Panasonic®

プログラマブルコントローラ
MEWNET FP3
FP3H-BASIC
命令語マニュアル

MEWNET FP3 FP3H-BASIC 命令語マニュアル
FAF−191 '94・11月

松下電工

安全に関するご注意

ケガや事故防止のため、以下のことを必ずお守りください。

据付、運転、保守、点検の前に、必ずこのマニュアルをお読みいただき、正しくご使用下さい。
機器の知識、安全の情報、その他注意事項のすべてを習熟してからご使用下さい。

このマニュアルでは、安全注意事項のレベルを「警告」と「注意」に区分しています。

⚠警告

取扱いを誤った場合に、使用者が死亡または重傷を負う危険の状態を生じることが想定される場合

●本製品の故障や外部要因による異常が発生しても、システム全体が安全側に働くように本製品の外部で安全対策を行ってください。

●燃焼性ガスの雰囲気では使用しないでください。
爆発の原因となります。

●本製品を火中に投棄しないでください。
電池や電子部品などが破裂する原因となります。

⚠注意

取扱いを誤った場合に、使用者が傷害を負うかまたは物的損害のみが発生する危険の状態が生じることが想定される場合

●異常発熱や発煙を防止するため、本製品の保証特性・性能の数値に対し余裕をもたせて使用してください。

●分解、改造はしないでください。
異常発熱や発煙の原因となります。

●通電中は端子に触れないでください。
感電のおそれがあります。

●非常停止、インターロック回路は外部で構成してください。

●電線やコネクタは確実に接続してください。
接続不十分な場合は、異常発熱や発煙の原因となります。

●製品内部に液体、可燃物、金属などの異物を入れないでください。
異常発熱や発煙の原因となります。

●電源を入れた状態では施工（接続、取り外しなど）しないでください。
感電のおそれがあります。

著作権および商標に関する記述

●Windows および WindowsNT は米国 Microsoft Corporation の米国およびその他の国における登録商標です。

●その他の会社および製品名は、各社の商標または登録商標です。

●商品改良のため、仕様、外観およびマニュアルの内容を予告なく変更することがありますので、ご了承ください。

このたびは、FP-BASICをご購入いただきましてまことにありがとうございます。

FP-BASICは市販のパソコン（NEC・PC9801シリーズおよびEPSON・PC286/386/486シリーズ）を使用して、弊社プログラマブルコントローラのプログラミングを行うための専用ソフトウエアです。

この「FP3-BASICリファレンスマニュアル」は、FP3-BASICのコマンド、命令、関数について詳細を説明しています。

なお、ご使用にあたってはこのマニュアルの他に、次のマニュアルを参照してください。

●プログラマブルコントローラについて
「FP3H-BASICタイプハード導入マニュアル」
●プログラミングについて
「FP-BASICユーザーズマニュアル」

目次

はじめてご使用になる前に ご注意いただきたいこと

制限事項

(1) FP-BASICのすべての機能が使用できるのは、FP3H-BASICタイプおよびVer2.0以降のFP3-BASIC編集ソフトを使用した場合に限定されます。

(2) FP-BASICには、MS-DOSのシステムは含まれていません。別途「NEC日本語MS-DOS (Ver.3.10以降)」を用意してください。

記述上のご注意

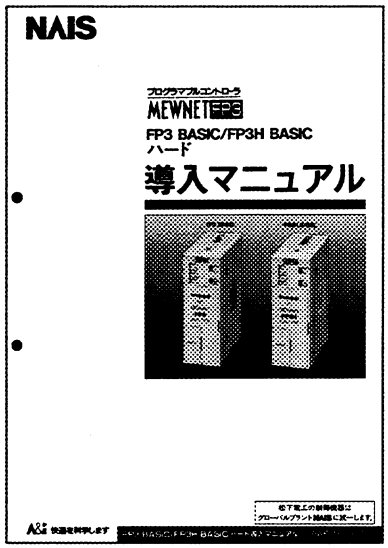
●本文中では「FP3H-BASICタイプ」「BASICタイプ」と記述している場合、または単に「CPU」と記述している場合、特に断らない限り弊社プログラマブルコントローラFP3H-BASICタイプおよびFP3-BASICタイプを意味します。

