ロボット頭脳開発支援

ＭＥＰＳ２１

Ｖｅｒｓｉｏｎ　1.30

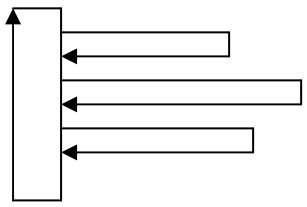
発行　2003.5.30　　第１版
発行　2008.8.30　　第２版
発行　2015.3.31　　第３版

（有）加藤エムイーピーシステム

目　　次

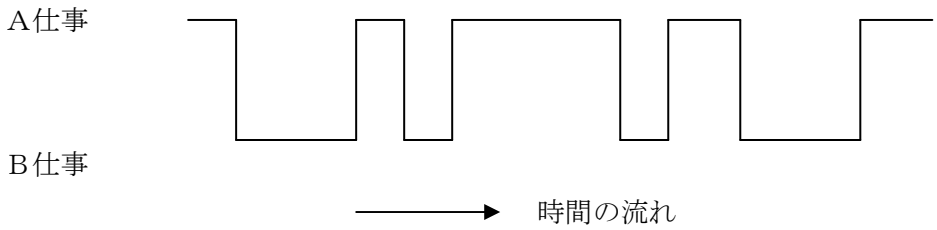
１．概　　要

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ーで 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 コンパイラーは 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増大して行きます。
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コンパイラーは、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、タスクス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。



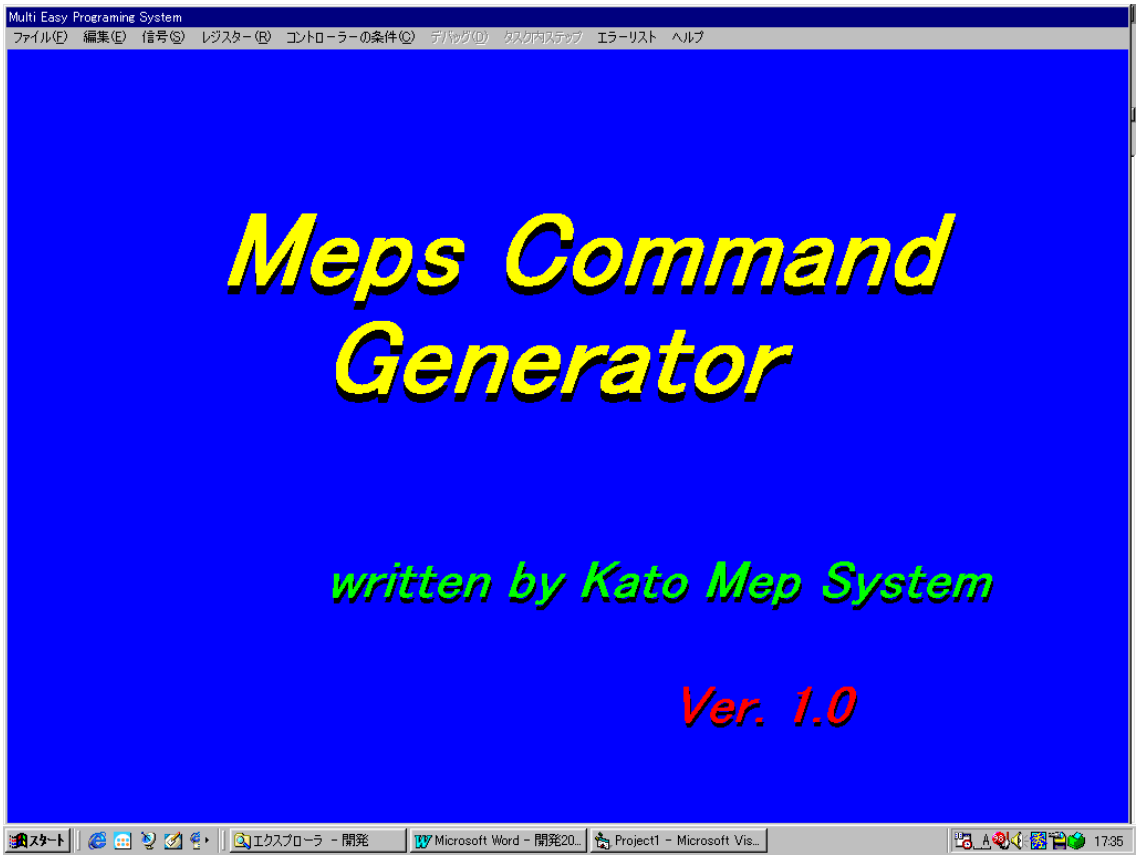
横方向一つをタスク（仕事の単位）として、マルチ（多重）タスクで処理をするということになります。

マルチタスクの概念の具体的な説明は、いわゆる、ながら族の話しです。
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



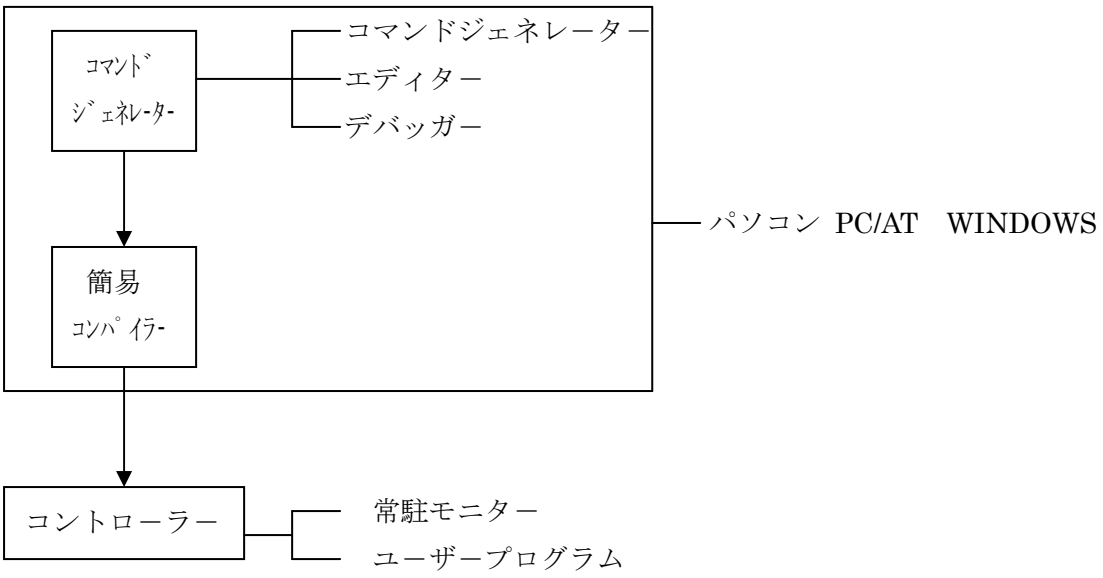
仕事をしている時間はA，B共、断続的ですが、何か待つたびに、積極的に仕事を切替（スイッチング）て、処理すれば効率的です。プログラム上待つ事が生じる点に切替（スイッチング）を用いた命令を書く事により、マルチタスクが可能となります。待つ事がない連続的即時処理の場合、強制的に切替を入れなければなりません。誤解のないようにしなければならない事として、本マルチタスクは自動的になされるのではなく自分で切替を設定して構築する事です。もし仮に切替を用いなければ最初のタスクを延々と実行するシングルタスクの状態になります。しかしこれが意図的であれば一向に差支えはありません。
Ｍｕｌｔｉ Ｅａｓｙ Ｐｒｏｇｒａｍｍｉｎｇ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

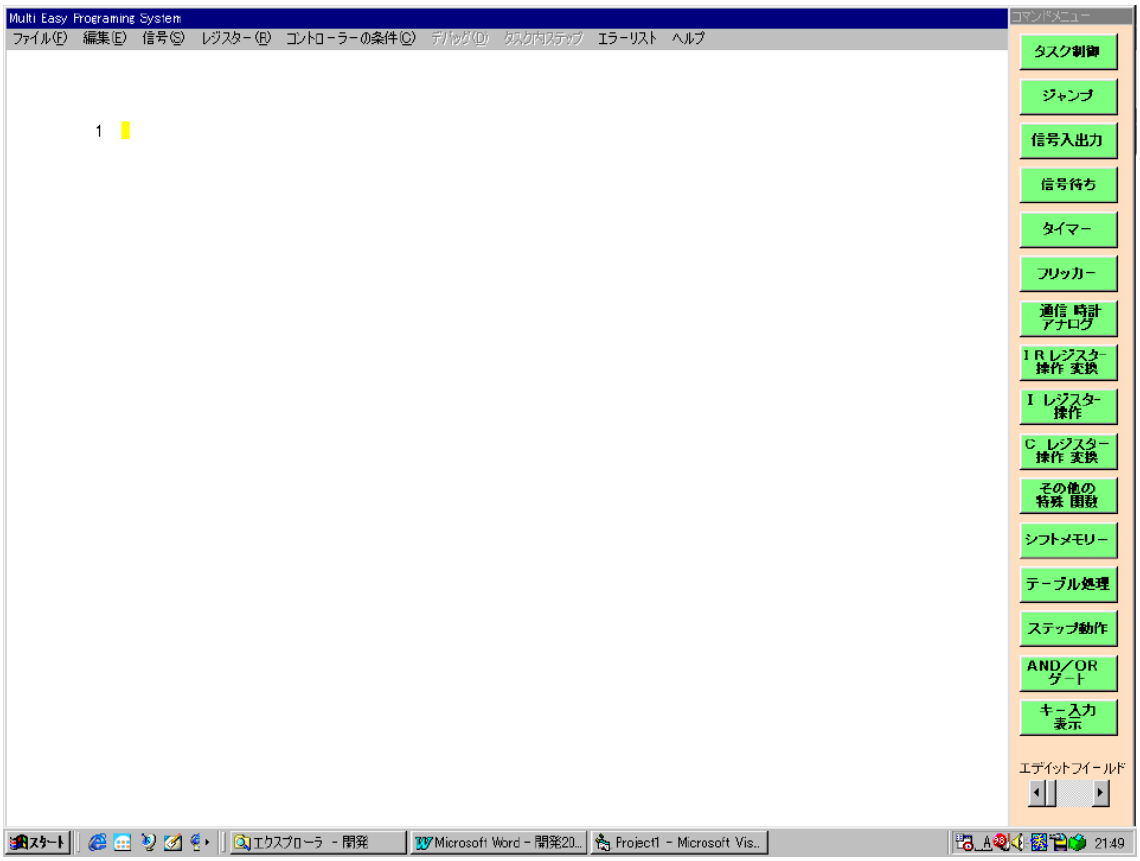


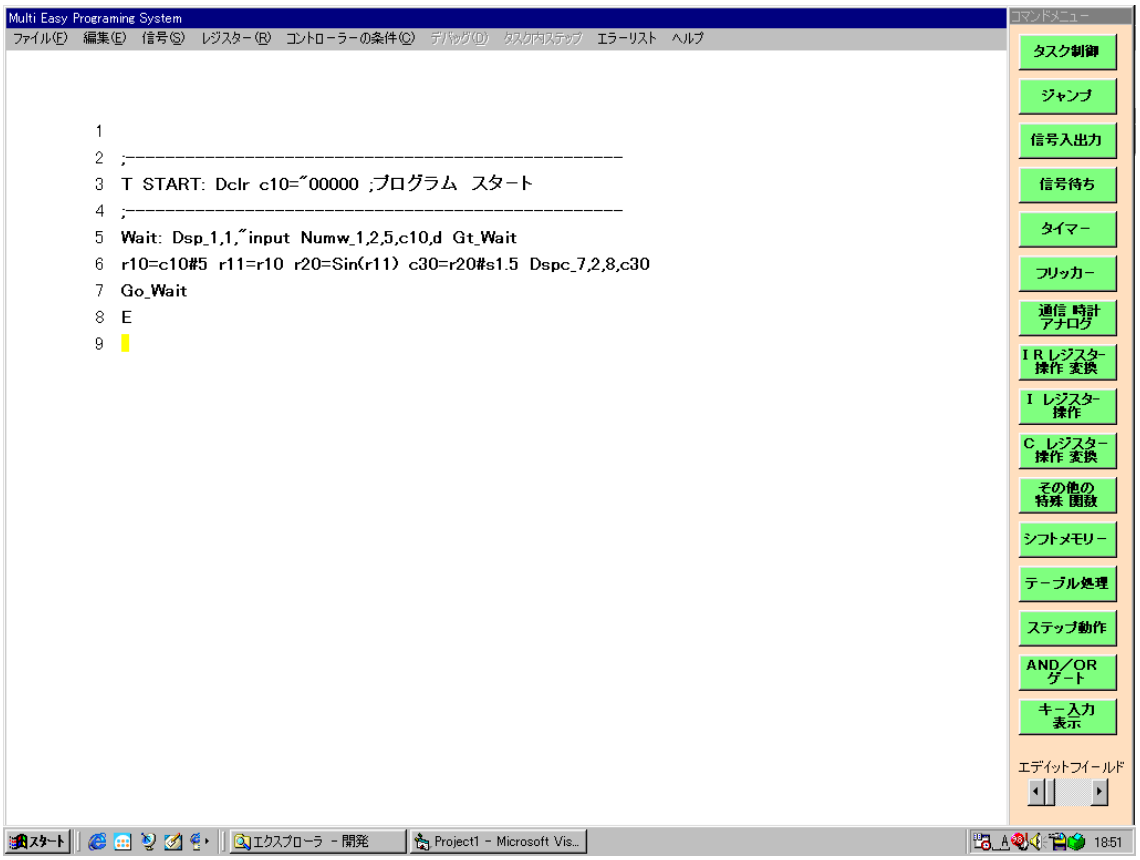
【図１】　ＭＥＰＳシステムの構成図

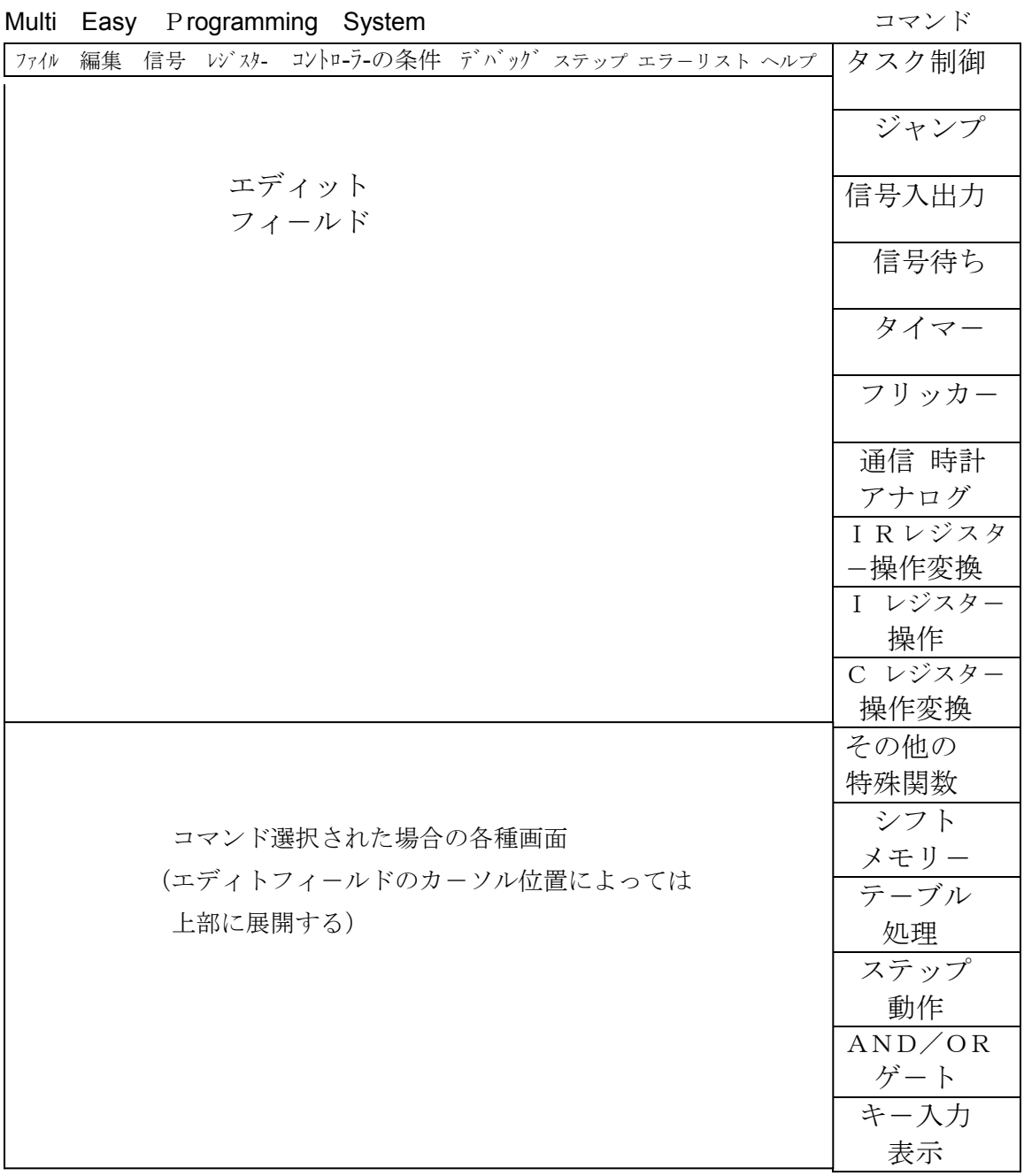
コントローラー仕様：縦１７０ｍｍ　×　横１２０ｍｍ　×　高さ６０ｍｍ
キー12 入力、LCD 6×16 桁、I／O 各 32 点フォトカプラー付き、カレンダー時計、RS232C １CH、ＣＰＵ　Ｚ８０　１０　ｍＨＺ、フラッシュメモリー５１２ＫＢ、スタティックＲＡＭ　５１２ＫＢ バッテリーバックアップ付き
オプションとしてA／D 8CH、D／A 2CH、RS422 又は RS232C　1 CH

もちろん簡易ロボット製作等の場合、ＣＰＵ基板（PIO24 ビット、RS232C　1 CH）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

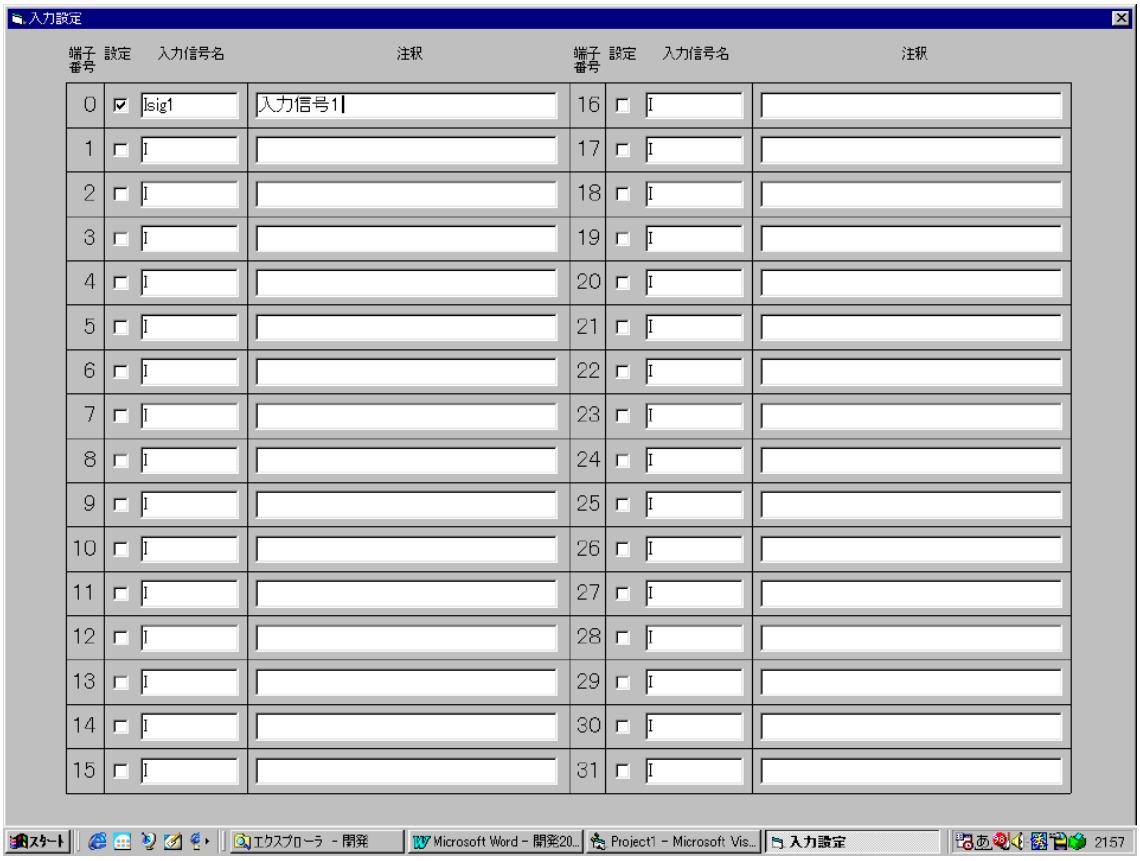






【図２】コマンドジェネレーターの、メイン画面

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ーＮｏにも、出来るだけ注釈を入れて下さい。



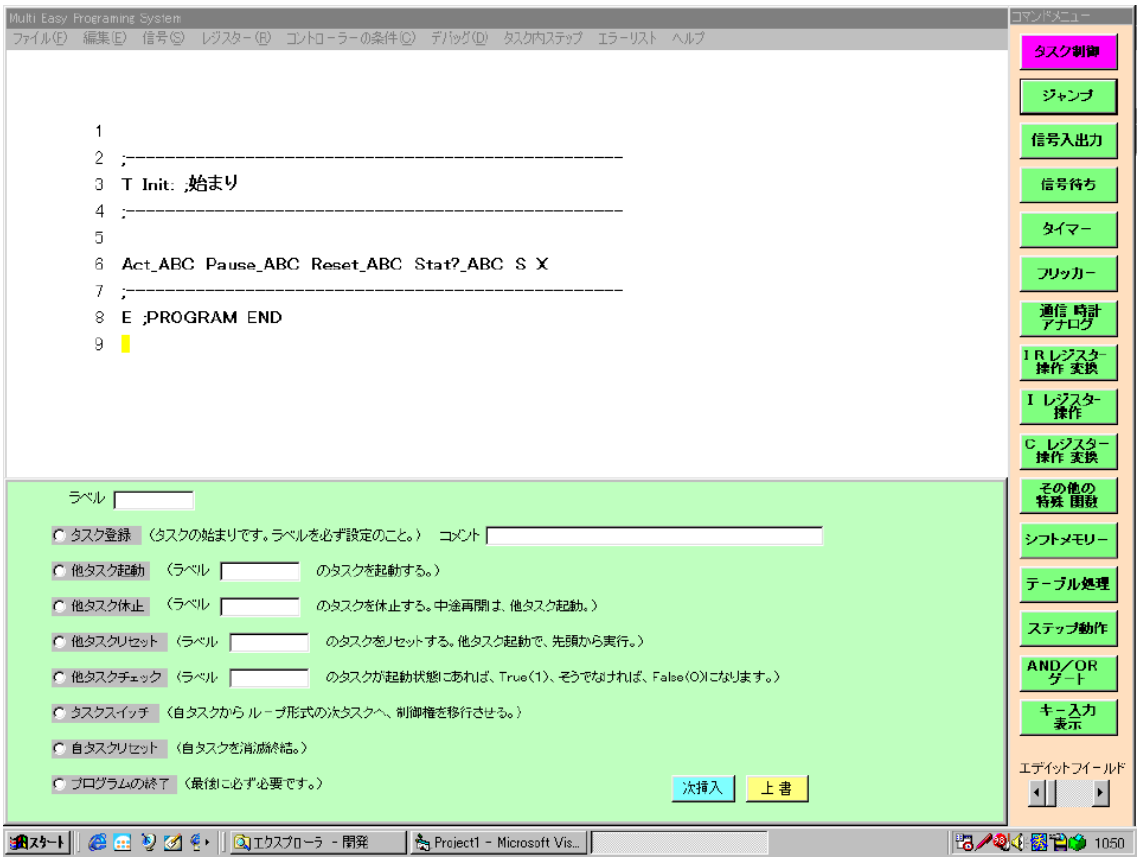
番号	設定	入力信号名	注釈	番号	設定	入力信号名	注釈
0				１６			
1				１７			
2				１８			
3				１９			
4				２０			
5				２１			
6				２２			
7				２３			
8				２４			
9				２５			
１０				２６			
１１				２７			
１２				２８			
１３				２９			
１４				３０			
１５				３１			

