボーレート、ビット長、ストップビットパリティチェック選択
- 2、→キーで各パラメーター選択、ENTキーで確定します。
- 3、MDキーでCOM1画面です。
- 4、CheckのNは無し、Eは偶数、Oは奇数パリティチェック

### 1 c h

```
Com1 Parameter
Baud Rate : 38400
(48, 96, 192, 384)
Bit       : 8 (7, 8)
Stop      : 1 (1, 2)
Check     : N (N, E, O)
```

- 1、↓キーでボーレート、ビット長、ストップビットパリティチェック選択
- 2、→キーで各パラメーター選択、ENTキーで確定します。
- 3、MDキーでその他の通信パラメーター画面です。
- 4、CheckのNは無し、Eは偶数、Oは奇数パリティチェック

## 5.11 その他の通信パラメーター設定画面

```
Com. Terminator
0No (No, Cr, Lf, Ex)
1No (No, Cr, Lf, Ex)
Com. Chr Ovr Time
0 : 20ms 1 : 20ms
F1 : Num Set Mode
```

- 1、↓キーで0ch最終コード、1ch最終コード、文字間オーバータイム0ch、1chを指定
- 2、→キーで各最終コード指定ENTキーで確定します。
- 3、文字間オーバータイムはF1キーで数値入力開始ENTキーで確定、CEキーで元のデータに戻ります。
- 4、MDキーでメニュー画面です。

## 5.12 カレンダークロック設定画面

```
      C l o c k

0 3 - 0 5 - 0 9

0 1 : 2 8 : 5 7

F 1 : W r i t e   M o d e
```

- 1、この画面で時刻は歩進しています。
- 2、F1キーを押すと歩進はとまり、設定モードになります。→キーで桁送りし、数値入力し、ENTキーでセットし最後の秒の設定で書き込みをします。途中でCEキー押せば、元の時刻に戻ります。
- 3、MDキーでメニュー画面です。

### 5.13 オプション設定画面

S e t   O p t i o n

A/D : N (N, 1, 2, 3  
4, 5, 6, 7, 8)

D/A : N (N, 1, 2)

- 1、↓キーで、A/Dか、D/A を指定します。
- 2、オプション数を→で指定します。
- 3、ENTキーで確定します。
- 4、MDキーで表示スピード画面です。

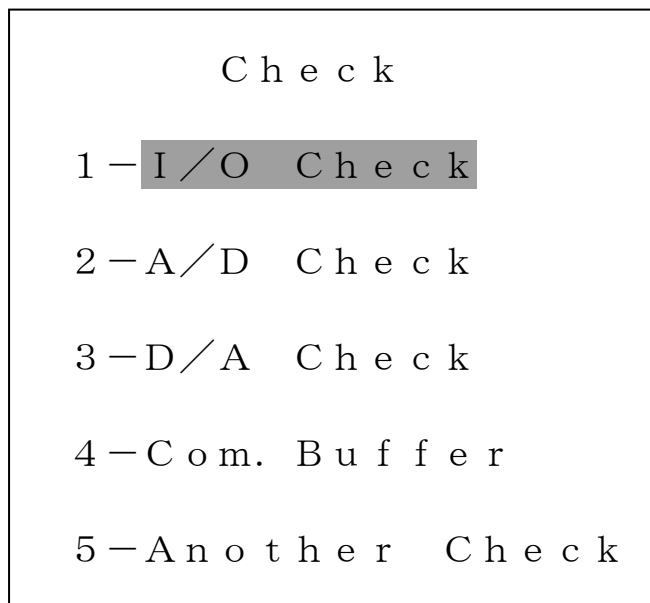
### 5.14 表示スピード選択画面

S e t   D i s p   S p e e d

S p d : L O (L O, H I)