●本文中で「MS-DOS」と記述している場合は、特に断りのない限り、「NEC日本語MS-DOS (Ver.3.10以降)」を意味します。

マニュアルの種類と内容

各マニュアルの内容

FP3H-BASICタイプハード導入マニュアル

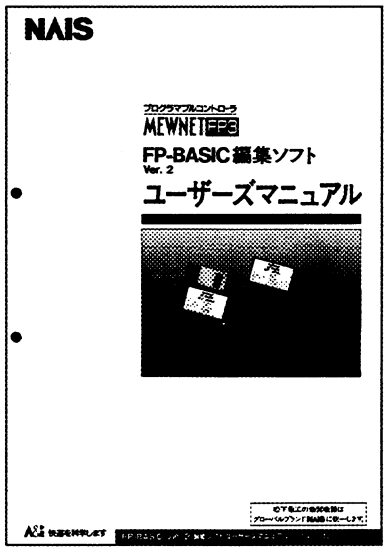


FP3H-BASICタイプの特長、システム概要、機能説明、仕様、設置、配線、運転、および保守について説明しています。

FP3H−BASIC命令語マニュアル

FP3H-BASICで使えるコマンド、命令、関数の詳しい解説をすべて収録しています。ユーザーズマニュアルと合わせてご覧ください。

FP-BASICユーザーズマニュアル



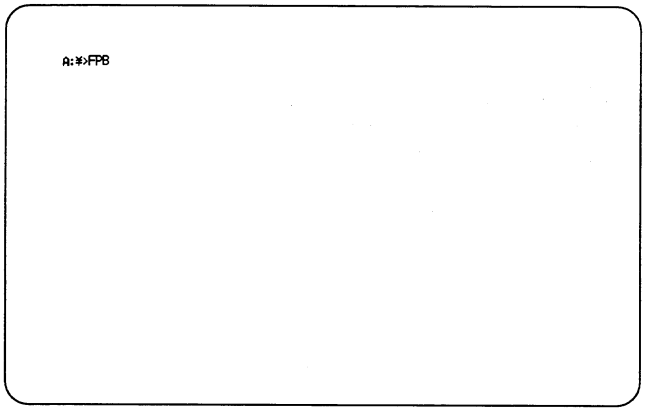
FP3H-BASICタイプのプログラミングツールであるFP-BASICに基本的な操作方法について説明しています。

1章 FP-BASICの基本ルール

1-1 FP-BASICの起動と終了

(1)FP-BASICの起動

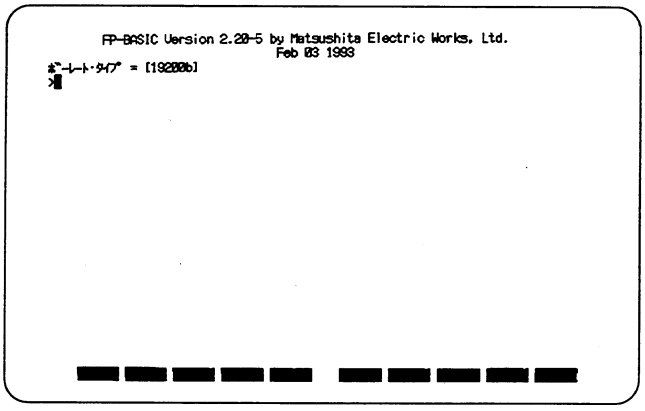
FP-BASICを起動するにはMS-DOSのコマンドラインから Ⓕ Ⓟ Ⓑ ↵ と入力します。



注意:FP-BASICの起動時にコマンドオプションが必要な場合があります。Ⓕ Ⓟ Ⓑ の後にコマンドオプションを入力してください。コマンドオプションの入力方法はは「起動時のコマンドオプションについて」を参照してください。。

FP-BASICの起動

FP-BASICの起動画面は次のとおりです。

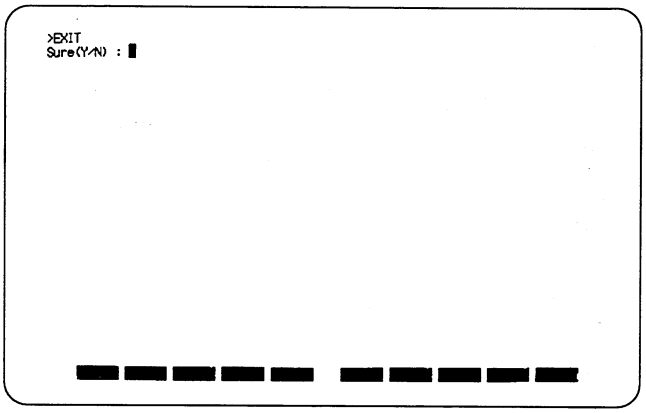


「>」は、プロンプトと呼び、コンピュータがユーザからの命令入力を待っていることを示します。
「▮」は、カーソルといい、キーボードから入力された文字がここに表示されます。