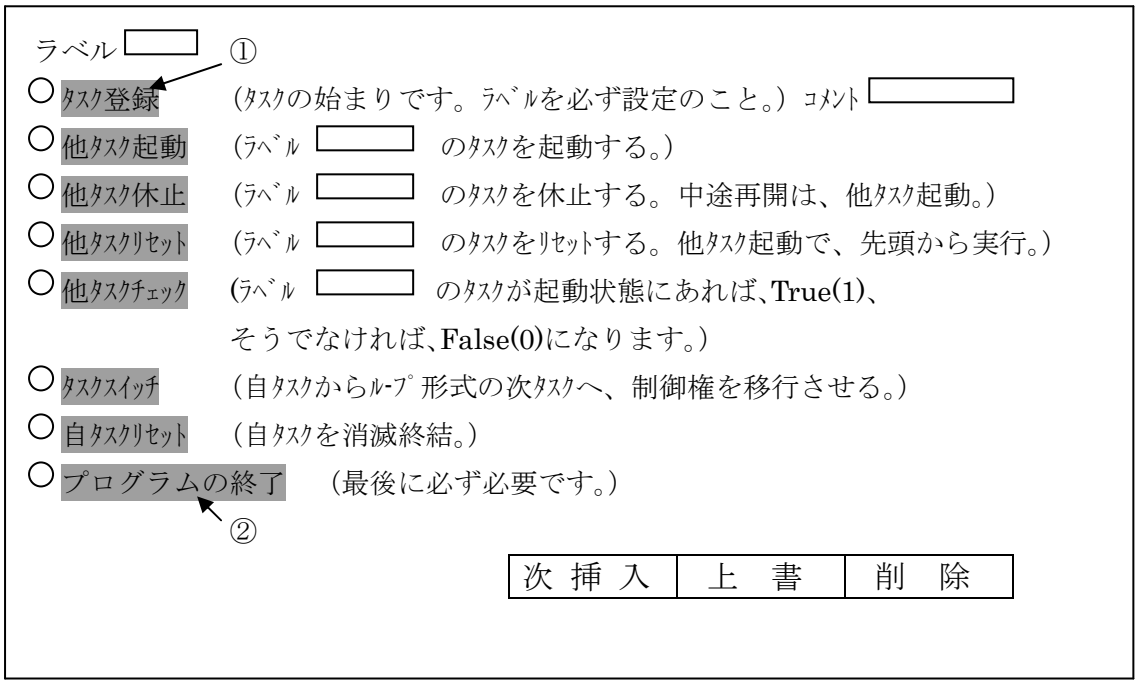
【図３】入力信号登録画面

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。

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、ＩＯ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（ジェネレイト）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。
レジスター類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

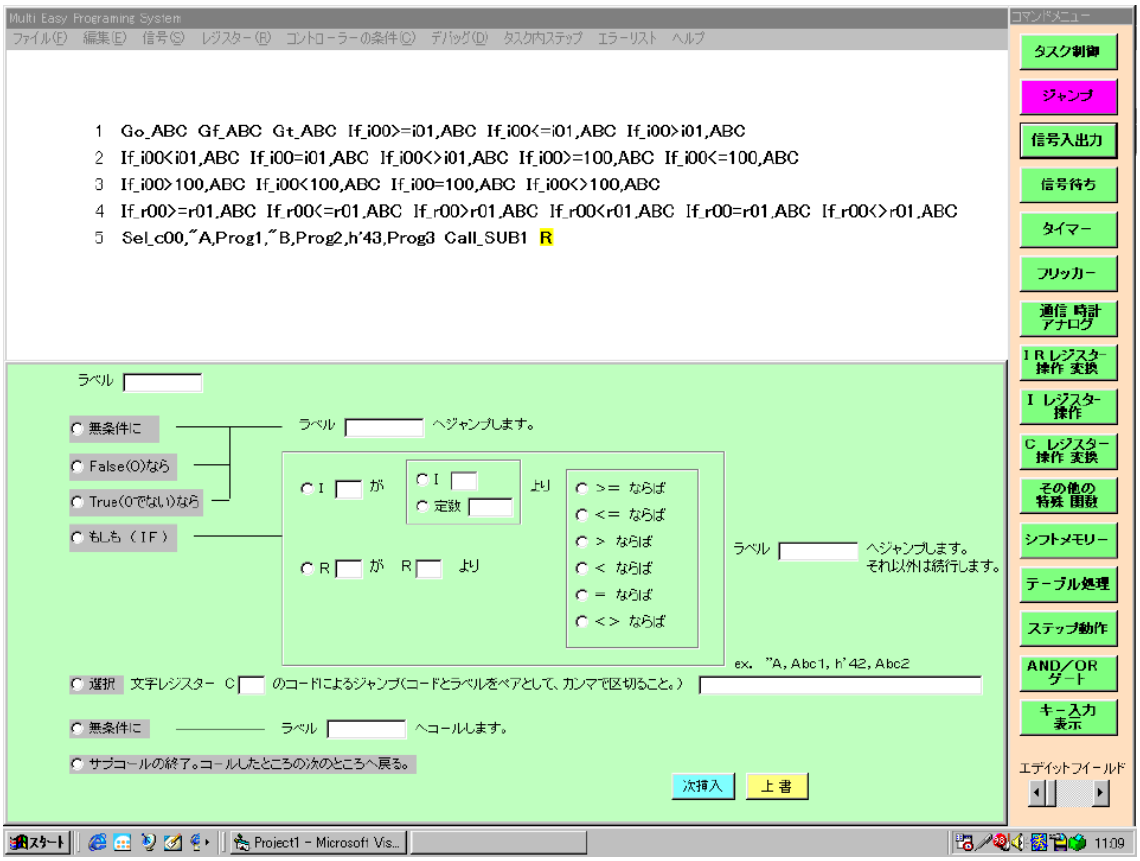


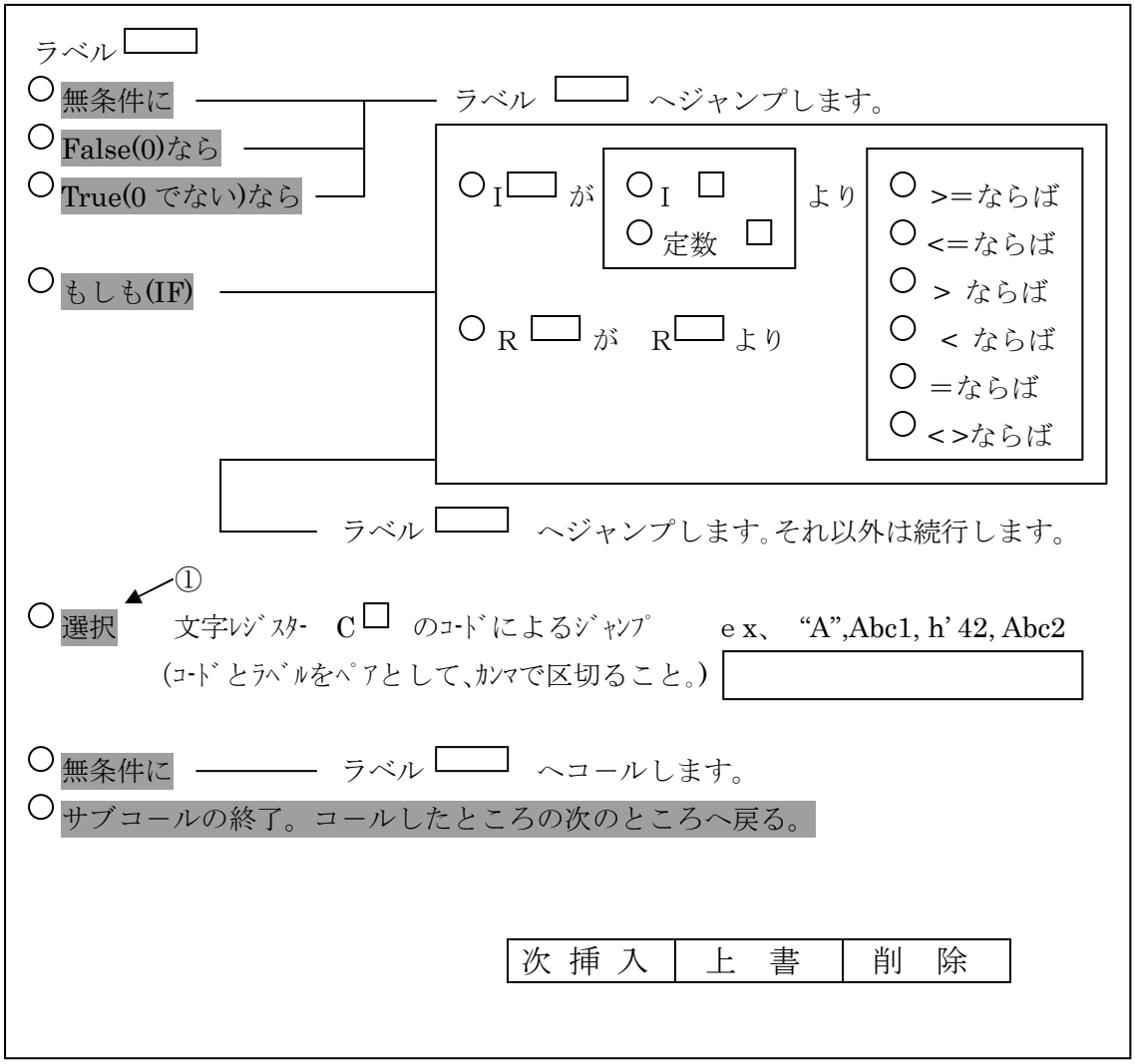


【図４】タスクコマンドの選択画面

- 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、最初に実行するのが、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

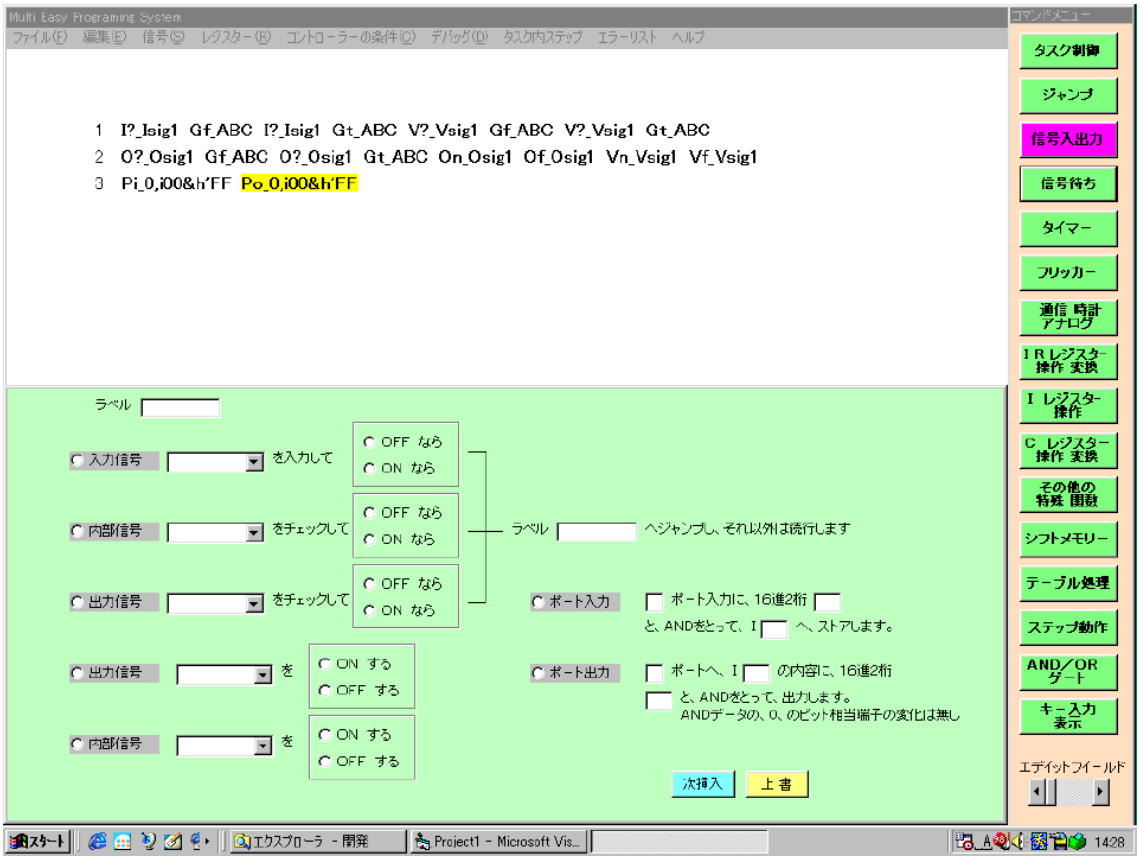

Multi Easy Programing System
ファイル(F)　編集(E)　信号(S)　レジスター(R)　コントローラーの条件(C)　デバッグ(D)　タスク内ステップ　エラーリスト　ヘルプ
1　Go_ABC　Gf_ABC　Gt_ABC　If_i00>=i01,ABC　If_i00<=i01,ABC　If_i00>i01,ABC
2　If_i00<i01,ABC　If_i00=i01,ABC　If_i00<>i01,ABC　If_i00>=100,ABC　If_i00<=100,ABC
3　If_i00>100,ABC　If_i00<100,ABC　If_i00=100,ABC　If_i00<>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
False(0)なら
True(0でない)なら
もしも（IF）
ラベル　へジャンプします。
I が　I 定数　より
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。
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。)
ex.　"A, Abc1, h'42, Abc2
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
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 レジスター操作 変換
I レジスター操作
C レジスター操作 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
スタート
Project1 - Microsoft Vis...
11:09

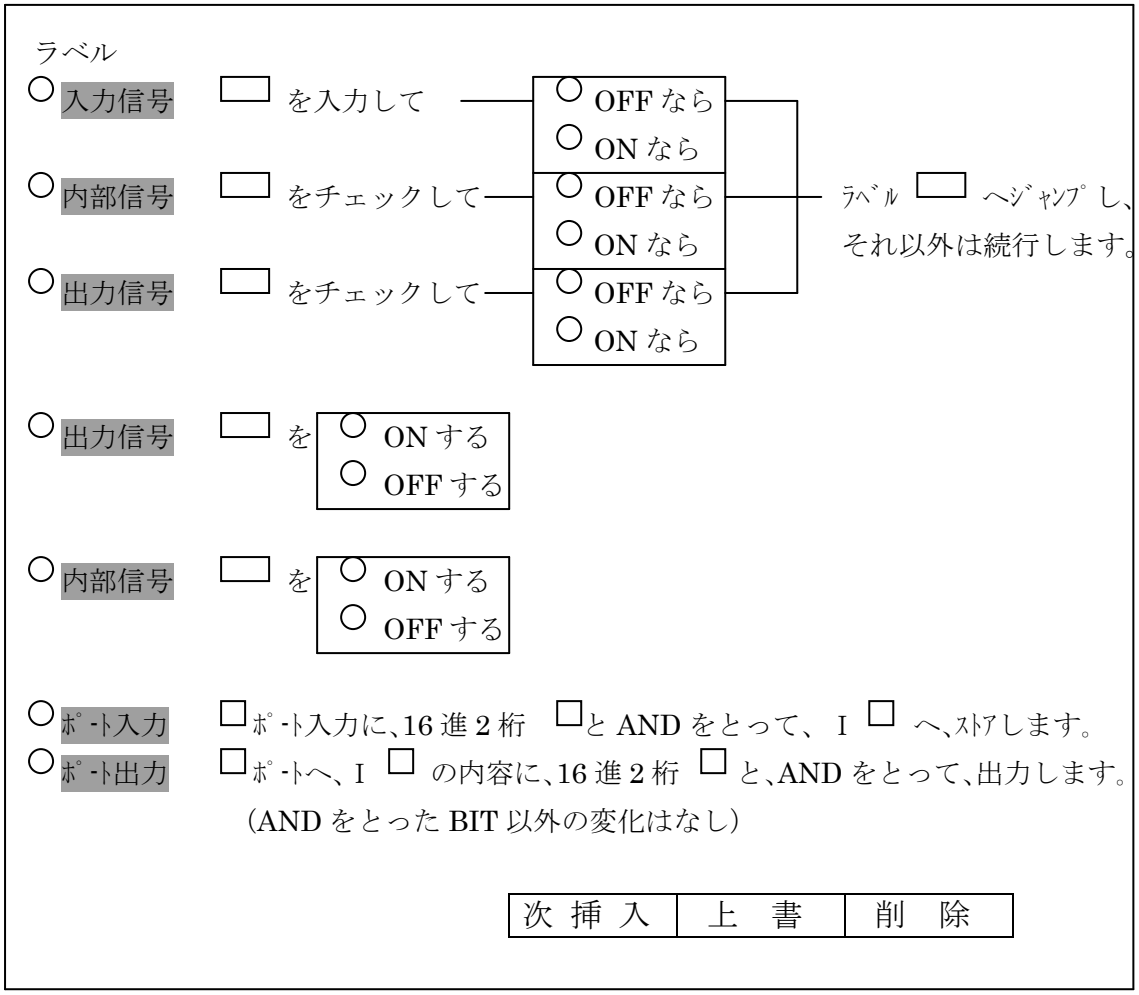


【図５】ジャンプコマンドの選択画面

- 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

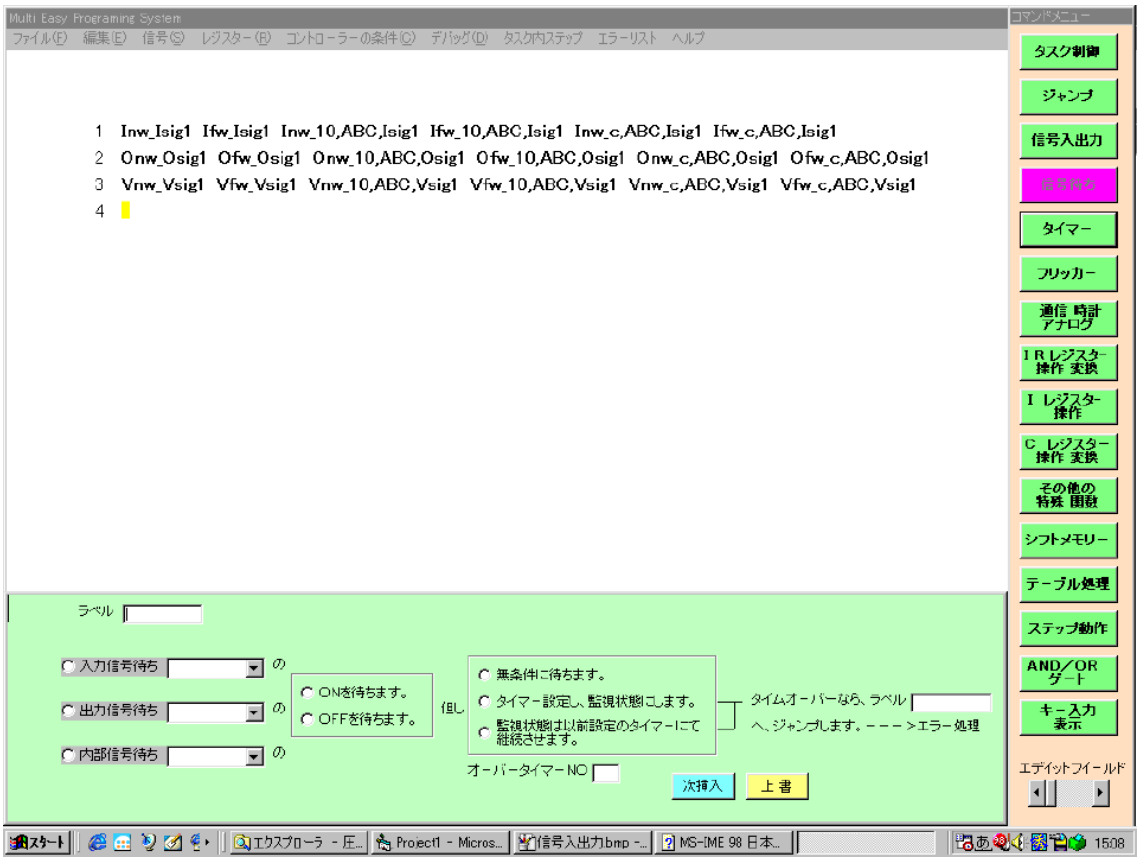


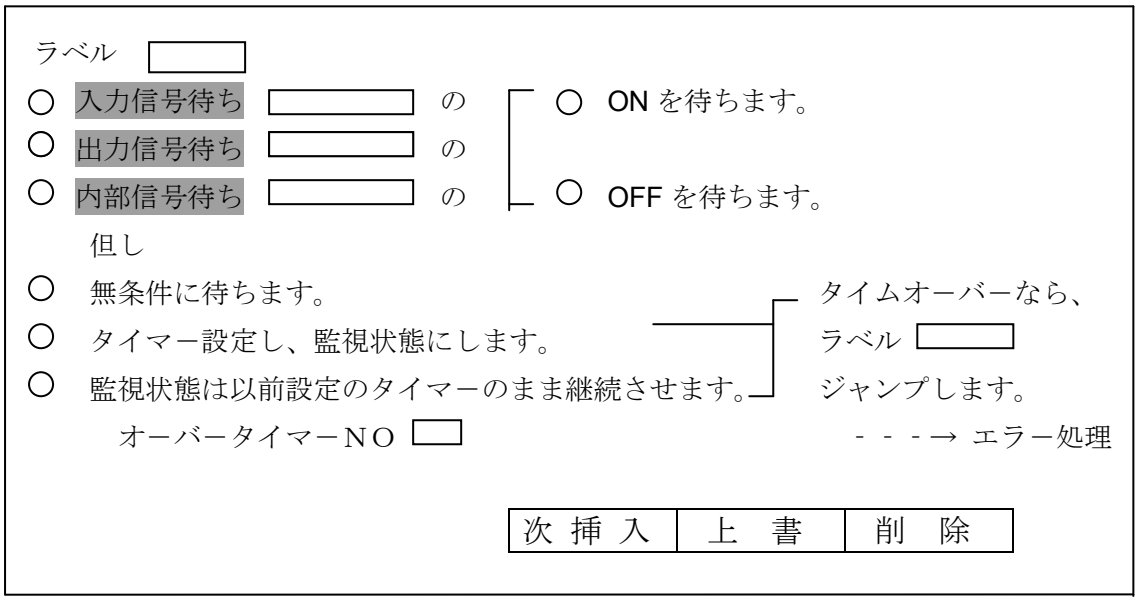


【図６】信号入出力コマンドの選択画面

- 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、８×４＝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