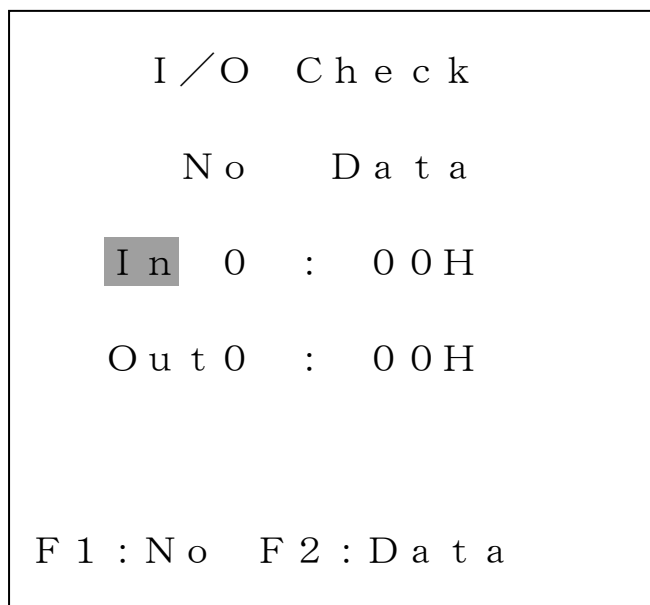
- 1、→キーでLO、HIのどれかを指定します。
- 2、ENTキーで確定します。
- 3、MDキーでメニュー画面です。
- 4、表示を早くすることは、良い。しかし、早いとI/Oのスピードが落ちますので、注意してください。

## 5.15 チェック画面



- 1、入出力チェック
  - 2、アナログ入力チェック
  - 3、アナログ出力チェック
  - 4、通信バッファ参照
  - 5、その他のチェック
- ↓キーの選択とENTキーにて、各画面へと推移します。

## 5.16 入出力チェック画面



- 1、↓キーでInかOutを選択
- 2、F1キーでNoをF2キーで入出力データを数値セットし、ENTキーで確定します。CEキーでは元のNoに戻ります。
- 3、MDキーでメニュー画面です。

### 5.17 アナログ入力チェック画面

| A/D C h e c k |         |     |   |
|---------------|---------|-----|---|
| 1 :           | 4 0 9 5 | 5 : | 0 |
| 2 :           | 0       | 6 : | 0 |
| 3 :           | 0       | 7 : | 0 |
| 4 :           | 0       | 8 : | 0 |

- 1、0から4095は0Vから4.095Vに対応
- 2、リアルタイムに入力と表示
- 3、未使用chはA/Dオプション設定
- 4、MDキーでメニュー画面です。

### 5.18 アナログ出力チェック画面

| D/A C h e c k |           |
|---------------|-----------|
| D a t a       |           |
| 1 c h         | : 4 0 9 5 |
| 2 c h         | : 0 0 0 0 |
| F 2 : D a t a |           |

- 1、↓キーで1chか2chを選択
- 2、F2キーで出力データを数値セットし、ENTキーで出力する。CEキーなら元のデータのままです。
- 3、0から4095は0Vから5Vに対応します。
- 4、MDキーでメニュー画面です。

## 5.19 通信バッファ参照画面

### 受信バッファ

```
Com0 Rcv Buffer
000000000000000000
000000000000000000
Com1 Rcv Buffer
000000000000000000
000000000000000000
```