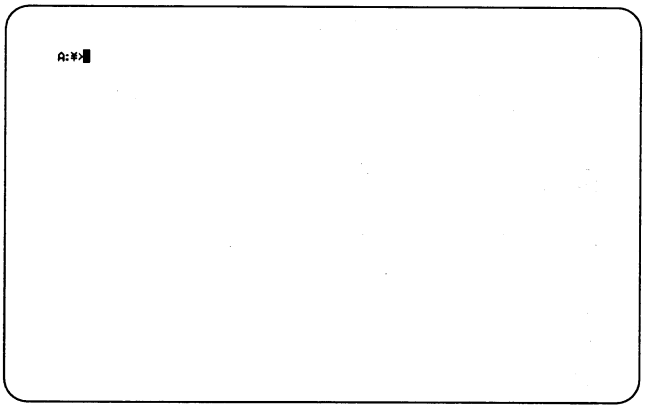
FP-BASICの起動と、終了方法について説明します。
FP-BASICのインストールの方法については、ユーザーズマニュアルを参照ください。

2)FP-BASICの終了

FP-BASICを終了するには、MS-DOSのコマンドラインから Ⓔ Ⓧ Ⓘ Ⓣ ↵ と入力します。



プログラム編集中の場合、確認メッセージが表示されますので、そのまま終了する場合は Ⓨ を入力します。
(Ⓝ を入力すると、プログラムの編集に戻ります。)
FP-BASICが終了し、MS-DOSのプロンプトが表示されます。



MS-DOSのコマンドレベルに移ります

MS-DOSの入力待ち(「A>」が表示されている状態)で、Ⓕ Ⓟ Ⓑ と入力して、↵ キーを押すと、再びFP-BASICが起動されます。

1-2
FP-BASIC起動時のコマンドオプションについて

■起動時のコマンドオプション

FP-BASICは起動時にコマンドオプションを付けることで、(1)～(9)の起動方法が可能になります。

(1)FPBで起動(オプション無し)した場合。
INS キーを押すことにより、カーソル位置にスペースを挿入します。
通信ボーレートが19200bpsになります。

(2)FPB␣/iで起動した場合。
INS キーを押すことにより、上書きモードと挿入モードの切り替えができます。
挿入モード時には、カーソルの形状が"▮"から"■"に変わります。

(3)FPB␣/Iで起動した場合。
(2)と同じ動作をしますが、カーソルの形状は"▮"の形状のまま変わりません。

(4)FPB␣/9で起動した場合。
通信ボーレートが9600bpsになります。(/9をつけない場合の通信ボーレートは19200bpsです。)
注意:
パソコンの機種によっては19200bpsで通信できないものがありますので、その場合は、"/9"をつけて起動してください。

(5)FPB␣/Hで起動した場合。
Ver2.0未満のFP3-BASIC CPU(AFP3251等)とFP1-BASIC(AFP10216C等)に対応します。
注意:
FP1-BASICは/9も付けて起動してください。
記述例:FPB␣/H␣/9

(6)FPB␣/Uで起動した場合。(Ver2.3以降)
起動時にPC本体よりプログラムをアップロードします。
注意:
LINKしたファイルはアップロードできません。

(7)FPB␣/Sで起動した場合。(Ver2.3以降)
作成中のプログラムをSAVEせずにEXITした場合、SYSTMP.PRGというファイル名でSAVEされます。
次回/Sを付けて起動するとSYSTMP.PRGをロードします。

(7)FPB␣/F〈ファイル名〉で起動した場合。
(Ver2.3以降)
KEYコマンドによりキー割付された内容がEXITする時にFKEY.DEFというファイル名で保存されます。そのファイル名を変更し保存しておき、次回起動時に/F〈ファイル名〉で変更、保存したファイル名を指定すると、保存しておいたキー割付で起動されます。/F〈ファイル名〉を付けない場合は通常のキー割付で起動されます。

(9)FPB␣/B〈ファイル名〉で起動した場合。
(Ver2.3以降)
〈ファイル名〉で指定されたバッチファイルを起動時に実行します。
バッチファイルの内容はFP-BASICのオンラインコマンド、オフラインコマンドにしてください。
注意:
バッチファイルはFP-BASIC上では作成できません。他のエディタ(MIFES等)で作成してください。