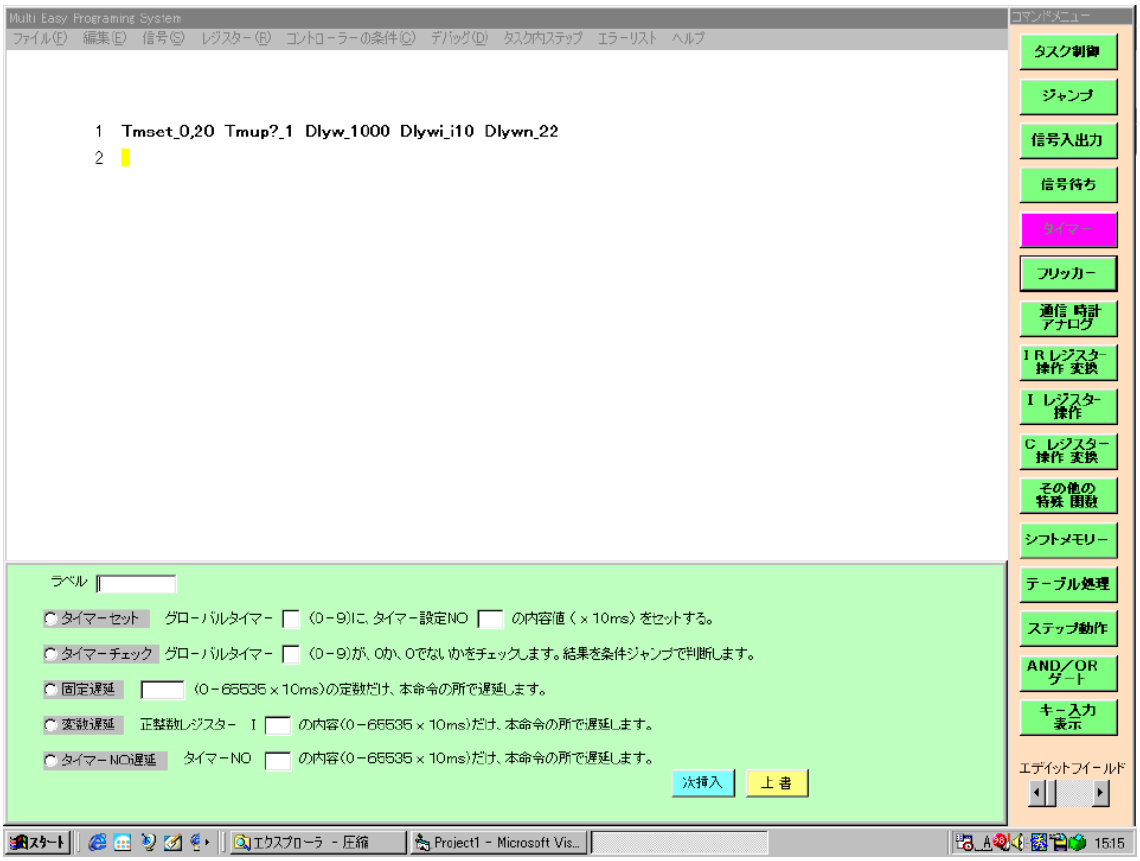




【図７】信号待ちコマンドの選択画面

- 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



ラベル

○ ﾀｲﾏｰｾｯﾄ　ｸﾞﾛｰﾊﾞﾙﾀｲﾏｰ　(0－9)に、ﾀｲﾏｰ設定 NO　(2 桁)の内容値(x10ms)をｾｯﾄする。

○ ﾀｲﾏｰﾁｪｯｸ　ｸﾞﾛｰﾊﾞﾙﾀｲﾏｰ　(0－9)が、0 か、0 でないかをﾁｪｯｸします。結果を条件ｼﾞｬﾝﾌﾟで判断します。

○ 固定遅延　(0－65535 x 10ms)の定数だけ、本命令の所で遅延します。

○ 変数遅延　正整数ﾚｼﾞｽﾀｰ　I　の内容(0－65535 x 10ms)だけ、本命令の所で遅延します。

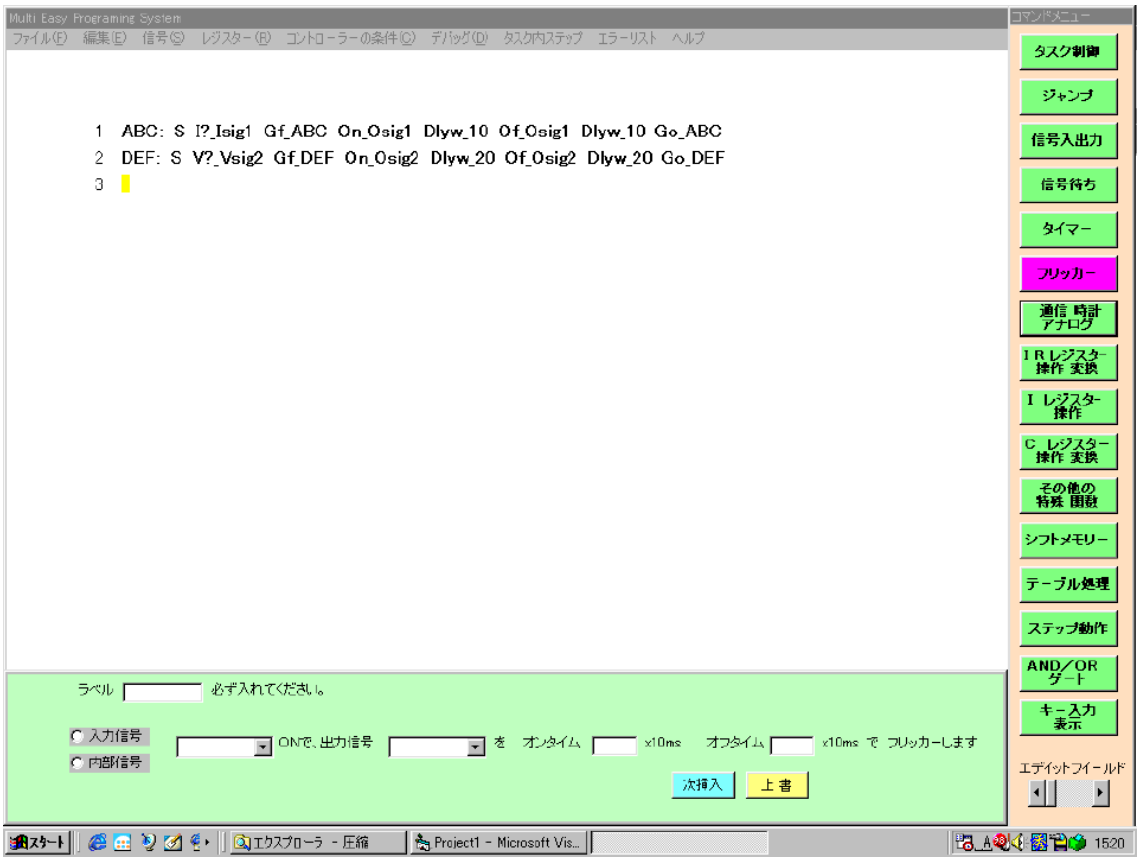
○ ﾀｲﾏｰNO 遅延　ﾀｲﾏｰ設定 NO　の内容(0－65535 x 10ms)だけ、本命令の所で遅延します

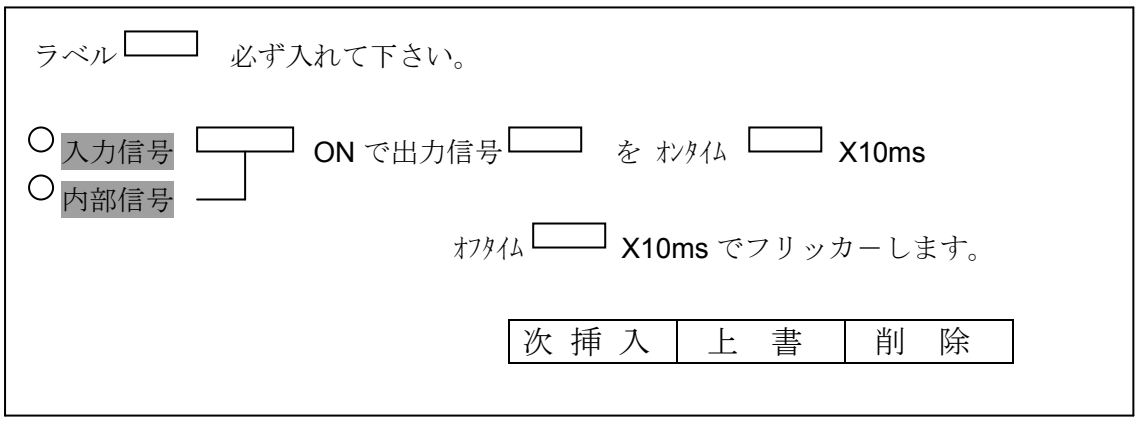
次 挿 入	上　書	削　除

【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

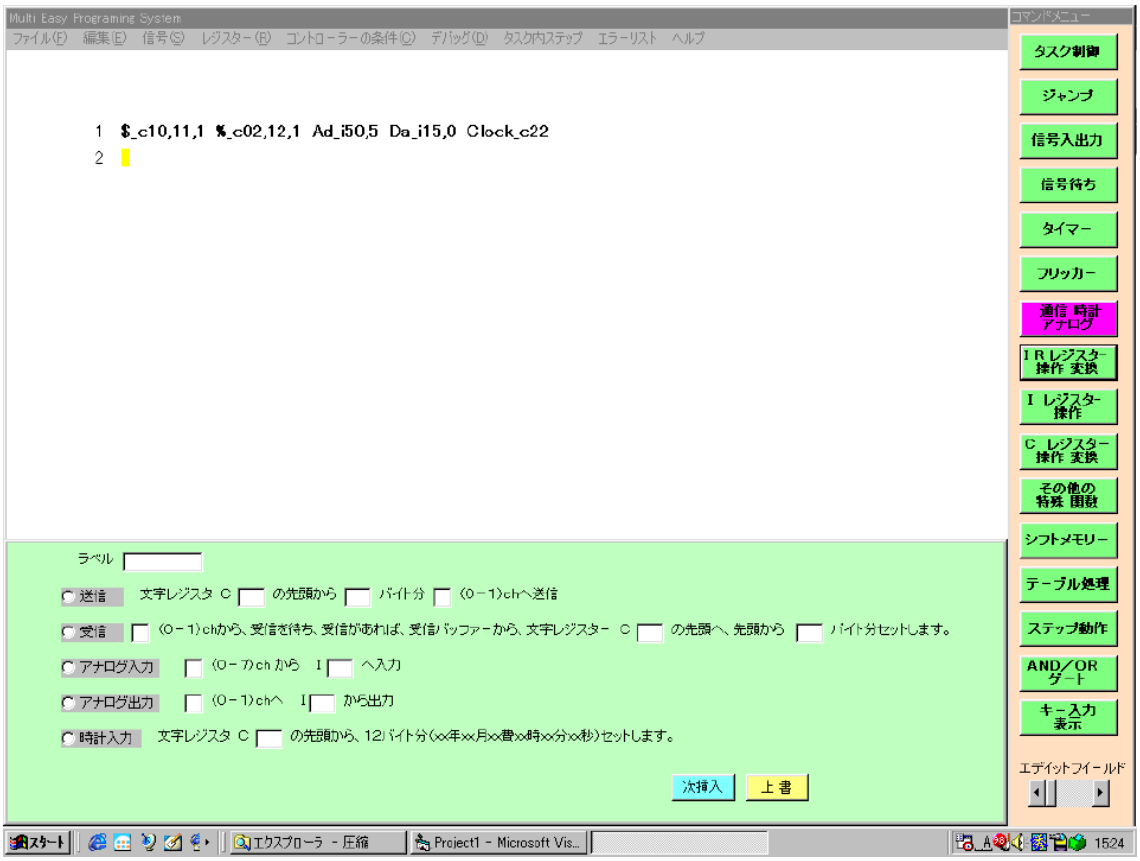


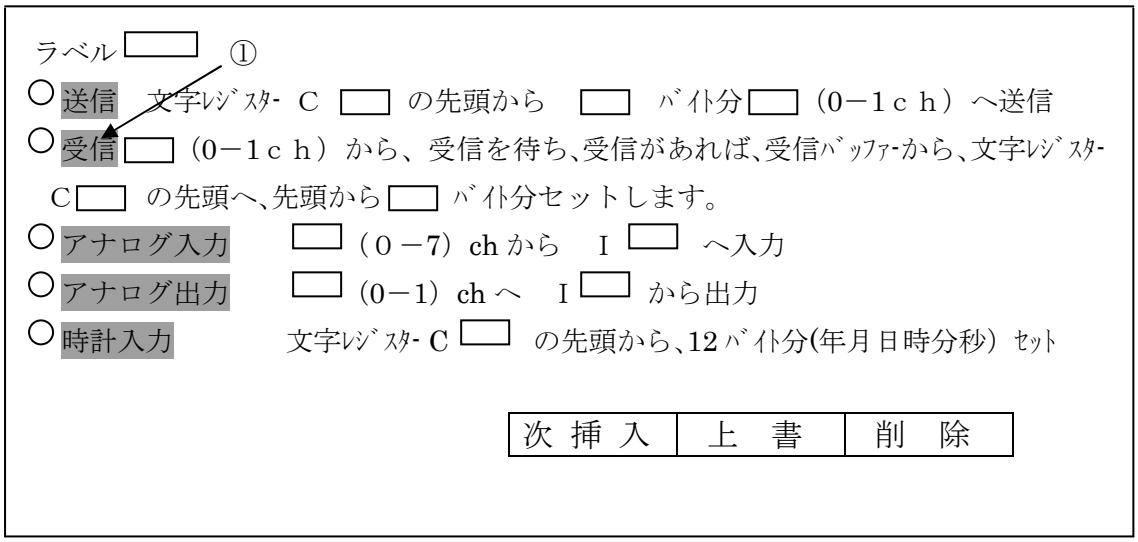


【図９】フリッカーコマンドの選択画面

- 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

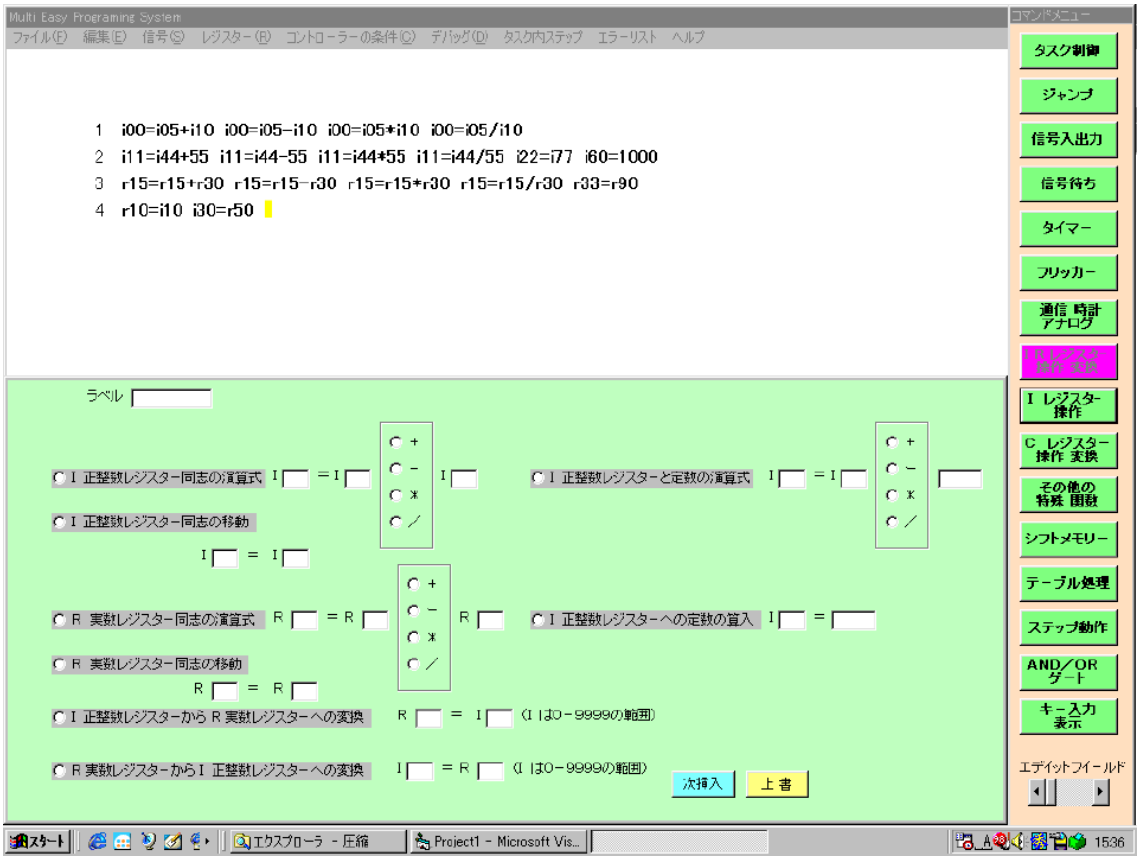


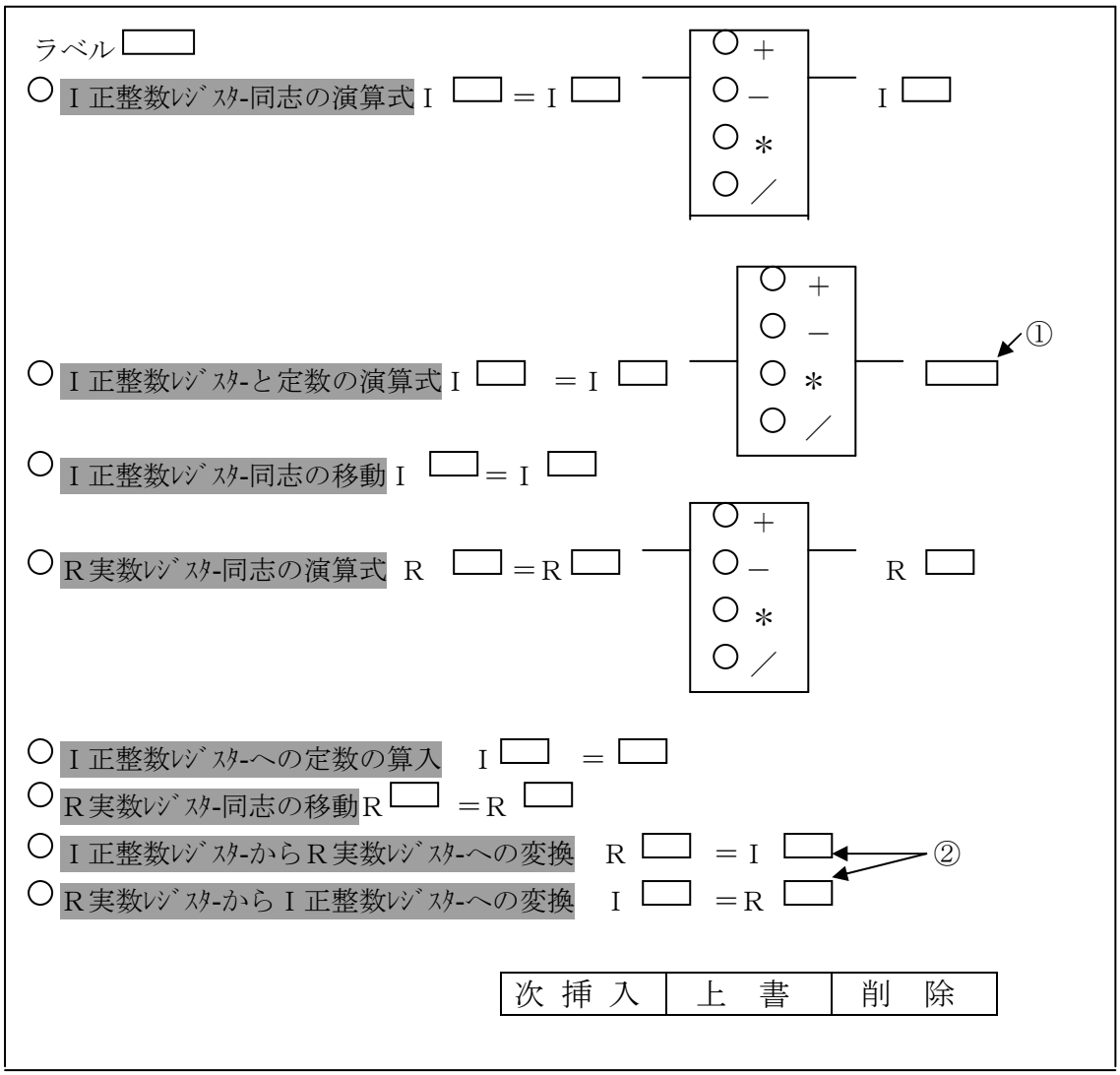


【図１０】通信、時計、アナログコマンドの選択画面

- 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、レジスター操作、変換コマンド

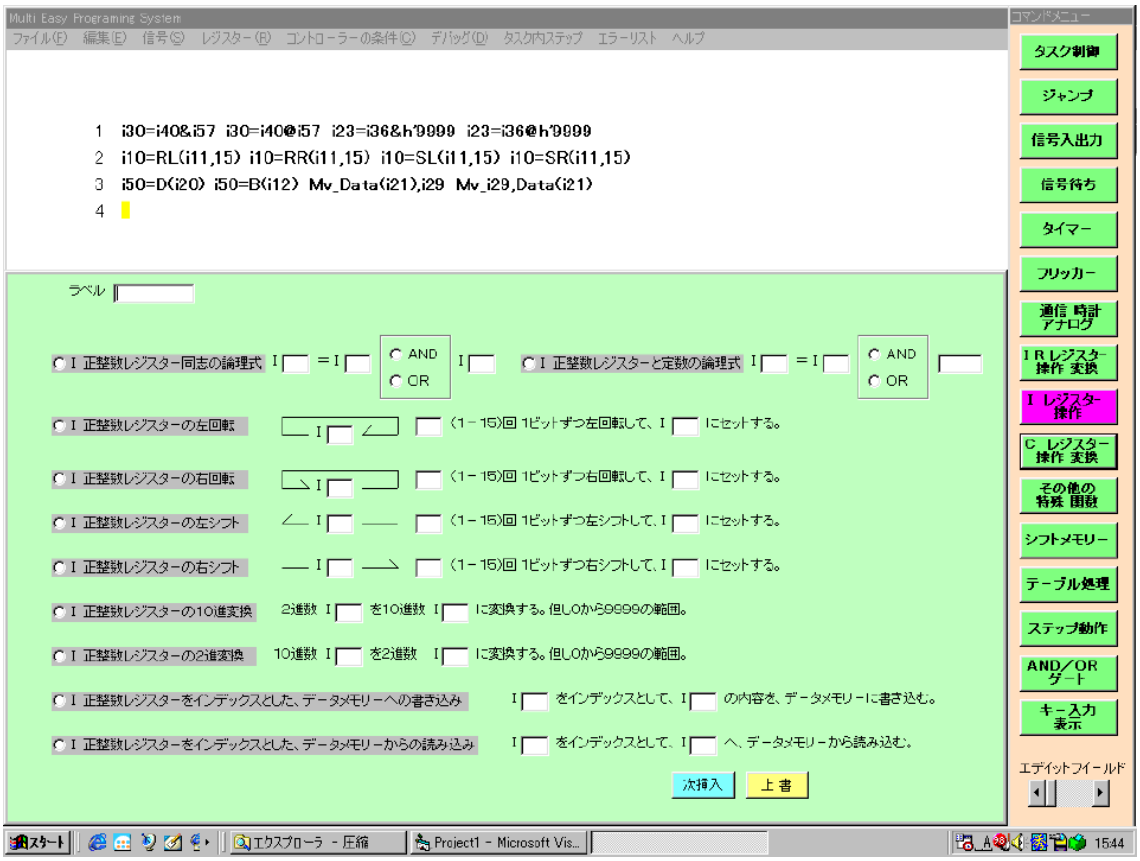


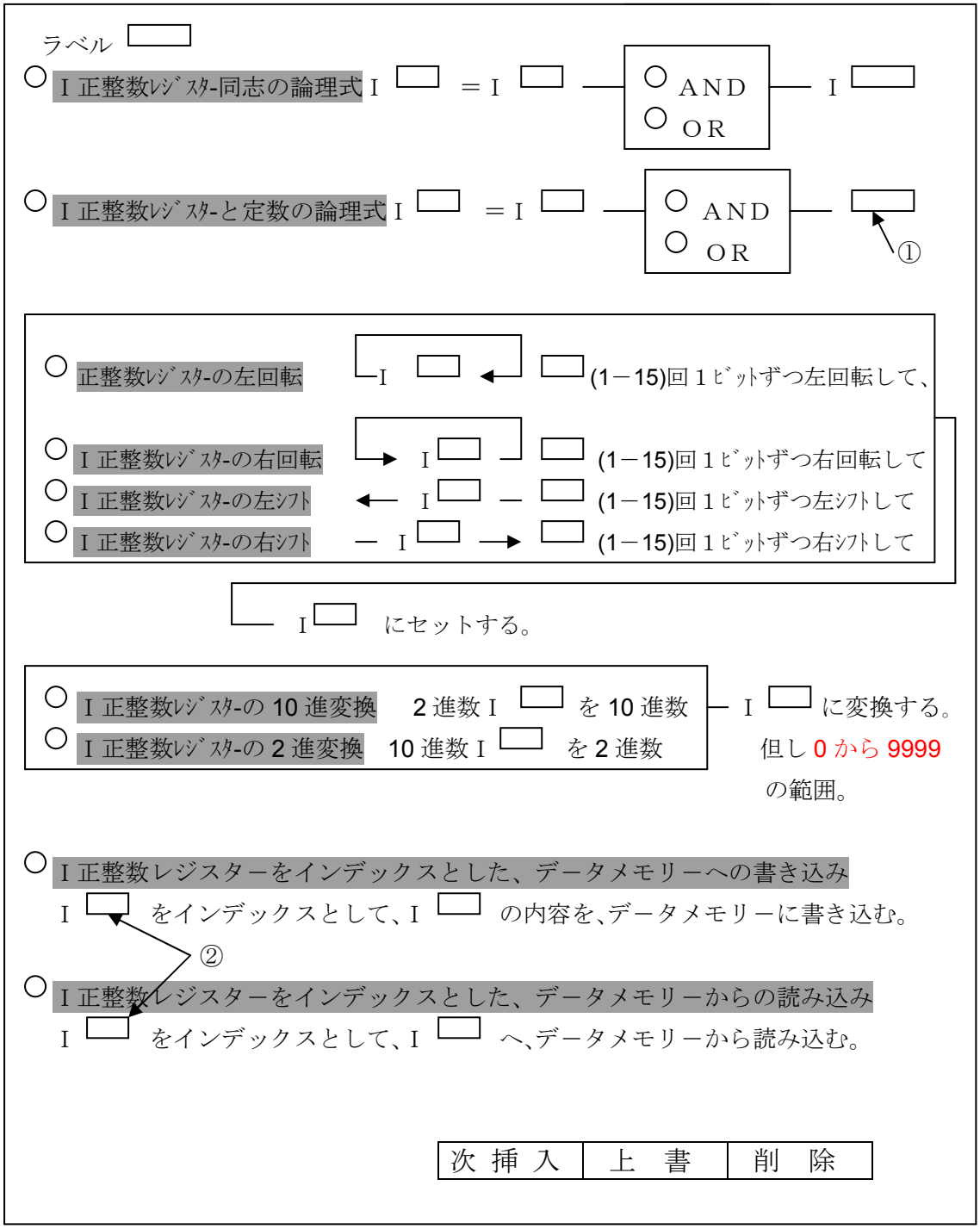


【図１１】Ｉ、Ｒ、レジスター操作、変換コマンドの選択画面

- Ｉ正整数レジスターへの定数①の算入は０から６５５３５までです。
- ②Ｉ正整数レジスターとＲ実数レジスター間のデータ範囲は、０から９９９９までです。（Ｃ文字レジスターとＲ実数レジスター間の変換、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レジスター操作、変換コマンド参照の事。）