- 1、受信16バイト分、0ch  
1ch 表示
- 2、MDキーで送信バッファ  
表示になります。

### 送信バッファ

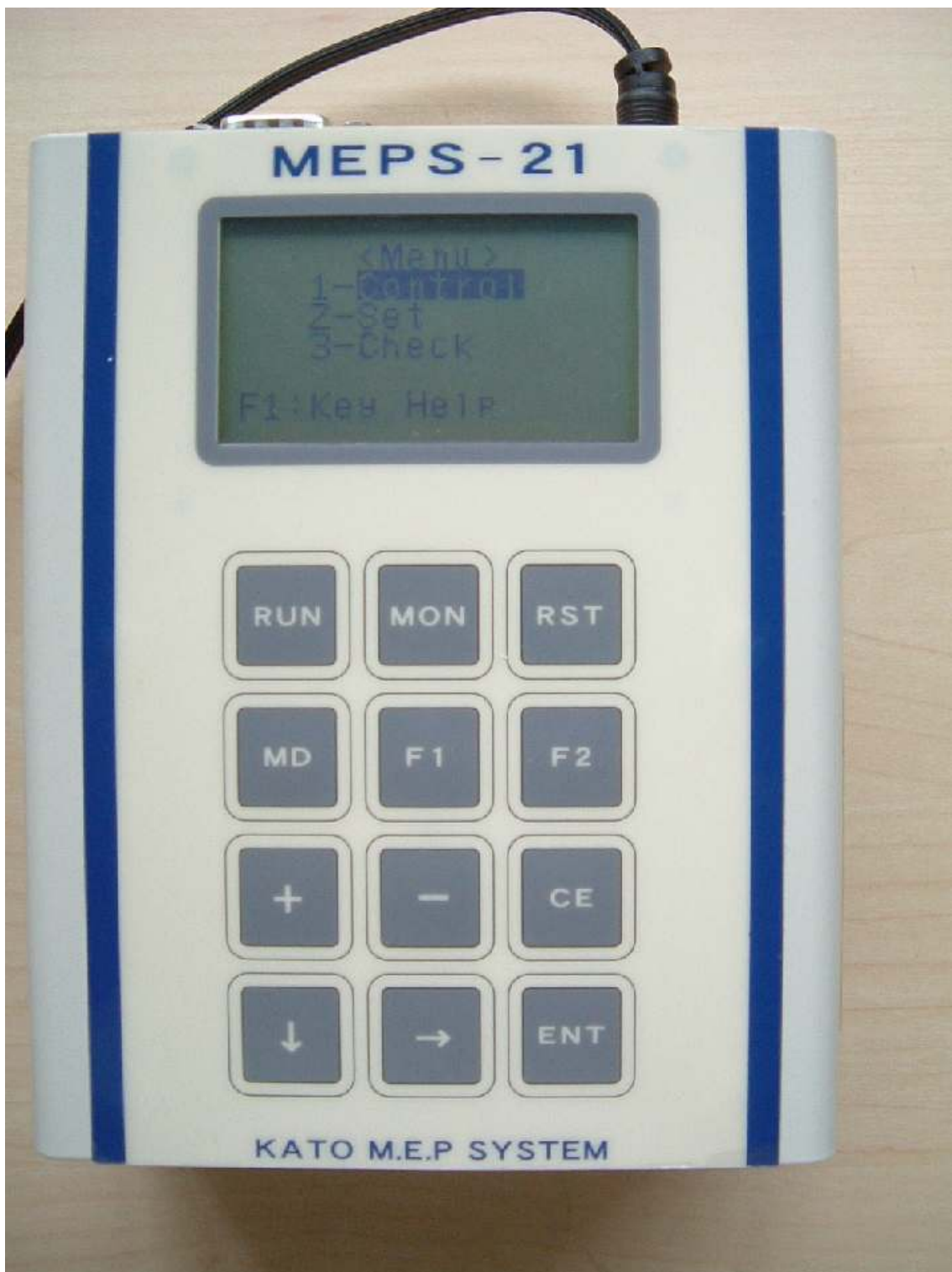
```
Com0 Snd Buffer
      F1:Only 8B Set
000000000000000000
Com1 Snd Buffer
      F2:Only 8B Set
000000000000000000
```

- 1、0ch、1chとも8バ  
イト分だけ送信チェッ  
ク  
できます。
- 2、F1キー、F2キーで0  
ch、1chそれぞれの  
バッファへの数値入力  
モードとなります
- 3、入力は、16進で、ENTキ  
ーで00 (Null) までの  
データを送信します。  
CEキーでは送信バッファ  
ーを00 (Null) クリア  
します。
- 4、MDキーでメニュー画面で  
す。

## 5.20 その他のチェック画面

```
A n o t h e r   C h e c k  
M a x   C y c l e   T   1 m s  
I 0 0 : 0 0 0 0 0 0 C 0 0 : 0 0 H  
R 0 0 : + 0 . 0 0 0 0 0 0 + 0 0  
V 0 0 : O F F  
F 1 : N o
```

- 1、Max Cycle Time は、ユーザープログラムの全タスク一周時間の最大値を表示します。
- 2、↓キー-IXX、CXX、RXX VHH を選択します。
- 3、F1キーでそれぞれの数値入力モードとなります。VHH の HH は16進です。
- 4、CXX は16進表示、VHH は ON/OFF 表示。あとは、10進表示です。ENT キーで、表示します。CE キーでは元の表示になります。
- 5、MD キーでメニュー画面です。



(有) 加藤エムイーピーシステム

TEL. FAX 06-6915-7639

E-MAIL kt\_sft@d3.dion.ne.jp