1-3
コマンドラインコンパイル機能

FP-BASIC Ver2.3以降で下記のコマンドがMS-DOSのコマンドラインから使用できます。

●COMPILE

概要

ソースプログラムをオブジェクトプログラムに変換します。

書式

A:¥>FPB␣ [/H␣]COMPILE¥"[〈ドライブ番号〉:][〈パス名〉]〈ファイル名〉¥"[,H][,L]

詳細については命令語の説明をご覧ください。

●LINK

概要

分割コンパイルのオブジェクトをリンクし、実行オブジェクトを作成します。

書式

A:¥>FPB␣ [/H␣]LINK␣〈分割オブジェクトファイル名〉[+〈分割オブジェクトファイル名〉・・・],〈実行オブジェクトファイル名〉[,H]

詳細については命令語の説明をご覧ください。

左記コマンドが終了する時にリターンコード（正常終了時0，エラー発生時1）を返してきます。これはバッチファイル等で利用できます。

〈バッチファイル例〉

```
ST:
FPB COMPILE¥"TEST 1¥",L
IF ERRORLEVEL=1 GOTO ERR_COM
FPB COMPILE¥"TEST 2¥",L
IF ERRORLEVEL=1 GOTO ERR_COM
FPB LINK TEST1+TEST2,TEST
IF ERRORLEVEL=1 GOTO ERR_LNK
ECHO **正常終了**
GOTO END
ERR_COM:
ECHO **COMPILEエラー**
GOTO END
ERR_LNK:
ECHO **LINKエラー**
END:
```

1-4
キー操作の概要

キー	キーの名称	機能
↑ → ← ↓	カーソルキー	カーソルが上下左右に1文字分移動します。 行頭の1桁を除いて、文字が表示されていない位置には移動しません。 最上行・最下行で↑↓を押すと画面がスクロールします。
ROLL-UP ROLL-DOWN	スクロールキー	カーソルが最上行に移動し、画面がスクロールします。 カーソルが最下行に移動し、画面がスクロールします。
HOME/CLR	ホームクリアキー	エディット画面を消去します。
TAB	タブキー	カーソルを8桁右に移動します。
INS	インサートキー	カーソル位置に空白（スペース）を1文字挿入します。 起動時のオプションで機能が変化します。 （P.9のコマンドオプションについて」を参照してください）
DEL	デリートキー	カーソルの左隣の文字を消去し、その位置にカーソルを移動します。
BS	バックスペースキー	カーソルを左に1文字分移動します。 （そこにあった文字は消去しません）
STOP	ストップキー	実行中のプログラムを停止します。 エラー解除の際などに使用しますが、変数の内容は保持されています。 （ストップキーを押した際のBASICタイプCPUの状態はパラメータメモリの設定内容により異なります）
RETURN	リターンキー	キーから入力し画面に表示させた命令を実行するキーです。 （例えばLIST（プログラムの表示）命令の場合、L I S Tとキーを押しただけでは、画面に「LIST」と表示されるだけでプログラムリストは表示されません。RETURNキーを押してはじめて命令として解釈され、プログラムリストが表示されます。）

1-5
使用できる文字と記号

FP-BASICでは、1バイト文字の「英字（大文字・小文字）」「数字」「カタカナ」と、「特殊記号」が使用できます。1バイト文字の「英字」と「記号」には、特別の意味をもつものや、あらかじめ決められた意味を持つ予約語がありますので、使用にあたっては十分に注意してください。
また、FP-BASICでは、2バイト文字の「英字（大文字・小文字）」「数字」「カタカナ」「特殊記号」、「ひらがな」「漢字」が使用できます。ただし、2バイト文字で命令語や変数を記述することはできません。
2バイト文字の使用は、文字定数とコメントに限られます。

■文字の種類

文字の種類	例
英字	A B C D E F G…… a b c d e f g……
数字	0 1 2 3 4 5 6 7 8 9
特殊記号	! " # $ % & ( ) , . / ; : [ ] \ ^ -
カタカナ	アイウエオカキ……
ひらがな	あいうえおかき……
漢字	松下電工株式会社