2.10　Ｉ、レジスター操作コマンド

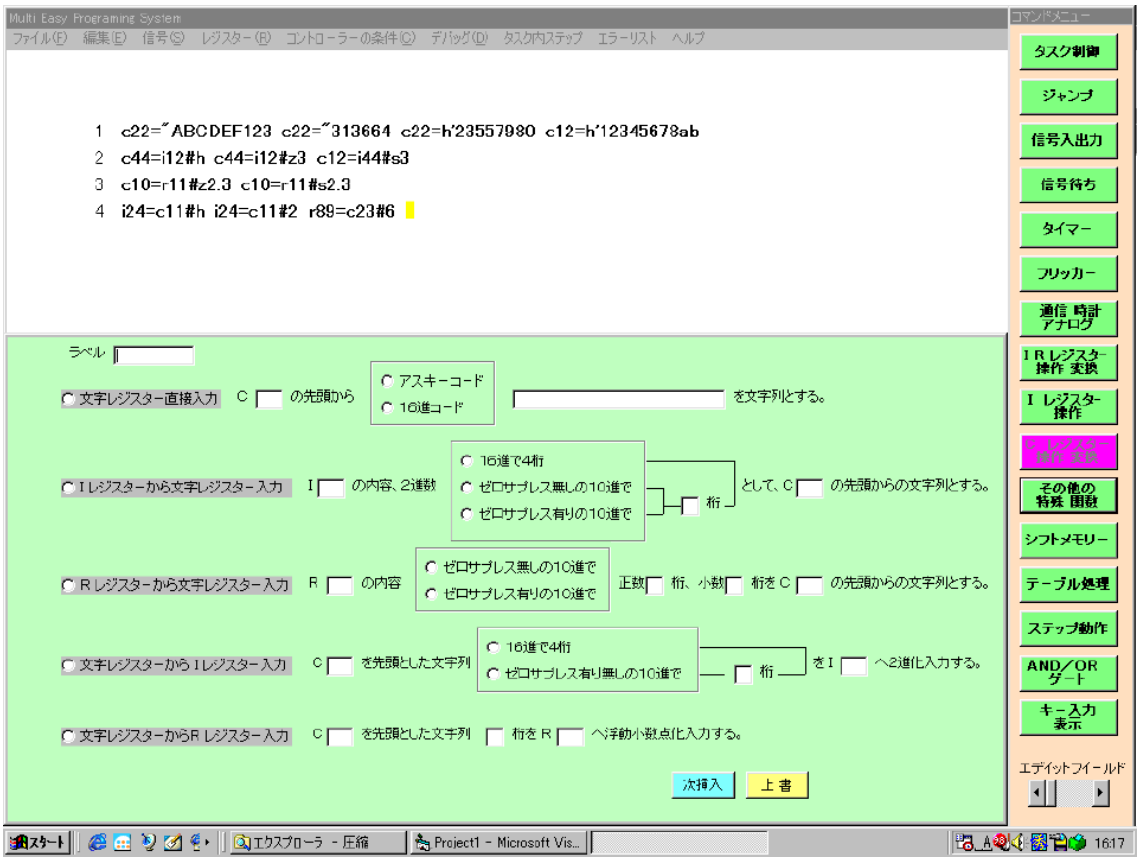


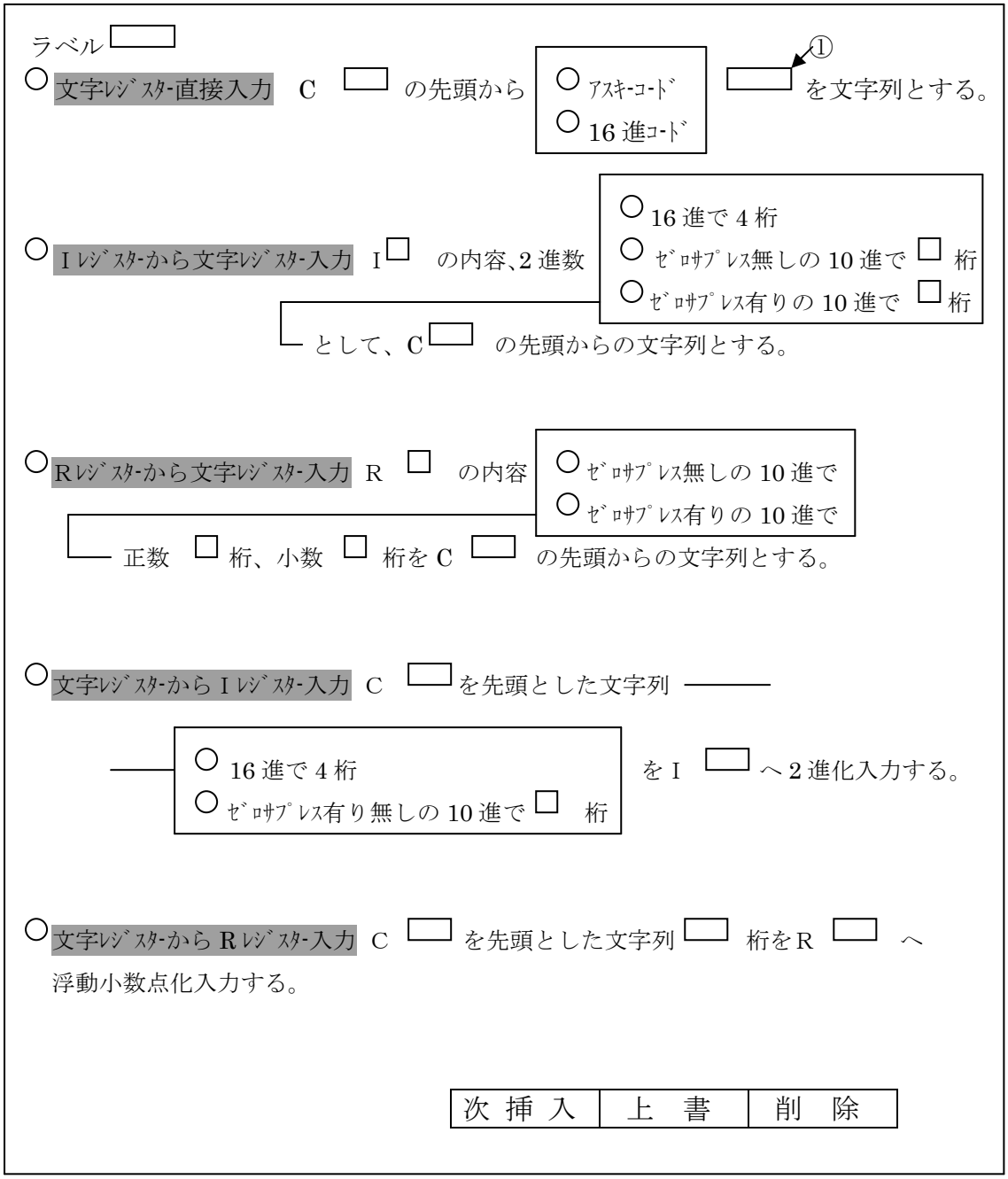


【図１２】Ｉ、レジスター操作コマンドの選択画面

- Ｉ正整数ﾚｼﾞｽﾀｰ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

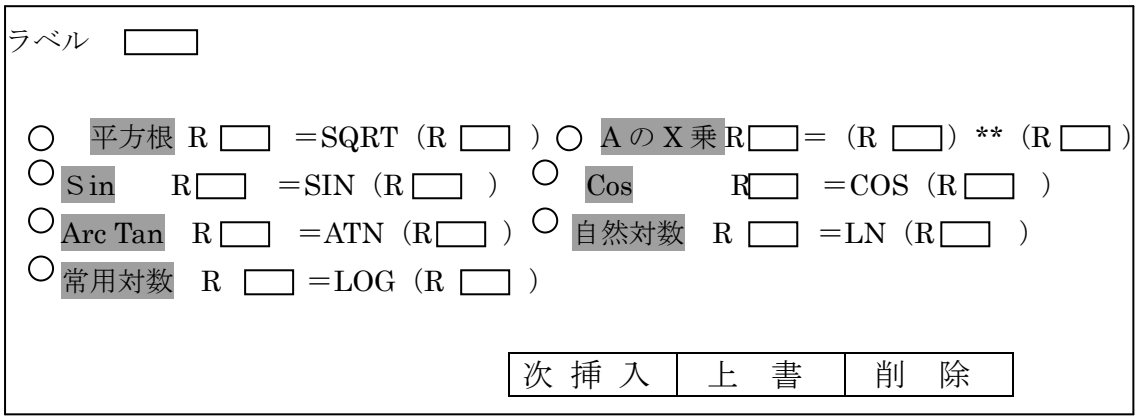




【図１３】C、レジスター操作、変換コマンドの選択画面

- 文字レジスター直接入力①での文字及び１６進に、“ や、h’は不要です。

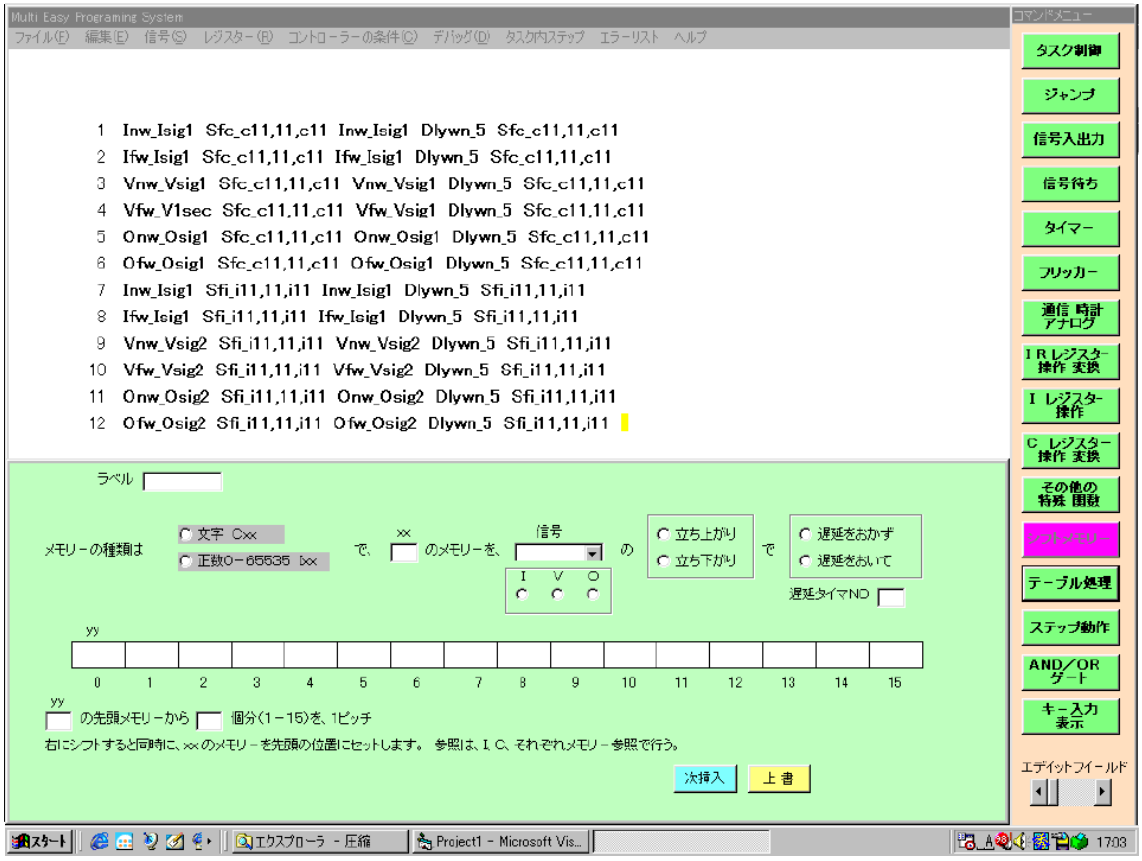
2.12　その他の特殊、関数コマンド

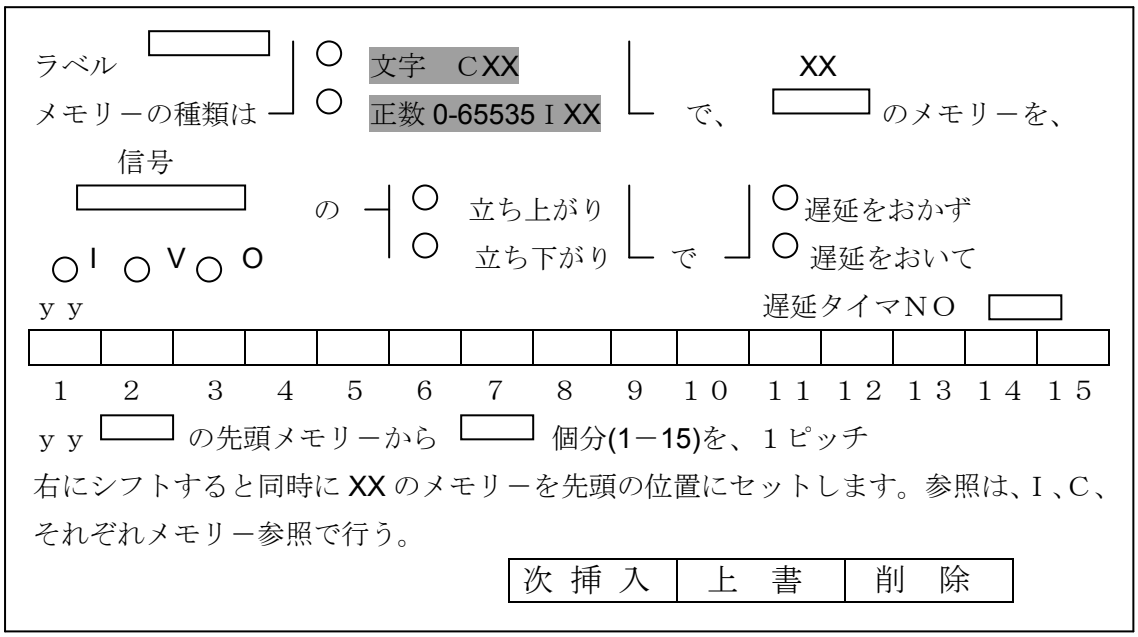


【図１４】その他の特殊、関数コマンドの選択画面

- 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

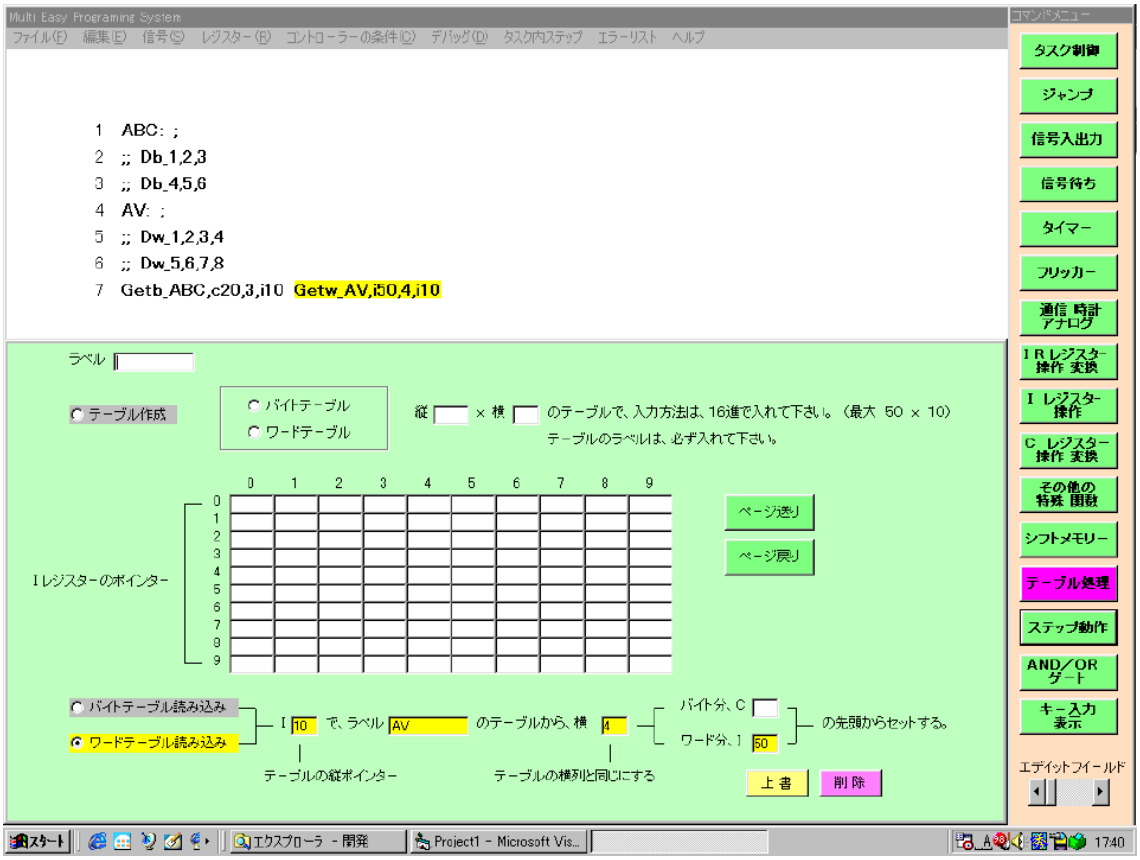


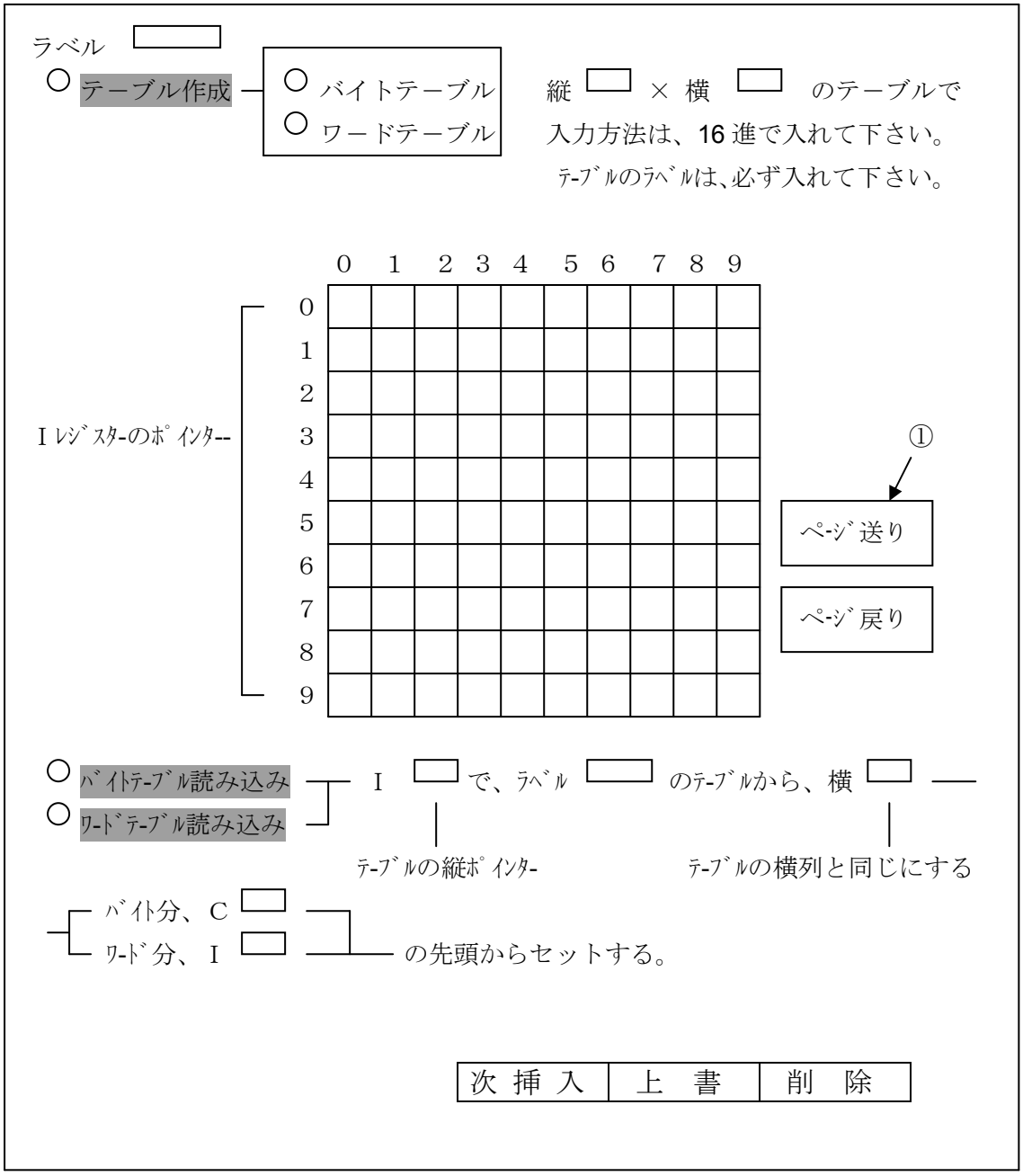


【図１５】シフトメモリーコマンドの選択画面

- 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

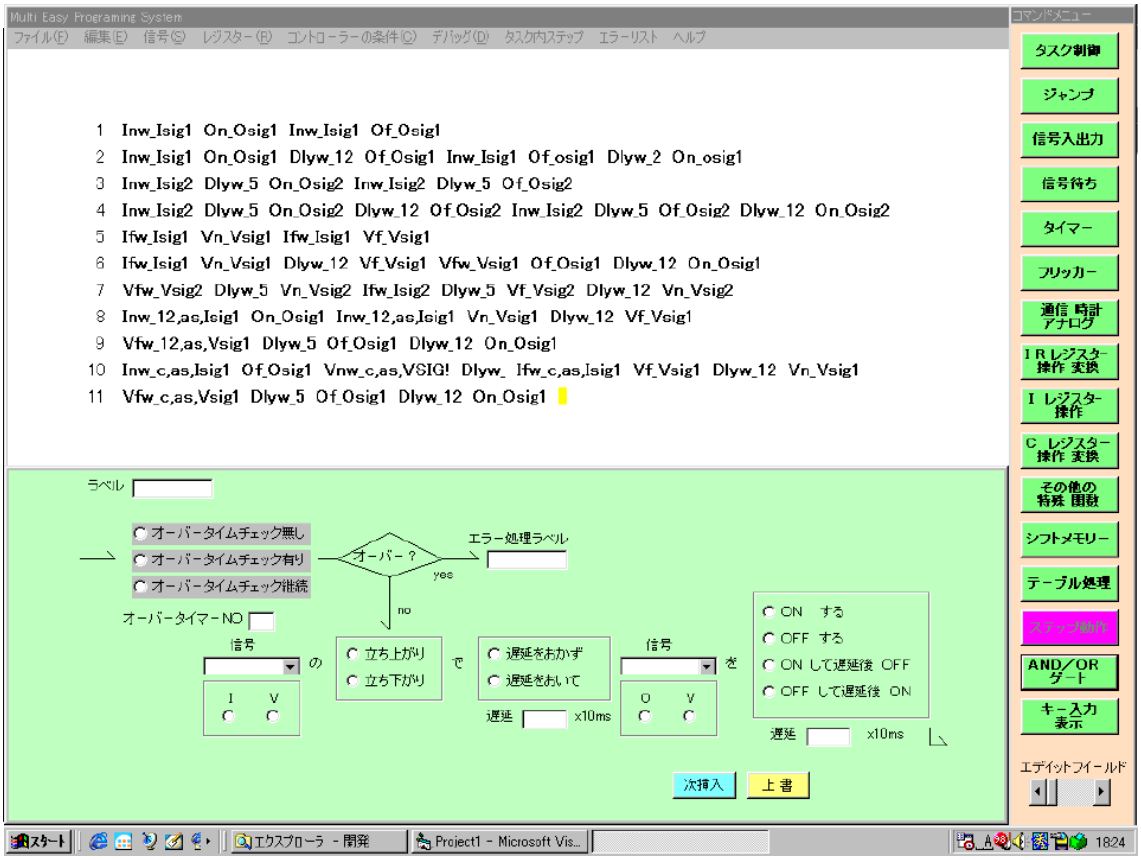


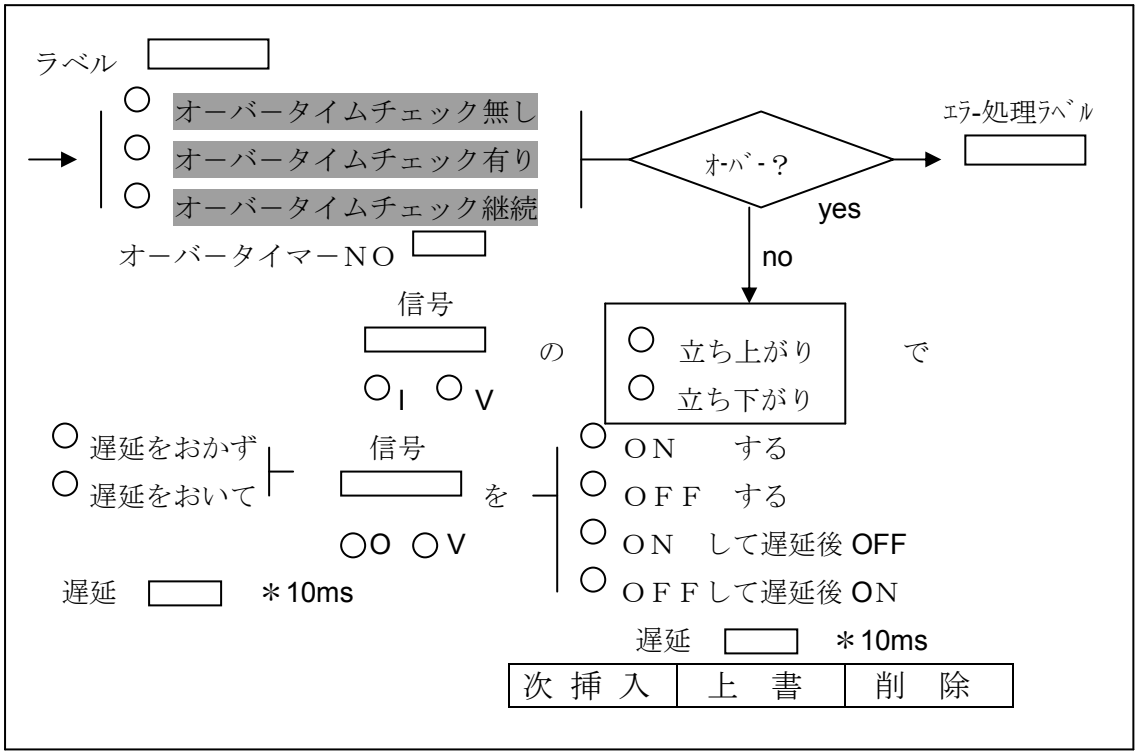


【図１６】テーブル処理コマンドの選択画面

- 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。
- 信号入力を縦ポイターとして、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。

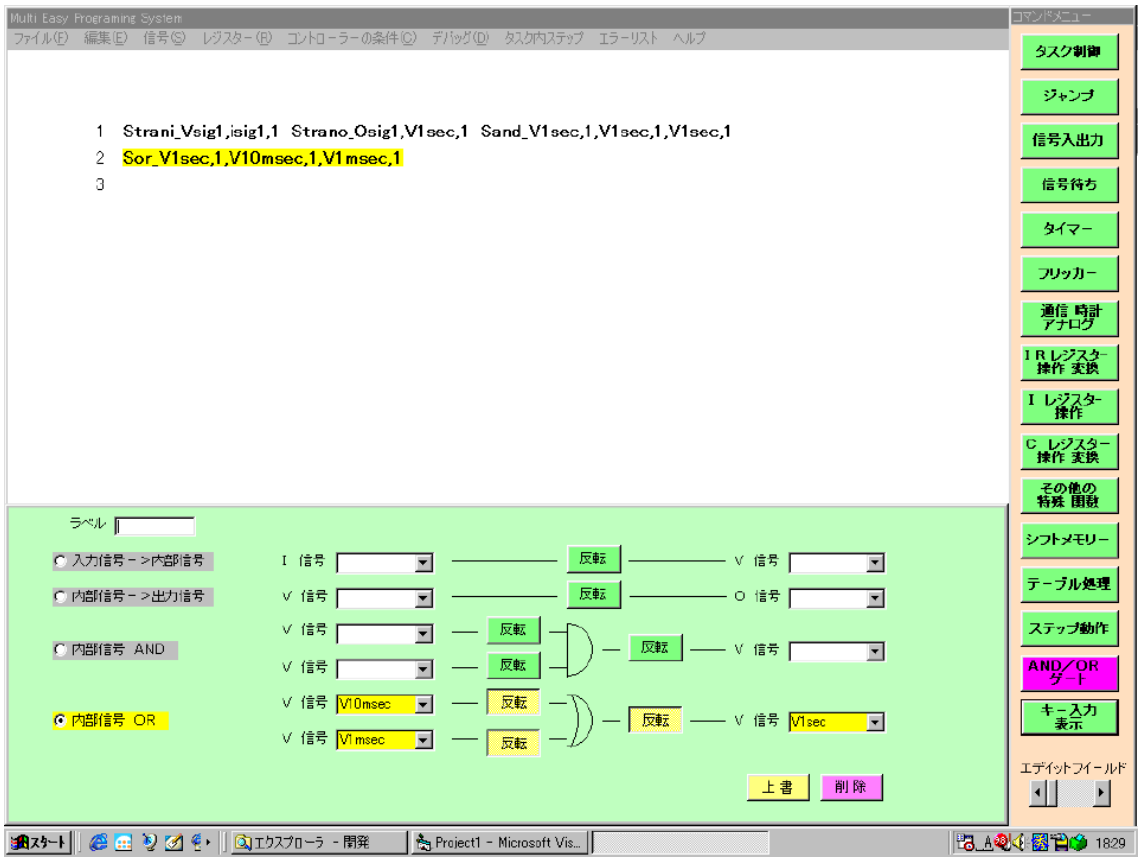
2.15　ステップ動作コマンド

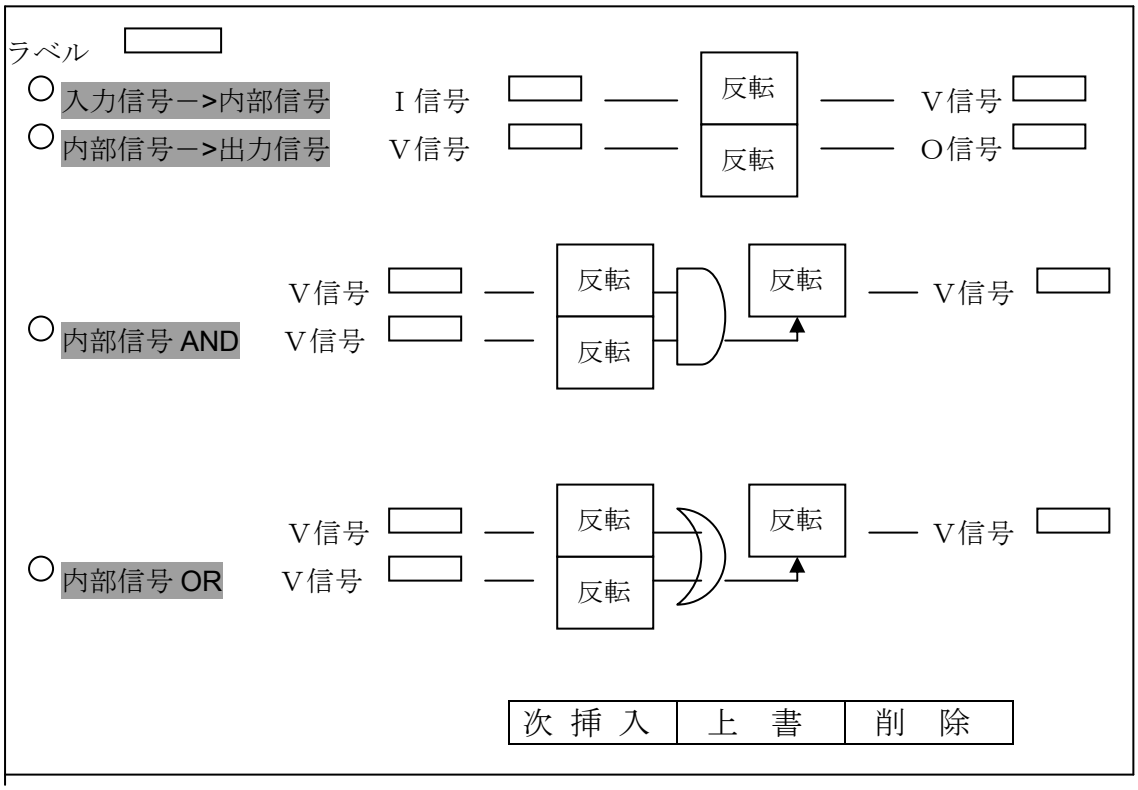




【図１７】ステップ動作コマンドの選択画面

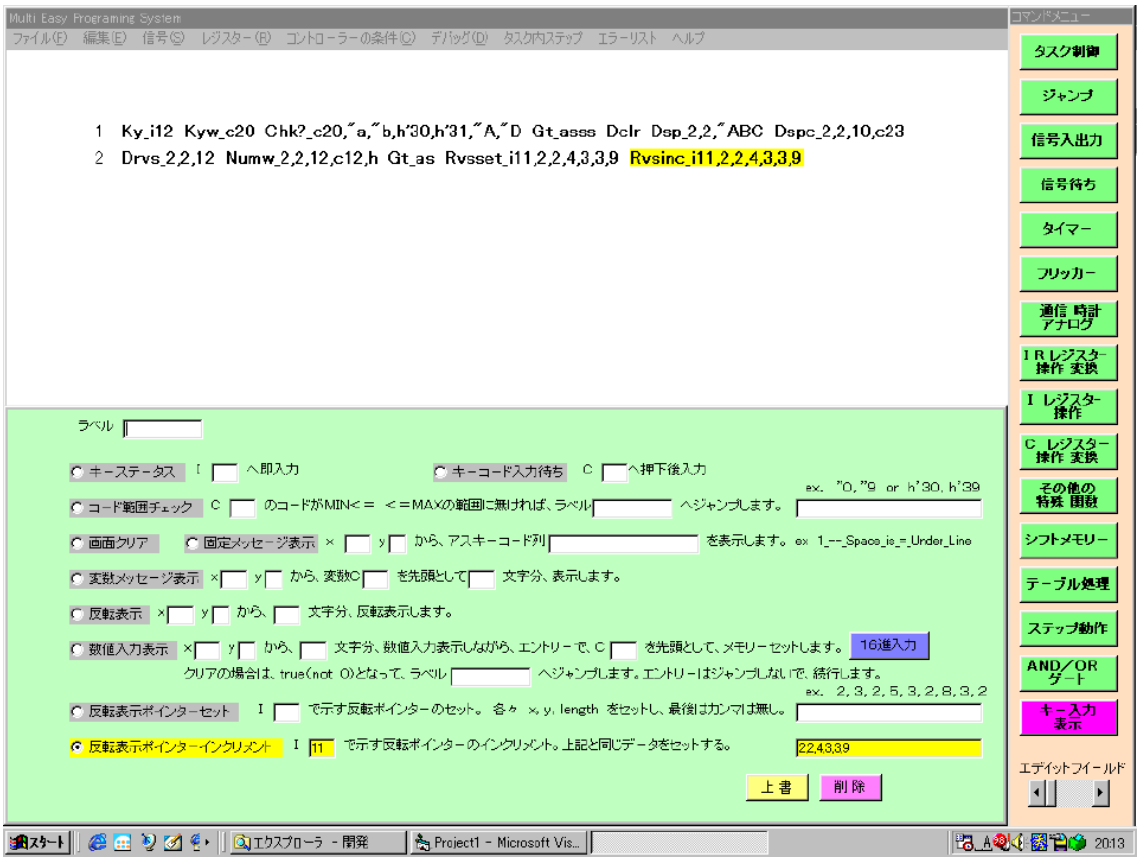
2.16　ＡＮＤ／ＯＲゲートコマンド

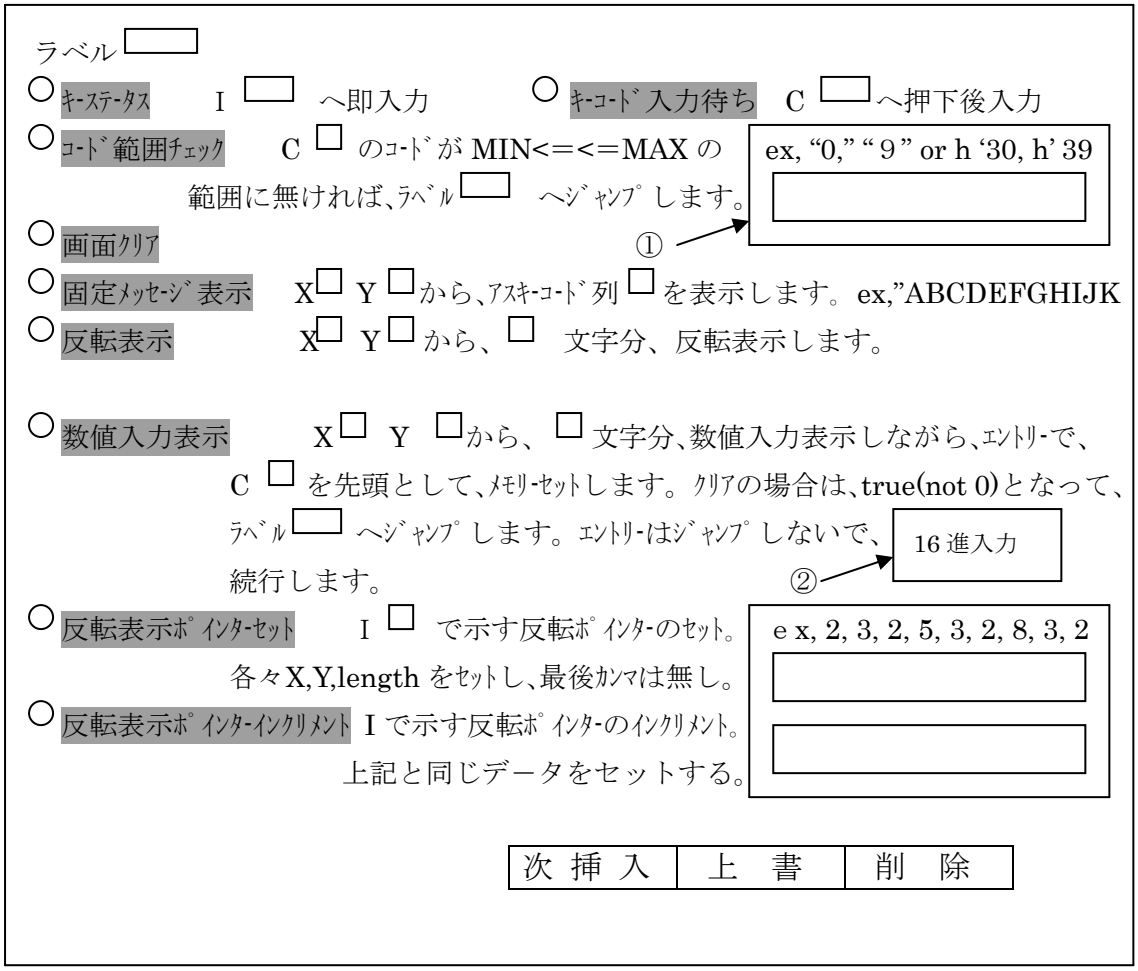