＊特殊記号のうち「"（ダブルクォーテーション）」は文字定数として使用できません。

■特殊記号

記号	読み方	特別の用途（使用時に注意が必要です）
−	マイナス	演算、範囲指定
:	コロン	ラベル行
;	セミコロン	マルチステートメント（複文）の区切り記号
'	シングルクォーテーション	REM（コメント）の省略型
"	ダブルクォーテーション	文字定数
,	コンマ	パラメータの区切り
	スペース	行番号、命令語、パラメータの区切り
.	ピリオド	ファイル名と拡張子の区切り
_	アンダースコア	なし
*	アスタリスク	ファイル名、拡張子のワイルドカード指定
?	クエスチョン	なし

＊特殊記号のうち「"（ダブルクォーテーション）」は文字定数として使用できません。

注意

1. ひらがな・漢字などの2バイト文字の入力方法については、使用するかな漢字変換ソフトのマニュアルをご覧ください。
2. 「_（アンダースコア）」は、変数名、ラベル名の先頭には使用できません。

1-6 予約語

FP-BASICの予約語は、以下のとおりです。
FP-BASICの変数名、ラベル名には、予約語を使用しないでください。また、予約語の直後が数字または「_アンダースコア」の文字列も、変数名、ラベル名に使用できません。なお、FP-BASICでは、英字の大文字と小文字の区別をしませんので、変数名、ラベル名の重複に注意してください。

(	CALL	END	IDREADY ()	LOG ()	POKE	RENUM	SVIO
)	CASE	ENDIF	IDRECV	LONG	POORG	RESET	SVVAL
"	CHR$ ()	ENTRY	IDRV	LOOP	POPSET	RESTORE	SW ()
+	CLEAR	EOF ()	IDRVA	LROOL ()	POREOS	RESUME	SWAP
-	CLOSE	ERR ()	IDRVH	LSHIFT ()	PORESET	RETRN	SWITCH
/	CLRLIB	EXIT	IDRVHA	MAP	POSTART	RETURN	TAN ()
<	CLRVAL	EXP ()	IDRVHAID	MAXDEV	POSTAT ()	RFORCE	TEST
<=	CLRVAR	EXT	IDRXACK	MEW$ ()	PRGSIZE	RIGHT$ ()	THEN
<>	CLS	ERN	IDRXDT	MID$	PRINT	RIOCLR	TIME
=	COLOR	FEED	IDSEND	MID$ ()	PRIM	RIOSET	TIME$
=<	COMPILE	FEND	IDTCHR	MKB$ ()	NTR	RND ()	TIME$ ()
=>	CONSOLE	FERROR ()	IDTXACK ()	MKL$ ()	PRIVATE	RROLL ()	TIME ()
>	CONT	FILES	IDTXDT	MKS$ ()	PRMPRM ()	RSHIFT ()	TIMER
>=	COS ()	FLOAT	IDTXRDY ()	MKW$ ()	PRMCLR	RSLOTRUN	TIMER ()
X_	CVB ()	FOR	IDTYPE	MOD	PRMR	SDERR ()	TMOUT
Y_	CVL ()	FORCE	IDWI	MODEMON	PRMR ()	SDRDY ()	TO
R_	CVS ()	FRE	IDWRX ()	MONCLR	PSADRS ()	SDRECV	TOFF
L_	CVW ()	FRE ()	IDWV	MONDISP	PSADSET	SDRESET	TON
WX_	CX ()	FREV	IDWVA	MONENT	PSAID ()	SDRXACK	TSTAT
WY_	CY ()	FUNCTION	IDWVH	MONTS	PSBUSY ()	SDRCDT	TSTAT
WR_	DAALM ()	GO	IDWVHA	MOVE	PSCTRL	SDSEND	TW ()
WL_	DAERR ()	GOSUB	IDWVHAID	MYINT	PSDSET	SDTCHR	TWAIT
DT_	DALMIT	GOTO	IF	MYTASK ()	PSECLR	SDTERM	UNIT ()
LD_	DALSET	HALT	IFBIN ()	NAMENEG	PSERR ()	SDTXACK ()	UPLDVAL ()
FL_	DARDY ()	HCADATA ()	INCIND ()	()	PSJOBNO	SDTXDT	VARIABLE
ABS ()	DASET	HCFSET	INH ()	NEXT	PSLOAD	SDTXRDY ()	VER
ACCEL	DATA	HCISET	INL ()	NGX	PSORGH	SDWRX ()	VERINIT
ADALM ()	DATA$	HCOUTEN	INPUT	NGY	PRORGS	SEGT ()	WAIT
ADASET	DATE$	HCPSET	INPUT$ ()	NOP	PSPRM	SEL	WEND
ADAVRG	DEC	HCRESET	INPUTR	NOTOCT$ ()	PSPRM ()	SELECT	WHILE
ADDATA	DECO ()	HCSTAT ()	INSTR	OFF	PSREADY ()	SELEND	WRITE
ADDATA ()	DEFAULT	HEX$ ()	INSTR ()	ON	PSSAVE	SEND	WRITE
ADERR	DIM	HOME	INT	ONERR	PSSTART	SENDB	RXO
ADLIMIT	DIST ()	HORDR	INT ()	ONSW	PSSTAT ()	SET	RXQTZE
ADLSET	DLOAD	HTEST	INTEGER	OPEN	PSSTOP	SGN ()	ROFLG ()
ADRDY ()	DO	IDAC	INV ()	OR	PSTYPE	SIN ()	¥
ADSCALE	DREAD	IDBC	INW ()	OUT	PSX	SLOT ()	_ACLMAX
ADRSET	DRV	IDCAN	KILL	OUTD	PSY	SLOTCLR	_ENCDIV
AND	DUMP	IDCLR	LDCR	OUTH	PULSE	SLOTI	_HDIR
ASC ()	DUMPC	IDCRSP ()	LDIO	OUTL	QUIT	SLOTICLR	_HSENSE
ATAN ()	DUMPINT	ID	LDVAL	OUTW	RANDMIZE	SLOTR ()	_HSPEED
ATN ()	DUMPP	IDBRID	LEFT$ ()	PALET	RANGE	SPACE$ ()	_HTELIM
BCD ()	DUMPR	IDBWID	LEN ()	PAUSE	READ	SPEED	_MNAME
BIN$ ()	DWNL	IDRID	LINE	PDEL	READR	SQR ()	_PMATCH
BIN ()	DECLR	IDW	LIST	PEEK ()	REAL	STEP	_SELAXIS
BLKMOV	ELSE	IDPRR	LOC ()	PODATA ()	RECV	STR$ ()	_SPDMAX
BLKOUT	ELSEIF	IDPRW	LOCATE	POFCHG	RECVB	STRING	_ZPHASE
BYTE	ENCO ()	IDRI	LOF ()	PIOSET	REM	SVCR	