【図１８】ＡＮＤ／ＯＲゲートコマンドの選択画面

2.17　キー入力表示コマンド

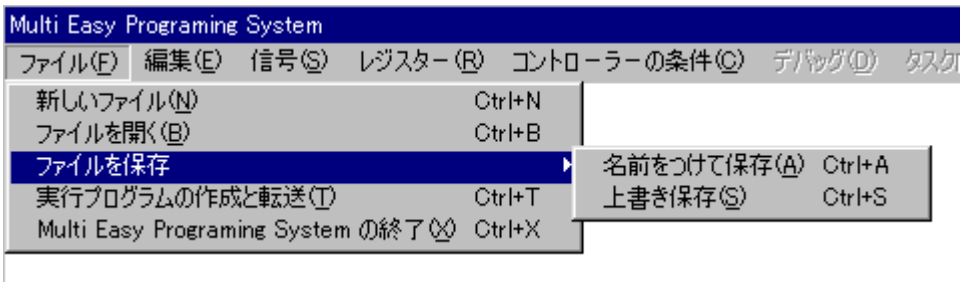




【図１９】キー入力表示コマンドの選択画面

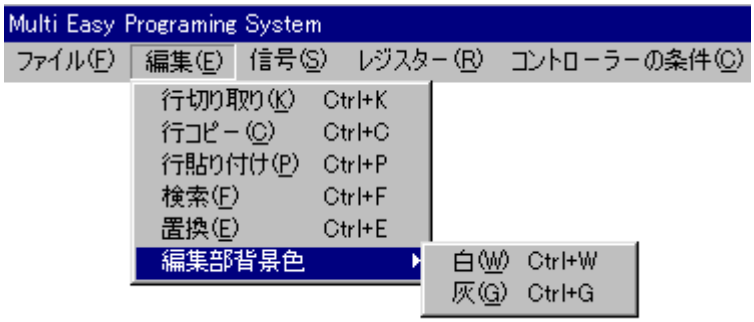
- キーコードについては、ＭＥＰＳ Ｉ／Ｏ コンパイラー 又は、ＭＥＰＳコントローラー参照の事。
- ①コード範囲チェックの範囲データ入力で、“文字形式とｈ’の 16 進形式と min／max 毎なら（ペアー毎なら）両方使用してかまいません。
- 数値入力表示で②の１６進入力のスイッチがセットしているならば、０から９の次は A. . . Ｆとなります。詳細はＭＥＰＳ Ｉ／Ｏ コンパイラーの１２、ＬＣＤ出力の中のＮｕｍｗ＿命令を参照の事。
- 反転表示ポインターセットと反転表示ポインターインクリメントとは、３バイトづつ同じデータ（位置と長さ）をセットし、インクリメント命令を実行する毎に、反転表示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

2.18　ファイルメニュー



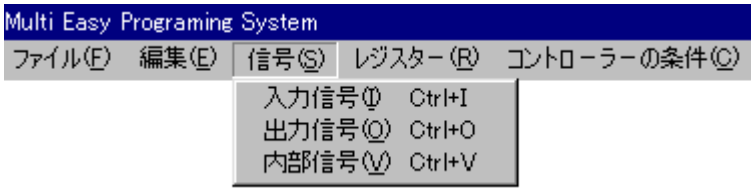
ファイルの開閉、実行プログラムの作成と転送、MEPS の終了など、MEPS の操作コントロールに寄与するところが大です。

2.19　編集メニュー



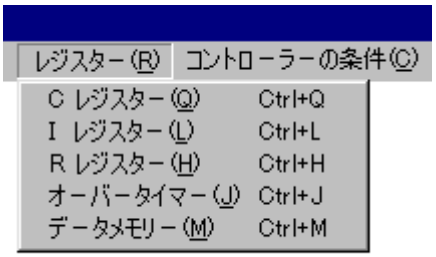
行レベルでの編集と、各命令における検索、置換です。後述する編集機能もありミックスして使用してかまいません。編集部背景色は、デフォルトは白ですが、各命令毎はっきり識別させるには、灰がいいです。

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ーの設定は、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）

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）

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実数レジスター（Ｒ００－Ｒ９９）

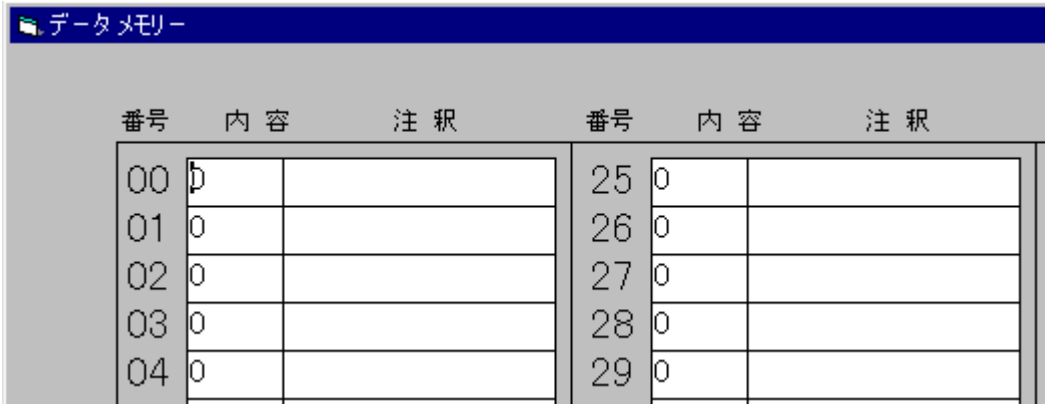
内容は実数±０．ＤＤＤＤＤＤＥ±ＮＮの浮動小数点形式で、±０．ＤＤＤＤＤＤ（000000－999999）有効数字１０進６桁×±ＮＮ（+63‥‥‥－64）乗の範囲です。

2.25　オーバータイマー値（００－９９）

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）



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

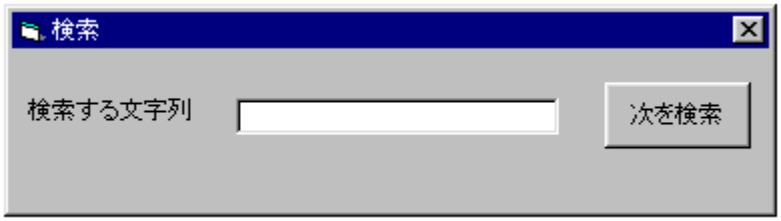
ＣＨ１、ＣＨ２の通信条件の設定とＡＤＣ、ＤＡＣのオプション設定、ＭＥＰＳコントローラーの表示スピード設定です。オンラインデバッグ時は、上図ＣＨ１の設定をして下さい。

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。）

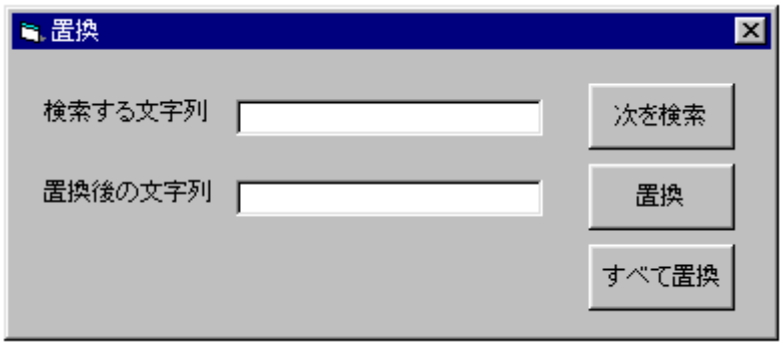
2.28　デバッグメニュー

- 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では、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

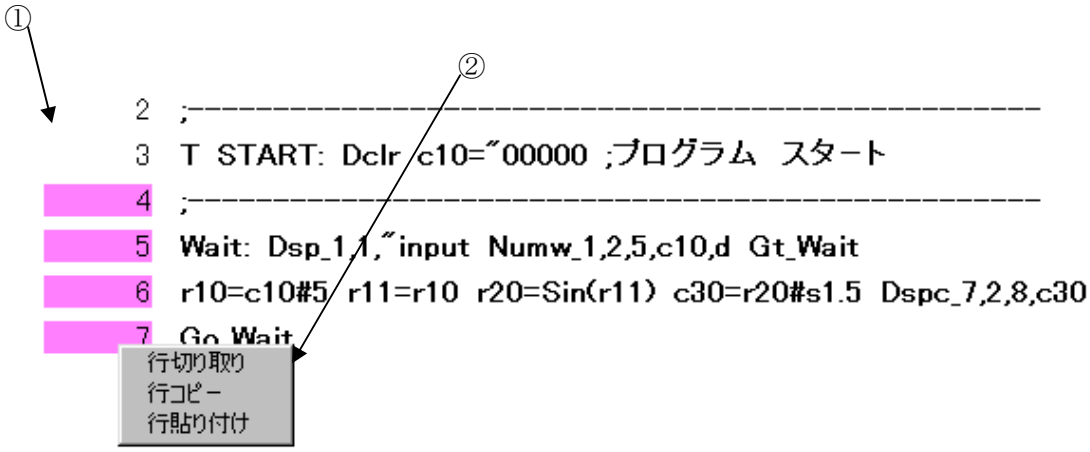


- エディットフィールドにおける検索文字列をサーチします。



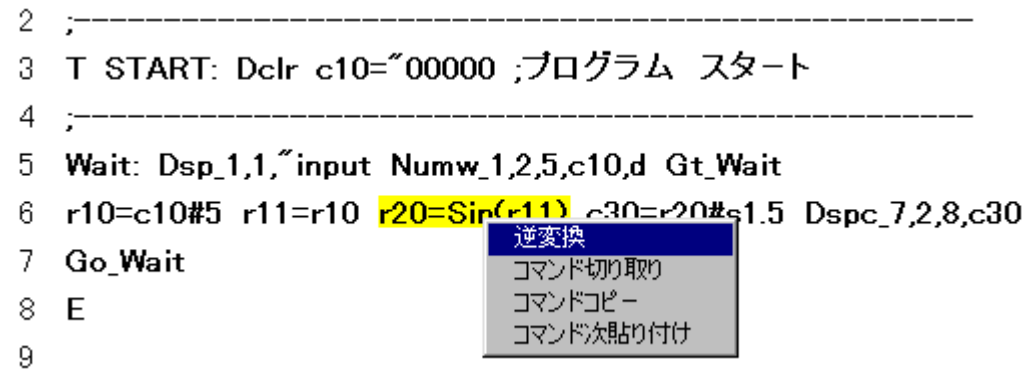
- エディットフィールドにおける検索文字列を置換後の文字列に置換します。
- すべて置換で一斉に置換します。

2.30　行表示からの編集



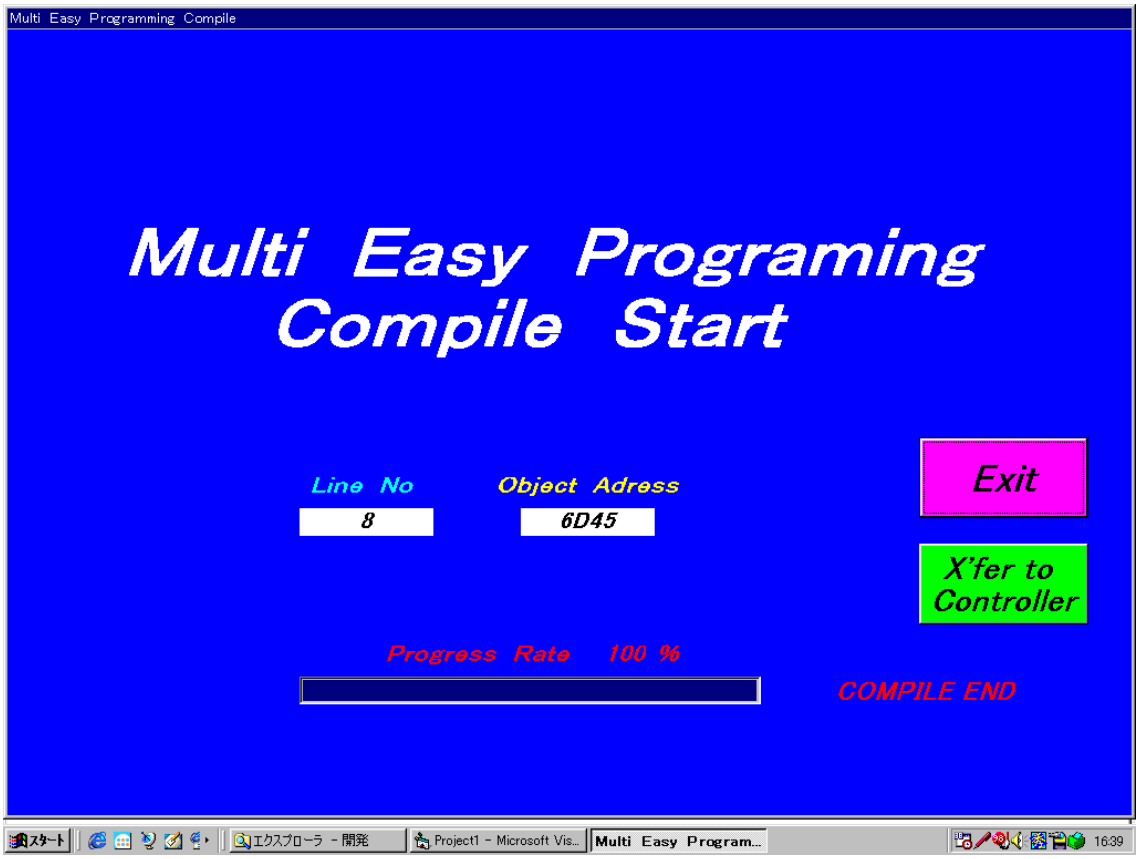
- エディットフィールドの左部分、行表示エリアの、ある行で一回マウス左クリックして下方へスライド（ドラッグではありません）すると、上図①の如くピンク色のエリアが出来ます。もう一度クリックして、ピンクエリアを確定すると、次は右クリックして、②の如き編集メニューを出して、左クリックで、行全体の切り取り、コピー、貼り付けを行う事が出来ます。これは１９、編集メニューからの行全体の切り取り、コピー、貼り付けと同じです。

2.31　各命令からの編集



- 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ＭＥＰＳコントローラーへの転送を、行います。

3.　Ｉ／Ｏコンパイラー命令

＜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)

NNNNN　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、プログラムに関与しません。

ｅｘ、緑部分はコメント

；ａｂｃｄｅｆｇｈｉｊｋ

------　；ａｂｃ；　----

＋、－、＊、／ 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づ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コンパイラーの文字の大文字、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。

ｅｘ、

```
Ｔ　Ｐ１：　--------
　　　　--------
Ｔ　Ｐ２：　--------
　　　　--------
Ｔ　Ｐ３：　--------
　　　　--------
```

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。

本コンパイラーの特長は、各タスクは、フローチャートどう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。筋書きどうりの動作をします。プログラムは、サブルーチンではなく、共通ルーチン（Ｃｏ－Ｒｏｕｔｉｎｅ）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 Ａｃｔ＿********

Ｔ ********：で始まるタスクの起動。又は再開（Ｐａｕｓｅの後は再開、Ｒｅｓｅｔの後、又は電 ON 後は、そのタスクの最初から起動）。

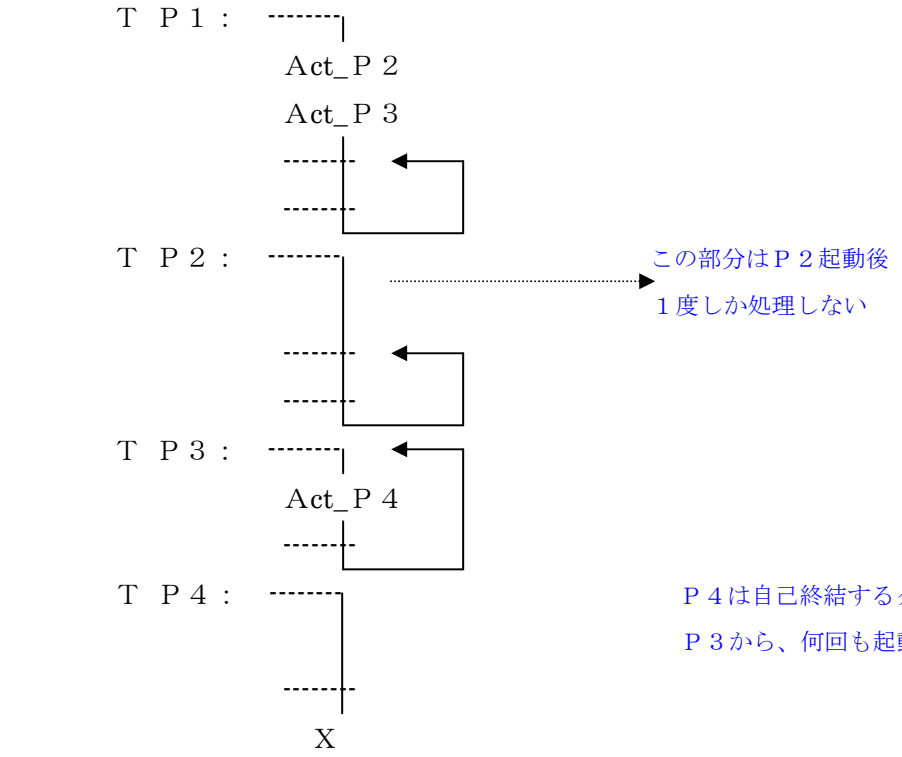
タスクが起動中ならば、無視されます。Ｐａｕｓｅの後、Ｒｅｓｅｔの後

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、ＭＥＰＳコントロラーシステムが、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。

ｅｘ、



→ 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　Ｐａｕｓｅ＿********

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

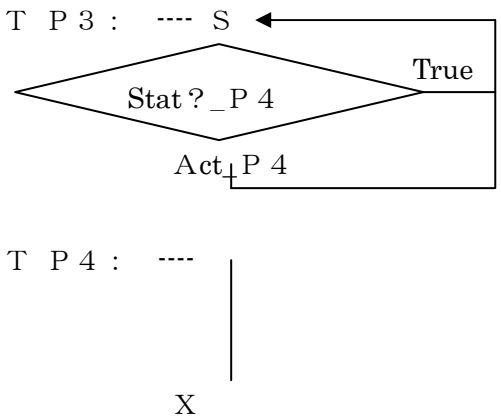
④　Ｒｅｓｅｔ＿********

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

⑤　Ｓｔａｔ？＿********

********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、



Ｐ４が自己終結するまで、Ｐ３は待って、Ｐ４が終結したら、再度、Ｐ４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、ｗａｉｔ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)。

Ｃａｌｌ 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 Ｇｏ＿********
無条件ジャンプ

② Ｇｔ＿********
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Ｎｕｍｗ命令の直後に使用します。
信号チェック命令では、ＯＮでジャンプします。
タイマーチェック命令（Ｔｍｕｐ？）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（Ｃｈｋ？）では、範囲外なら、ジャンプします。
数値入力命令（Ｎｕｍｗ）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（Ｓｔａｔ？）では、起動中又は休止中でジャンプします。

③ Ｇｆ＿********
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Ｎｕｍｗ命令の直後に使用します。
信号チェック命令では、ＯＦＦでジャンプします。
タイマーチェック命令（Ｔｍｕｐ？）では、０でジャンプします。
コード範囲チェック命令（Ｃｈｋ？）では、範囲内なら、ジャンプします。
数値入力命令（Ｎｕｍｗ）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（Ｓｔａｔ？）では、リセット状態、又は終結状態でジャンプします。

④ Ｃａｌｌ＿********
サブルーチンコール（ネスティンク数最大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

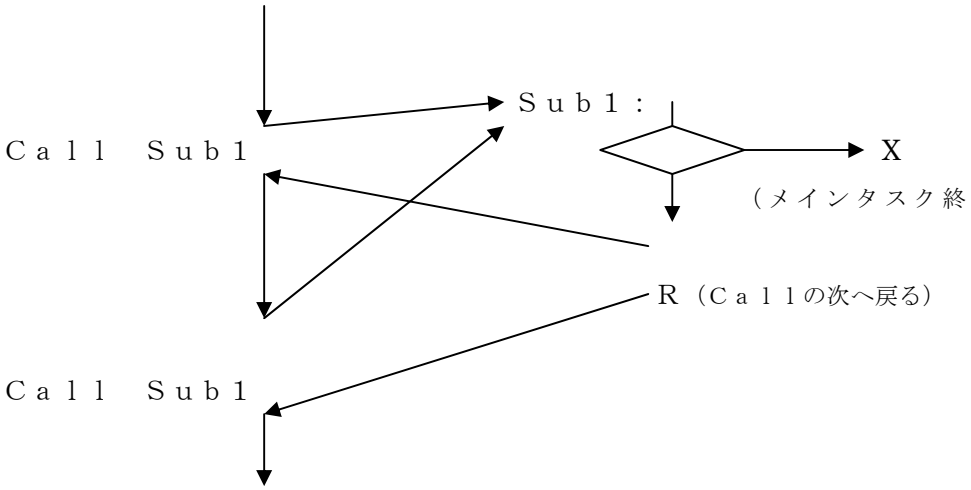
Ｃａｌｌ命令でコールされたサブルーチンからのリターン。

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、Ｃａｌｌ命令でコールされるサブルーチンの中だけ、使用します。

ｅｘ、



⑨　正整数のＩＦ文

もしも、ＩＸＸ正整数、ＩＹＹ正整数、ＤＤＤＤＤ（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。

Ｉｆ＿ｉＸＸ＝ｉＹＹ，********	一致すれば、ジャンプ
Ｉｆ＿ｉＸＸ＝ＤＤＤＤＤ，********	
Ｉｆ＿ｉＸＸ＜＞ｉＹＹ，********	不一致ならば、ジャンプ
Ｉｆ＿ｉＸＸ＜＞ＤＤＤＤＤ，********	
Ｉｆ＿ｉＸＸ＞＝ｉＹＹ，********	より大イコールならジャンプ
Ｉｆ＿ｉＸＸ＞＝ＤＤＤＤＤ，********	
Ｉｆ＿ｉＸＸ＜＝ｉＹＹ，********	より小イコールならジャンプ
Ｉｆ＿ｉＸＸ＜＝ＤＤＤＤＤ，********	
Ｉｆ＿ｉＸＸ＞ｉＹＹ，********	より大ならジャンプ
Ｉｆ＿ｉＸＸ＞ＤＤＤＤＤ，********	
Ｉｆ＿ｉＸＸ＜ｉＹＹ，********	より小ならジャンプ
Ｉｆ＿ｉＸＸ＜ＤＤＤＤＤ，********	

⑪　実数のＩＦ文

もしも、ＲＸＸ実数、ＲＹＹ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。

ｉＦ＿ｒＸＸ＝ｒＹＹ，********	一致すれば、ジャンプ
ｉＦ＿ｒＸＸ＜＞ｒＹＹ，********	不一致ならば、ジャンプ
ｉＦ＿ｒＸＸ＞＝ｒＹＹ，********	より大イコールならジャンプ
ｉＦ＿ｒＸＸ＜＝ｒＹＹ，********	より小イコールならジャンプ
ｉＦ＿ｒＸＸ＞ｒＹＹ，********	より大ならジャンプ
ｉＦ＿ｒＸＸ＜ｒＹＹ，********	より小ならジャンプ

⑫　Ｓｅｌ＿ｃＸＸ，”Ｄ，********，----------，ｈ’ＨＨ，@@@@@@@@，-------

選択ジャンプです。ＣＸＸ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　Ｏｎ__O.....

O.....の出力信号をＯＮする。

⑤　Ｏｆ__O.....

O.....の出力信号をＯＦＦする。

⑥　Ｖｎ__V.....

V.....の内部信号をＯＮする。

⑦　Ｖｆ__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無く、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、Ｉｎｗ命令で詳しく説明しますが、他の命令も同様です。

①　Ｉｎｗ__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ー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　Ｖｎｗ__V.......

V.......の内部信号のＯＮを待ちます。

Ｖｎｗ__V.......　は、無条件にＯＮを待ちます。

Ｖｎｗ__ＴＴ，********，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。

Ｖｎｗ__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　Ｖｆｗ__V.......

V.......の仮想信号のＯＦＦを待ちます。

Ｖｆｗ__V.......　は、無条件にＯＦＦを待ちます。

Ｖｆｗ__ＴＴ，********，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。

Ｖｆｗ__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　Ｏｎｗ__O.......

O.......の出力信号のＯＮを待ちます。

Ｏｎｗ__O.......　は、無条件にＯＮを待ちます。

Ｏｎｗ__ＴＴ，********，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。

Ｏｎｗ__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　Ｏｆｗ__O.......

O.......の出力信号のＯＦＦを待ちます。

Ｏｆｗ__O.......　は、無条件にＯＦＦを待ちます。

Ｏｆｗ__ＴＴ，********，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。

Ｏｆｗ__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 Ｐｉ＿ｎ，ｉＸＸ＆ｈ'ＨＨ
入力ポートｎ（０－３）から入力した８ビットにＨＨの１６進データとＡＮＤをとりｉＸＸレジスターの下位へストアします。（上位は０になる）

② Ｐｏ＿ｎ，ｉＸＸ＆ｈ'ＨＨ
出力ポートｎ（０－３）へｉＸＸレジスターの下位８ビットにＨＨの１６進データとＡＮＤをとり出力する。ＨＨ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ＨＨはポート出力の本命令の有効ビットを指定し、ＨＨ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 Ｓｔｒａｎｉ＿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 Ｓｔｒａｎｏ＿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 Ｓａｎｄ＿Ｖｚ.....，ｋ，Ｖｘ.....，ｉ，Ｖｙ.....，ｊ
Ｖｘ..... 内部信号をｉ＝0 なら、そのまま、ｉ＝1 なら反転して、
Ｖｙ..... 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、
Ｖｚ..... 内部信号へ伝達します。

④ Ｓｏｒ＿Ｖｚ.....，ｋ，Ｖｘ.....，ｉ，Ｖｙ.....，ｊ
Ｖｘ..... 内部信号をｉ＝0 なら、そのまま、ｉ＝1 なら反転して、
Ｖｙ..... 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、
Ｖｚ..... 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　正整数レジスターの演算

正整数レジスターは、０－６５５３５までの正整数です。

ａ)、四則演算

ｉＺＺ＝ＮＮＮＮＮ（0−65535）	データーセット
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 以上をーとすると、
+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 はそのままで
正の意味で i00 に 1 をセットします。

ｂ)、論理演算

Ｉ正整数レジスター０－６５５３５を０－ＦＦＦＦの１６進数とみなします。

ＡＮＤ	ＯＲ
ｉＺＺ＝ｉＸＸ＆ｉＹＹ	ｉＺＺ＝ｉＸＸ＠ｉＹＹ
ｉＺＺ＝ｉＸＸ＆ｈ’ＨＨＨＨ	ｉＺＺ＝ｉＸＸ＠ｈ’ＨＨＨＨ

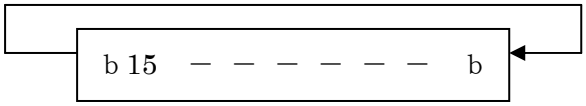
ＨＨＨＨは最大４桁の１６進数（0−FFFF）

ｃ)、ローテイトシフト

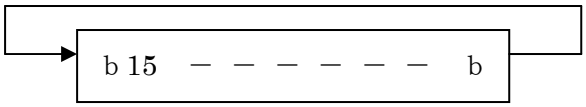
Ｉ正整数レジスターは、１６ビットのレジスターで２進数とみなします。

ｎｎは回数で１～１５

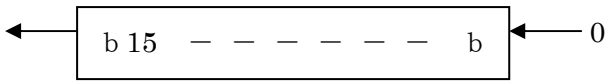
ｉＺＺ＝ＲＬ（ｉＸＸ，ｎｎ）　　ＲＬはローテイトレフト



ｉＺＺ＝ＲＲ（ｉＸＸ，ｎｎ）　　ＲＲはローテイトライト

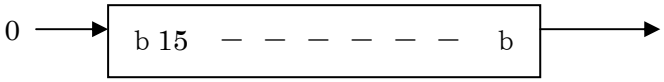


ｉＺＺ＝ＳＬ（ｉＸＸ，ｎｎ）　　ＳＬはシフトレフト



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。

ｉＺＺ＝ＳＲ（ｉＸＸ，ｎｎ）　　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

ｉＺＺ＝Ｄ（ｉＸＸ）　　２進から１０進変換

ｉＺＺ＝Ｂ（ｉＸＸ）　　１０進から２進変換

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

②　実数レジスター同志の演算

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。

ｒＺＺ＝ｒＸＸ　　コピー
ｒＺＺ＝ｒＸＸ＋ｒＹＹ
ｒＺＺ＝ｒＸＸ－ｒＹＹ
ｒＺＺ＝ｒＸＸ＊ｒＹＹ
ｒＺＺ＝ｒＸＸ／ｒＹＹ

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。

ｅｘ、

ｃＸＸ＝”ＡＢＣ　　ｃＸＸ＝”A”　ｃＸＸ＋１＝”B”　ｃＸＸ＋２＝”C”となります。アスキー文字セットです。

ｃＸＸ＝ｈ’４１４２４３　　ｃＸＸ＝h’４１　ｃＸＸ＋１＝h’４２　ｃＸＸ＝h’４３　と１６進コードセットです。

④　各レジスター間の変換

ＩＲ間　０～９９９９までの数

ｒＺＺ＝ｉＸＸ　　ＩＸＸの整数をＲＺＺの実数に変換
ｉＺＺ＝ｒＸＸ　　ＲＸＸの実数をＩＺＺの正整数に変換

ＩＣ間　０～６５５３５までの数

ｉＺＺ＝ｃＸＸ#ｎ　　ｎは１～５　ＣＸＸを先頭にｎ文字分ＩＺＺに２進数としてセットしまう。
ｃＺＺ＝ｉＸＸ#ｚｎ　　ｎは１～５　ＩＸＸの２進数をゼロサプレス無しの１０進数として、ＣＺＺを先頭にｎ文字分セットします。
ｃＺＺ＝ｉＸＸ#ｓｎ　　ｎは１～５　ＩＸＸの２進数をゼロサプレス有りの１０進数として、ＣＺＺを先頭にｎ文字分セットします。

ＩＣ間　０～ＦＦＦＦまでの１６進数

ｃＺＺ＝ｉＸＸ#ｈ　　ＩＸＸ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。

ｉＺＺ＝cＸＸ#ｈ　　ＣＸＸを先頭として、16 進数４文字をＩＺＺにセットします。

ＲＣ間　実数（浮動小数点形式）

ｒＺＺ＝ｃＸＸ#ｎ　　ｎは１～６　ＣＸＸの先頭からｎ文字をＲＺＺに変換します。
＋－．を含んでいてもかまいません。

ｅｘ、

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

ｃＺＺ＝ｒＸＸ#ｚｍ.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≦ ｍ＋ｎ ≦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。

ｅｘ、

```
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。

ｅｘ、

```
c10=”－32.5　r20＝c10#5　------　　；R20 に－32.5 が入る
Ｉf__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
```

ｃＺＺ＝ｒＸＸ#ｓｍ.ｎ　　ｓはスペースの意味でゼロサプレス付
但し１≦ ｍ＋ｎ ≦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

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

ｒＺＺ＝Ｓｑｒｔ（ｒＸＸ）　　ｒＸＸ＞０．０

② ＡのＸ乗

ｒＺＺ＝（ｒＸＸ）＊＊（ｒＹＹ）　　ｒＸＸ＞０．０

③ ＳＩＮ

ｒＺＺ＝Ｓｉｎ（ｒＸＸ）　　ｒＸＸはラジアン
２π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 ＣＯＳ

ｒＺＺ＝Ｃｏｓ（ｒＸＸ）　　ｒＸＸはラジアン
２π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 ＡＲＣ　ＴＡＮ

ｒＺＺ＝Ａｔｎ（ｒＸＸ）　　ｒＸＸはラジアン
２π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

ｒＺＺ＝Ｌｎ（ｒＸＸ）　　ｒＸＸ＞０．０

⑦ 常用対数

ｒＺＺ＝Ｌｏｇ（ｒＸＸ）　　ｒＸＸ＞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実数レジスターの重複は避けてください。

3.10　メモリー操作

① Ｓｆｃ＿ｃＺＺ，ｎ，ｃＸＸ

文字レジスターのシフト

ＣＺＺからｎ（１－１５）文字分ＣＺＺ＋ｎ－１まで、ＣＺＺ＋１からにシフトして、ＣＺＺ＋１からＣＺＺ＋ｎまでストアすると同じに、元もとのＣＸＸをＣＺＺにセットします。ＣＸＸがＣＺＺ＋ｎなら、回転する形になります。

② Ｓｆｉ＿ｉＺＺ，ｎ，ｉＸＸ

正整数レジスターのシフト

ＩＺＺからｎ（１－１５）レジスター分ＩＺＺ＋ｎ－１まで、ＩＺＺ＋１からにシフトして、ＩＺＺ＋１からＩＺＺ＋ｎまでストアすると同じに、元もとのＩＸＸをＩＺＺにセットします。ＩＸＸがＩＺＺ＋ｎなら、回転する形になります。

ｅｘ、

```
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
```

Ｉsig1 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 Ｍｖ＿ｉＹＹ，Ｄａｔａ（ｉＸＸ）

ＩＸＸの内容をインデックスとして、Ｄａｔａメモリーから ＩＹＹへセットします。

ＩＸＸの内容は、０－９９９９までとし、それ以外は何もしません。

④ Ｍｖ＿Ｄａｔａ（ｉＸＸ），ｉＹＹ

ＩＸＸの内容をインデックスとして、ＩＹＹからＤａｔａメモリーへセットします。

ＩＸＸの内容は、０－９９９９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、Ｄ００－Ｄ９９は、ＩＸＸの０－９９に対応するＤＡＴＡメモリーですが、コントローラーのサービスメニューで　Ｃｏｎｔｒｏｌ－Ｆｌａｓｈ Ｗｒｉｔｅ－Ｓｅｔ Ｄａｔａ　の書きこみで、スタティックＲＡＭ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。Ｄ００－Ｄ９９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。

ＩＸＸの１００－９９９９に対応するＤＡＴＡメモリーはデータロギング用です。

⑤　Ｄｂ＿ＨＨ，ＨＨ，------

１６進で２桁づ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、Ｃａｌｌ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エ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桁づ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

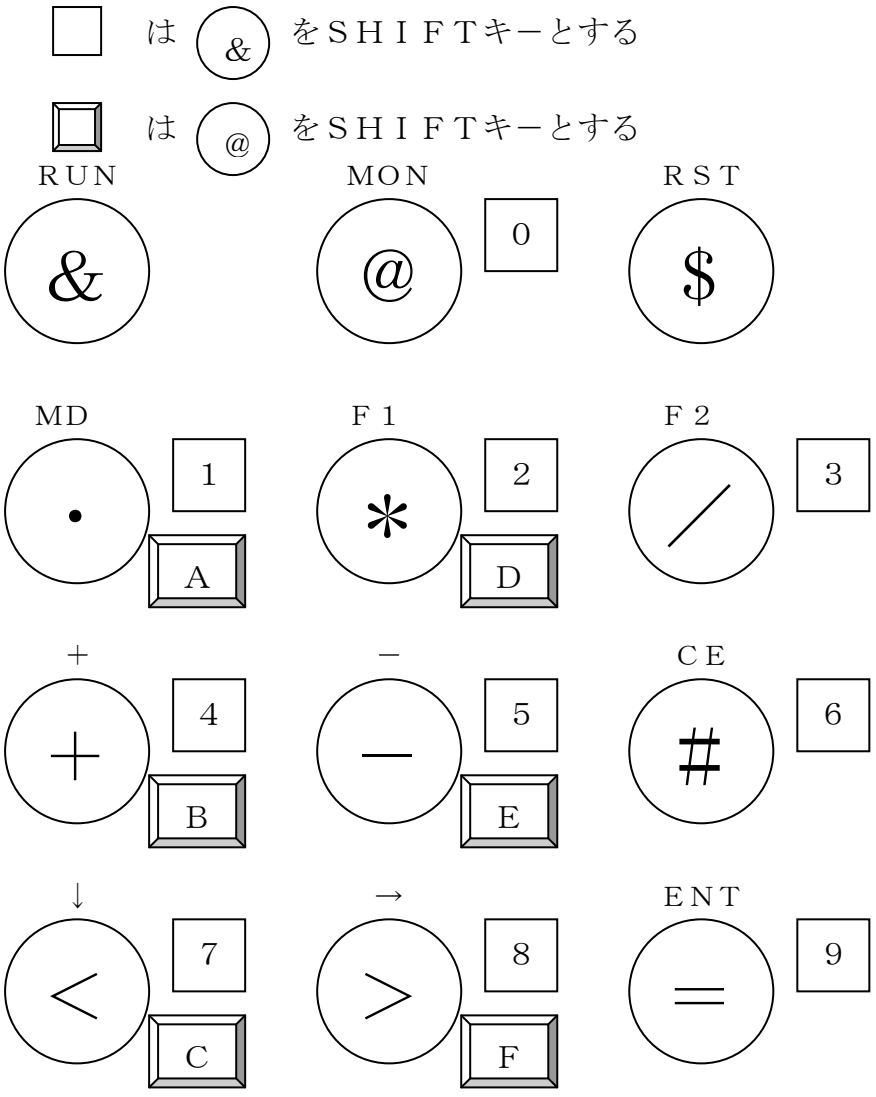
キー入力を即実行し、ｉＸＸに１２ビットセットします。

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

② Ｋｙｗ＿ｃＸＸ

キー入力をウエイトし、押されれば、その変換キーコードをＣＸＸにセットします。

＜Ｍｅｐｓ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＝＜CXX＜＝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 Ｄｓｐ＿ｉｉ，ｊ，”ABC‐‐‐
画面ｉｉ（０－１５）ｊ（０－５）座標から、 ＡＢＣ ‐‐‐ の文字列を出力します。

③ Ｄｓｐｃ＿ｉｉ，ｊ，ｎｎ，ｃＺＺ
ＣＺＺのレジスタからｎｎ文字分（ｉｉ＋ｎｎ＜１６）ＣＺＺ＋ｎｎ－１まで　画面ｉｉ（０－１５）ｊ（０－５）座標から出力します。

④ Ｄｒｖｓ＿ｉｉ，ｊ，ｎｎ
画面ｉｉ（０―１５）ｊ（０―５）座標から横列ｎｎ文字分（ｉｉ＋ｎｎ＜１６）反転文字にします

⑤ Ｎｕｍｗ＿ｉｉ，ｊ，ｎｎ，ｃＺＺ，ｄ
１０進入力
画面ｉｉ（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、ｎｎ文字分（ｉｉ＋ｎｎ＜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、ＣＺＺから、ｎｎ文字分元のデータになります。
ＥＮＴキーの場合は、ＣＺＺからｎｎ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０１２３４５６７８９－．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。
ＣＺＺからＣＺＺ＋ｎｎ－１をこの命令以前にクリアすべきです。

⑥ Ｎｕｍｗ＿ｉｉ，ｊ，ｎｎ，ｃＺＺ，ｈ
１６進入力
画面ｉｉ（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、ｎｎ文字分（ｉｉ＋ｎｎ＜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

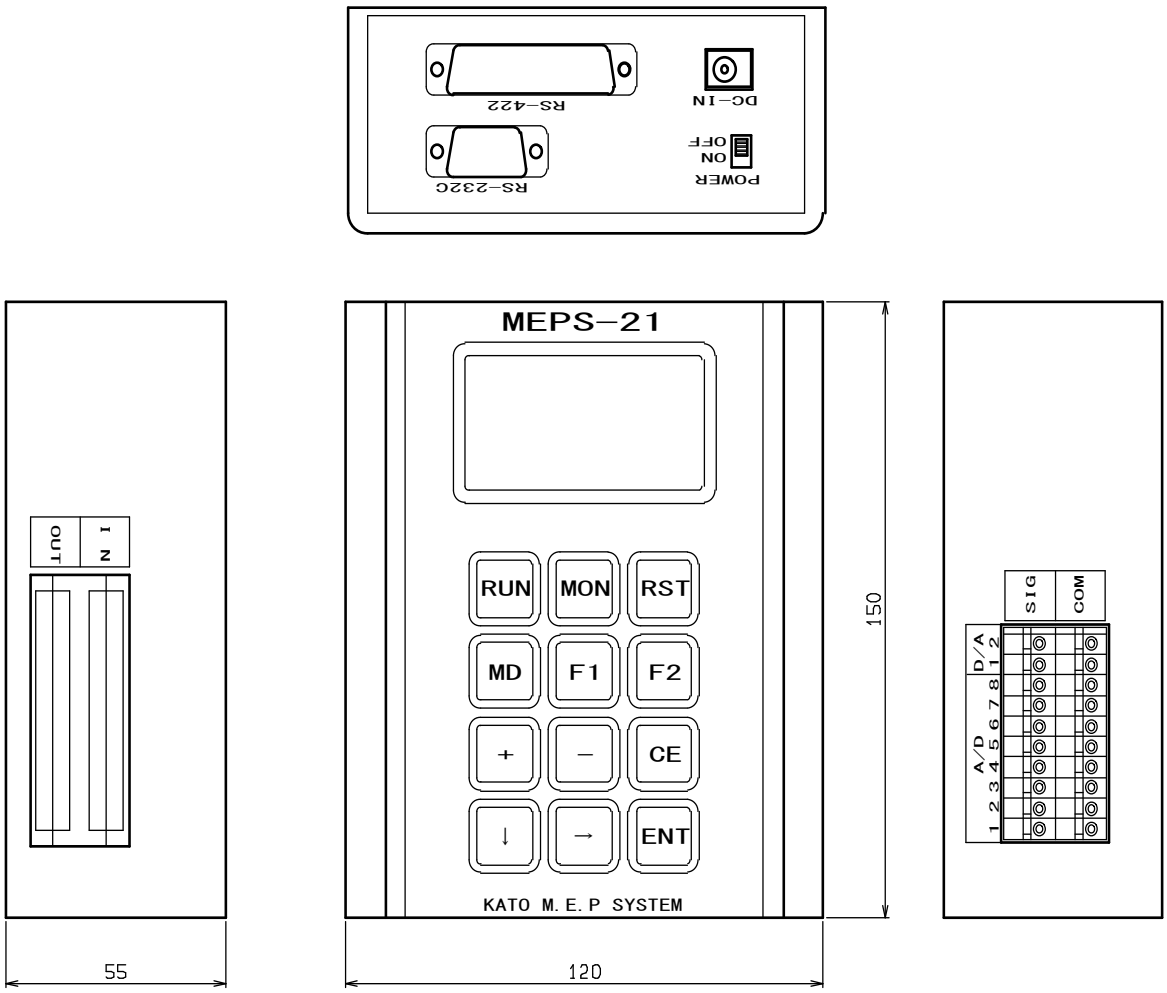
令の続行となります。
CE キーの場合は、ＣＺＺから、ｎｎ文字分元のデータになります。
ＥＮＴキーの場合は、ＣＺＺからｎｎ文字分、数値セットします。数値の入力方法に２とおりあり、キーマット図の如く、シフトキーを併用して直接数値を入力する方法と、＋－キーで０１２３４５６７８９ＡＢＣＤＥＦ をローテーションさせて設定する方法とが有ります。
パラメーターｈは１６進入力の意味です。
ＣＺＺからＣＺＺ＋ｎｎ－１をこの命令以前にクリアすべきです。

⑦ Ｒｖｓｓｅｔ＿ｉＺＺ，ｉｉ，ｊ，ｎｎ，ｉｉ１，ｊ１，ｎｎ１--------
ＩＺＺを、以下に続く ｉｉ（０－１５），ｊ（０－５）の位置からｎｎ文字分（ｉｉ＋ｎｎ＜１６）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 Ｒｖｓｉｎｃ＿ｉＺＺ，ｉｉ，ｊ，ｎｎ，ｉｉ１，ｊ１，ｎｎ１--------
Ｒｖｓｓｅｔ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。

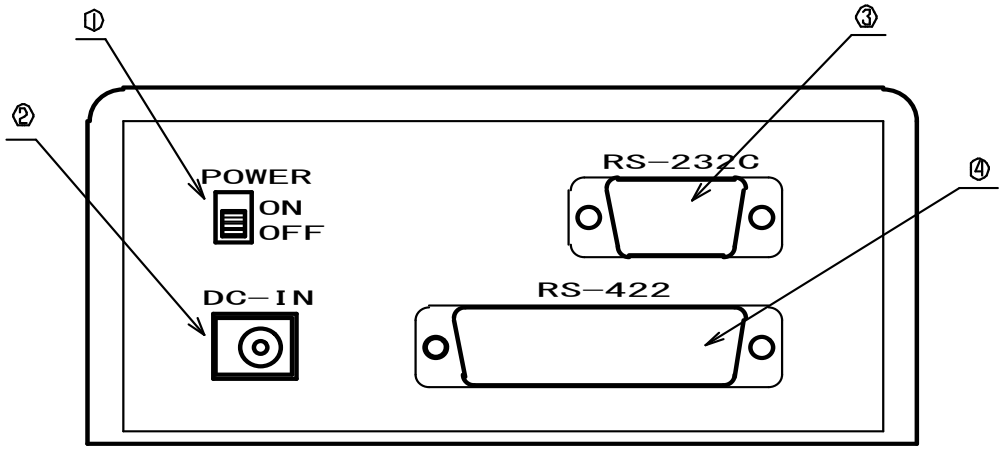
4. ＭＥＰＳ　コントローラー

４.１　外観図



＜外観図＞

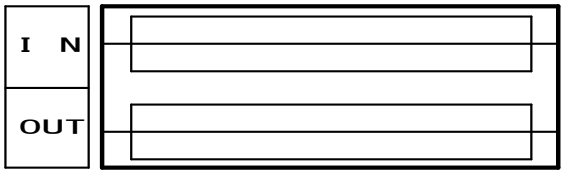
4.2　コネクター



① 電源スイッチ　　② ACｱﾀﾞﾌﾟﾀｰ入力

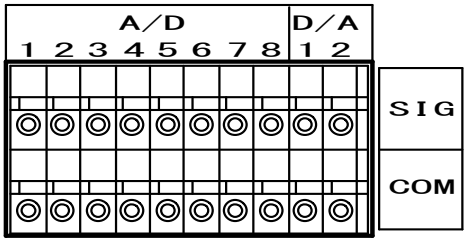
③ RS-232C　　④ RS-422 （ｵﾌﾟｼｮﾝ）

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ﾋﾟﾝ　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ﾋﾟﾝ　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：１２V／２４V　入力４．７K　５V／１２V

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

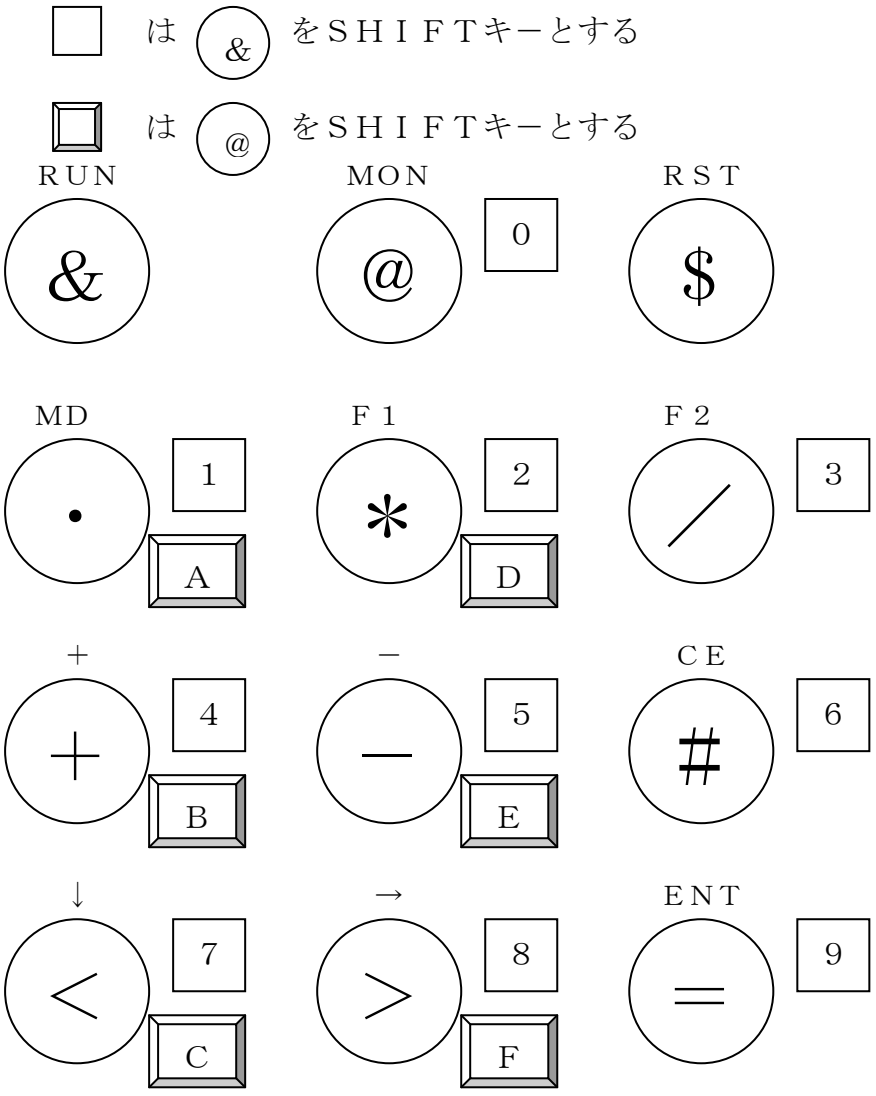
Ａ／Ｄ　１２ｂｉｔ　×　８ｃｈ　０－４．０９５　Ｖ

Ｄ／Ａ　１２ｂｉｔ　×　２ｃｈ　０－５　Ｖ

4.4　Ｍｅｐｓ 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　０－４．０９５　Ｖ　（オプション）
３、　Ｄ／Ａ　１２ｂｉｔ　×　２ｃｈ　０－５　Ｖ　（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Ｖ２５５：１ｍｓｄｕｔｙ　ｃｌｏｃｋ　サービス
　２)、内部接点Ｖ２５４：１０ｍｓｄｕｔｙ　ｃｌｏｃｋ　サービス
　３)、内部接点Ｖ２５３：１ｓｅｃｄｕｔｙ　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



5. Ｍｅｐｓ Ｃｏｎｔｒｏｌｌｅｒ Ｈｅｌｐ（MCH）画面

5.1　メニュー画面

```
＜Ｍｅｎｕ＞

１－Ｃｏｎｔｒｏｌ

２－Ｓｅｔ

３－Ｃｈｅｃｋ


Ｆ１：Ｋｅｙ　Ｈｅｌｐ
```

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

```
Ｒｕｎ＋Ｒｓｔ：Ｒｕｎ

Ｍｏｎ＋Ｒｓｔ：Ｍｅｎｕ

ＭＤ：Ｍｅｎｕ　Ｆ１，Ｆ２：？

　＋：Ｎｕｍ．＋　－：Ｎｕｍ．－

　↓：Ｄｏｗｎ　　→：Ｒｉｇｈｔ

ＣＥ：Ｃｌｒ　ＥＮＴ：Ｅｎｔｒｙ
```

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（ＭＣＨ）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

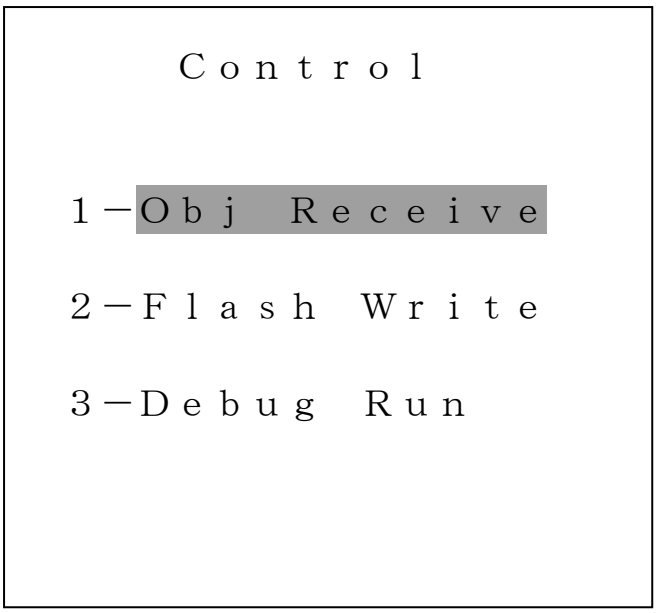
5、＋、－ キーは数値入力のＵＰ／ＤＯＷＮに使用します。

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



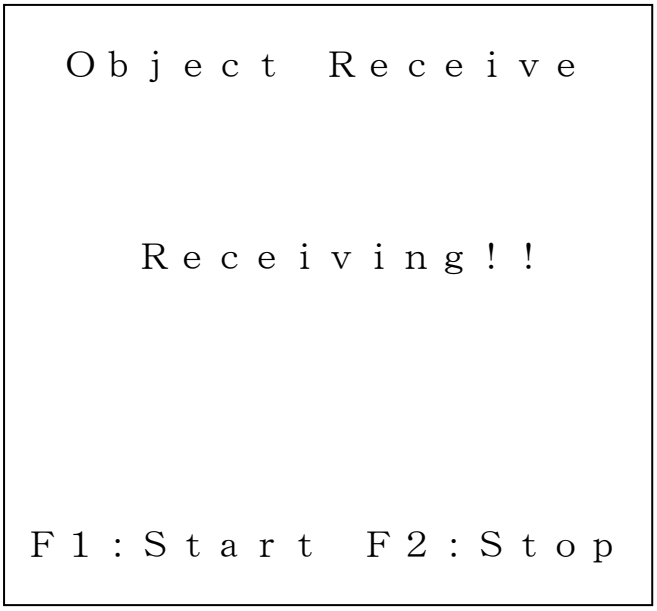
1、オブジェクトコード受信

2、フラッシュメモリー書きこみ

3、デバッグ開始

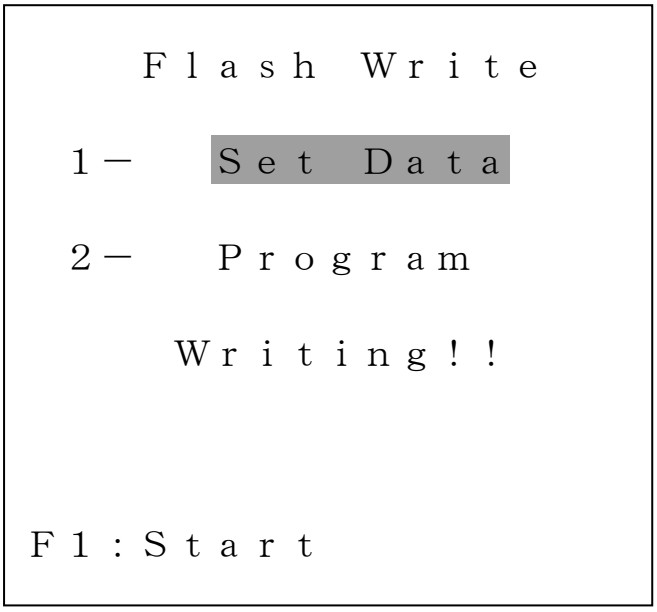
↓キーの選択とＥＮＴキーにて、各画面へと推移します。

5.4　オブジェクトコード受信画面



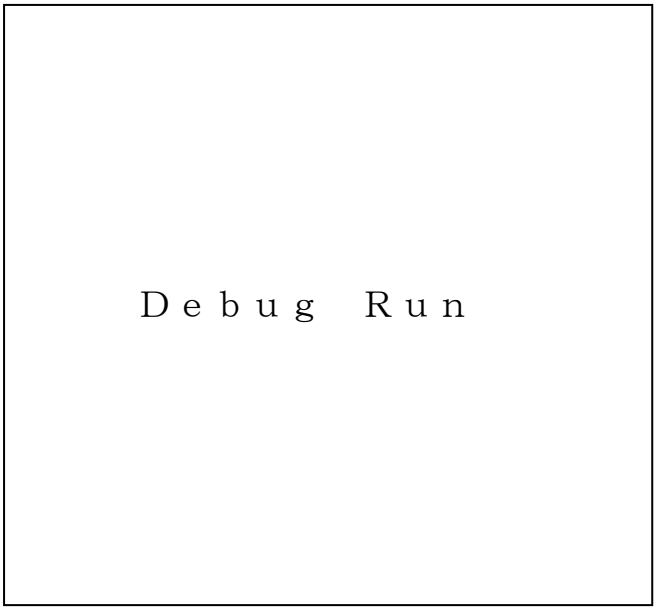
１、Ｍｅｐｓコマンドジェネレーターによって、ＰＣからオブジェクトコードを受信します。
２、Ｆ１キーを押しますと、真ん中にＲｅｃｅｉｖｉｎｇ！！を表示します。
ＰＣからの転送を開始してください。
終了、又はＦ２キーで中断すれば、Ｒｅｃｅｉｖｅｅｎｄ を表示して、終了します。
３、MDキーでメニュー画面です。

5.5　フラッシュメモリー書き込み画面



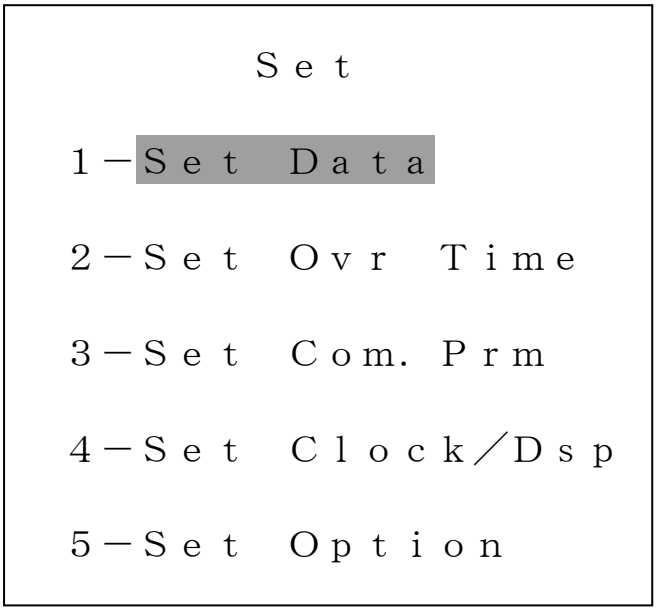
１、設定データ書き込みは、データメモリー００－９９とタイマー定数Ｎｏ００－９９の値、各種コントローラー条件を書き込みます。
２、プログラム書き込みは、ＰＣからのオブジェクトコード受信後、書き込みます。
３、↓キーで項目指定し、Ｆ１キーで 書き込み開始し、Ｗｒｉｔｉｎｇ！！を表示します。
４、MDキーでメニュー画面です。

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のデータメモリーの設定
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

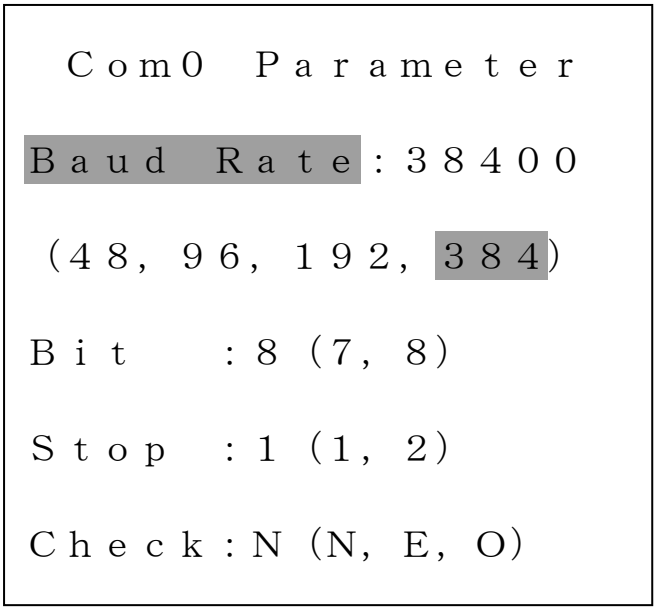
１、Ｆ１キーでＮｏ設定モード
２、Ｆ２キーでデータ定数設定モード
３、設定モードでは、数値が点滅→キー で、桁送り
４、ＥＮＴキーは確定、ＣＥキーでは元のデータにもどります。
５、MDキーでメニュー画面です。

5.9　タイマー値設定画面

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。

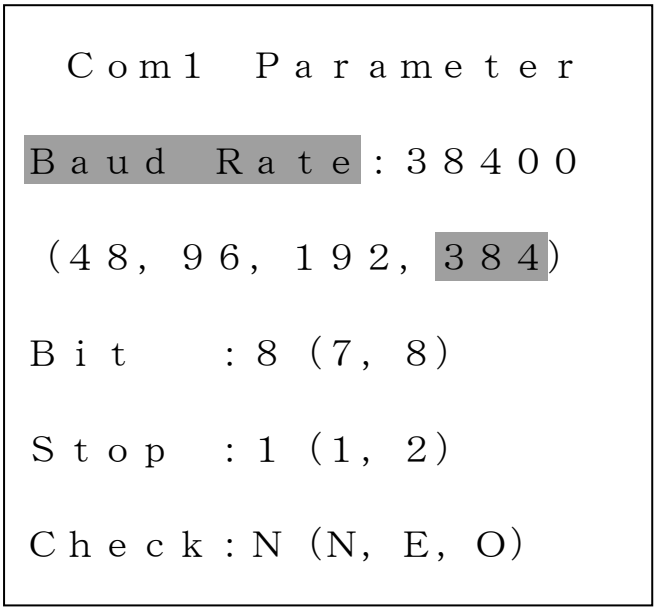
5.10　通信パラメーター設定画面

０ｃｈ



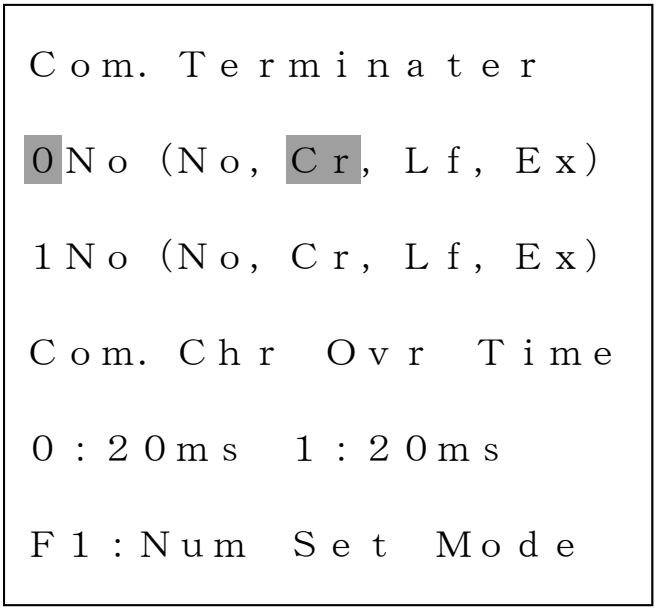
１、↓キーでボーレイト、
　　ビット長、ストップビット
　　パリティチェック選択
２、→キーで各パラメーター
　　選択、ＥＮＴキーで確定し
　　ます。
３、ＭＤキーでＣＯＭ１画面です。
４、ＣｈｅｃｋのＮは無し、Ｅ
　　は偶数、Ｏは奇数パリティ
　　チェック

１ｃｈ



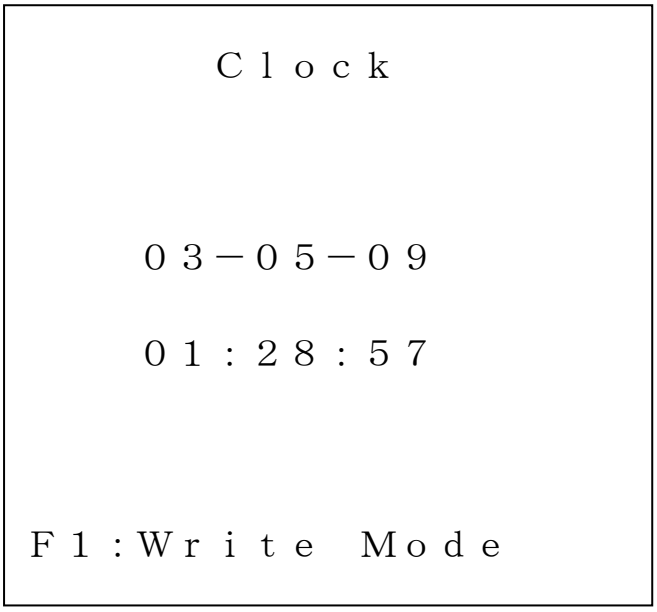
１、↓キーでボーレイト、
　　ビット長、ストップビット
　　パリティチェック選択
２、→キーで各パラメーター
　　選択、ＥＮＴキーで確定し
　　ます。
３、ＭＤキーでその他の通信パ
　　ラメーター画面です。
４、ＣｈｅｃｋのＮは無し、Ｅ
　　は偶数、Ｏは奇数パリティ
　　チェック

5.11　その他の通信パラメーター設定画面



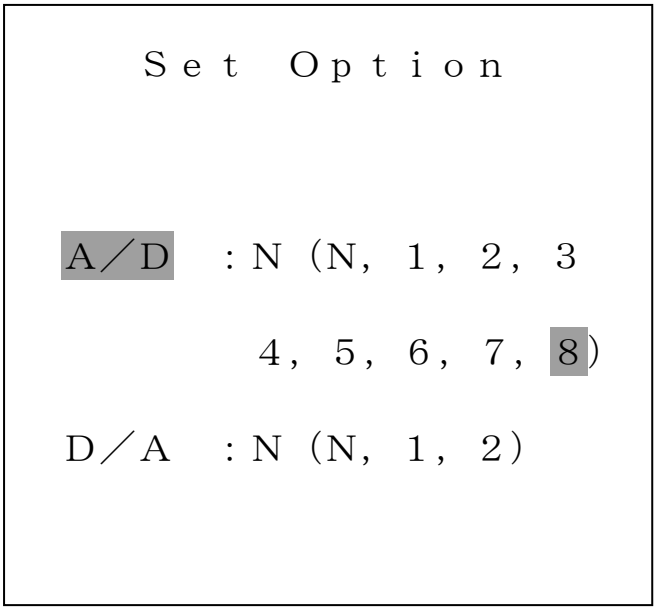
1、↓キーで０ｃｈ最終コード
　１ｃｈ最終コード、文字間
　オーバータイム０ｃｈ、
　１ｃｈを指定
2、→キーで各最終コード指定
　ＥＮＴキーで確定します。
3、文字間オーバータイムは
　Ｆ１キーで数値入力開始
　ＥＮＴキーで確定、ＣＥキー
　で元のデータに戻ります。
4、MDキーでメニュー画面
　です。

5.12　カレンダークロック設定画面



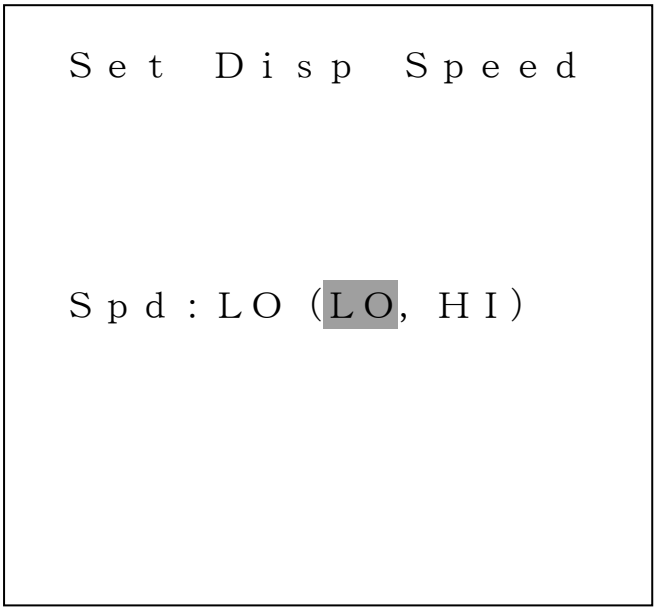
1、この画面で時刻は歩進し
　てます。
2、Ｆ１キーを押すと歩進は
　とまり、設定モードになり
　ます。→キーで桁送りし、
　数値入力し、ＥＮＴキーで
　セットし最後の秒の設定で
　書き込みをします。途中で
　ＣＥキー押せば、元の時刻
　に戻ります。
3、MDキーでメニュー画面で
　す。

5.13　オプション設定画面



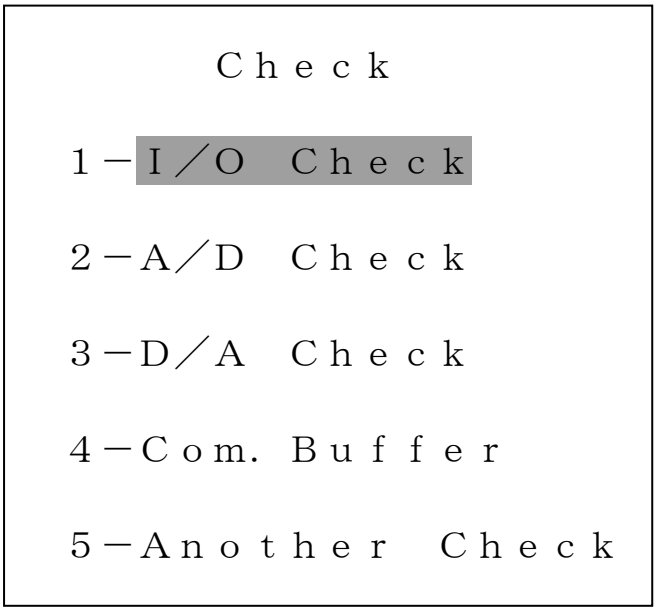
１、↓キーで、Ａ／Ｄか、
　Ｄ／Ａ を指定します。
２、オプション数を→で指定します。
３、ＥＮＴキーで確定します。
４、ＭＤキーで表示スピード画面です。

5.14　表示スピード選択画面



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

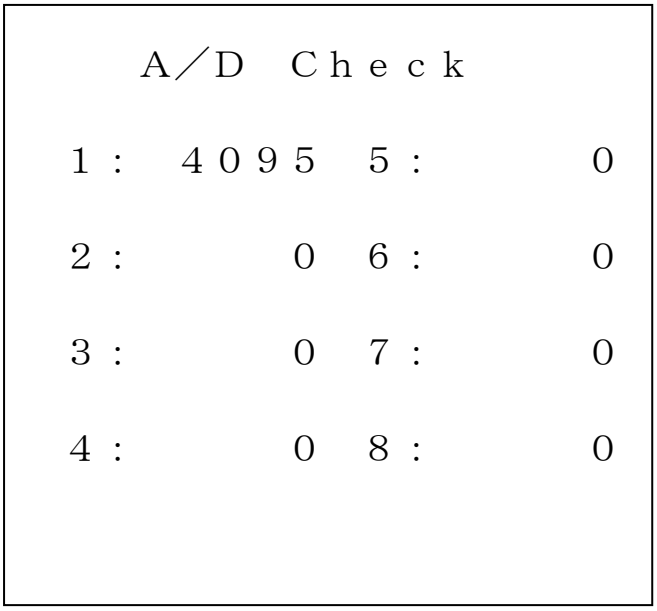


１、入出力チェック
２、アナログ入力チェック
３、アナログ出力チェック
４、通信バッファ参照
５、その他のチェック
↓キーの選択とＥＮＴキー
にて、各画面へと推移します。

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



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

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

受信バッファー

```
Ｃｏｍ０　Ｒｃｖ　Ｂｕｆｆｅｒ
０００００００００００００００
０００００００００００００００
Ｃｏｍ１　Ｒｃｖ　Ｂｕｆｆｅｒ
０００００００００００００００
０００００００００００００００
```

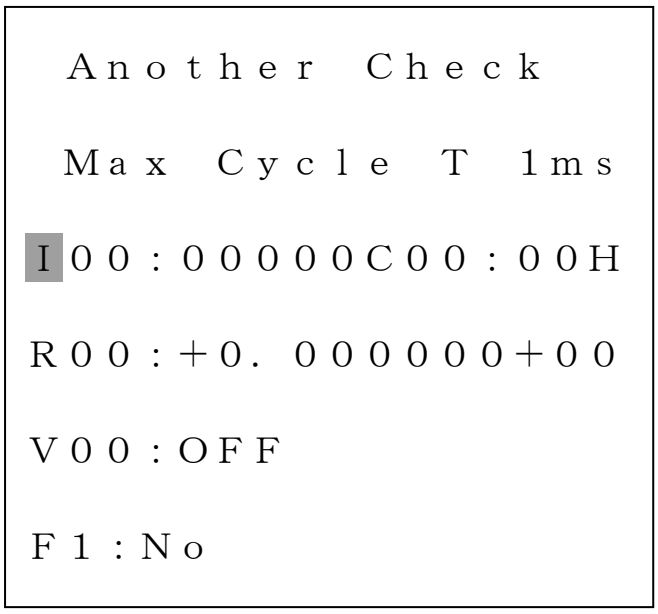
1、受信１６バイト分、０ｃｈ　１ｃｈ　表示
2、MDキーで送信バッファー表示になります。

送信バッファー

```
Ｃｏｍ０　Ｓｎｄ　Ｂｕｆｆｅｒ
　Ｆ１：Ｏｎｌｙ　８Ｂ　Ｓｅｔ
０００００００００００００００
Ｃｏｍ１　Ｓｎｄ　Ｂｕｆｆｅｒ
　Ｆ２：Ｏｎｌｙ　８Ｂ　Ｓｅｔ
０００００００００００００００
```

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０ｃｈ、１ｃｈそれぞれのバッファー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



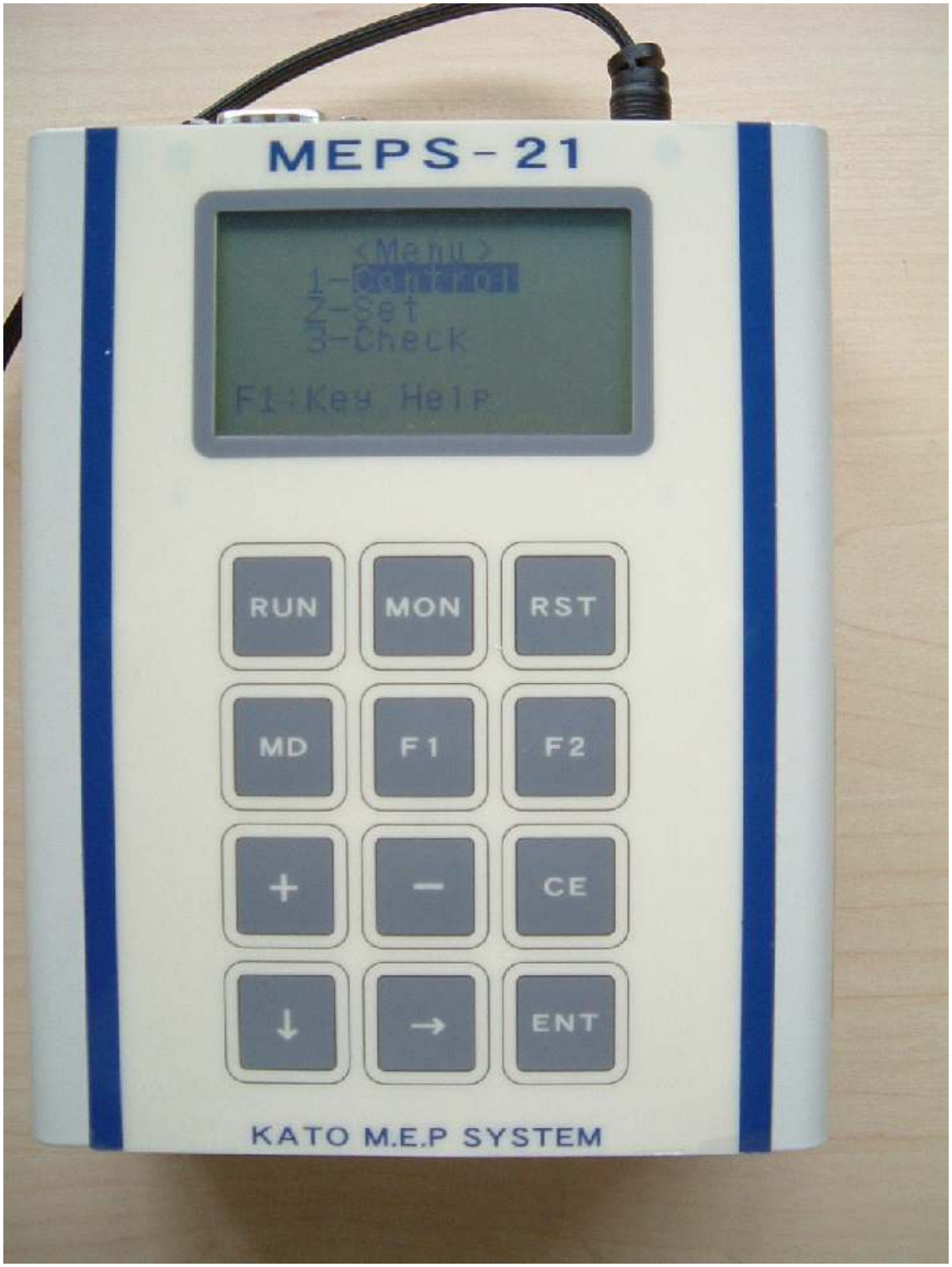
１、Max Cycle Time は、ユーザープログラムの全タスク一周時間の最大値を表示します。

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