1-7 変数

(1) 変数名

FP-BASICでは、配列を含む任意の変数を使用することができます。

変数名には、8文字までの英数字と「_(アンダースコア)」が使用できますが、先頭の1文字は必ず英字で書き始めなければならないという約束があります。なお、FP-BASICでは、英字の大文字と小文字を区別しませんので注意してください。

また、文字型の変数には、変数名の末尾に「$」を付けなければなりません。さらに、文字変数では、「$」の後に「*」と0～255数値を記述することにより、変数のサイズを任意に決めることができます。

FP-BASICの変数名には、「P」で始まる文字列および予約語は使用できません。また、予約語の直後が数字または「_(アンダースコア)」の文字列も、変数名に使用できません。

・数値変数の例

A　B1　C20　FP3　FPBASIC

・文字変数の例

A$　B1$　C20$　FP3$　FPBASIC$

A$*4　B1$*8　C20$*16　FP3$*255

(2) 変数の型と値

変数は代入される値の種類により、数値型と文字型に分けられます。さらに、数値型の変数の場合、1バイト整数型、2バイト整数型、4バイト整数型、4バイト実数型に分けられます。

4バイト整数型(LONG型)以外では、使用時に変数宣言(型宣言)をしなければなりません。なお、変数宣言をせずに変数を使用した場合、タスク間でグローバルな変数として定義されます。

注意

・文字型の変数(例:A$,FPB$)では変数の末尾に「$」記号を付けますが、数値型の変数は末尾に記号は付けません。

・値が代入される前の変数は、数値変数では「0」として、文字変数では「空の文字列(“”ヌルストリング)」として扱われます。

・変数宣言は、プログラム中の各タスクブロックの(FUNCTION～FEND)の最初にまとめて記述します(タスクブロックの途中で変数宣言をすると正しく解釈されない場合があります)。

変数宣言型	型宣言の意味	値	変数エリアの使用バイト数
BYTE A	1バイト整数型	-128～+127 &HFFFFFF80～&H0000007F (上位の0は省略可) &M0～&M7F	1バイト
INTEGER A	2バイト整数型	-32,768～+32,767 &HFFFF8000～&H00007FFF (上位の0は省略可) &M0～&M207F	2バイト
LONG A	4バイト整数型	-2,147,483,648～+2,147,483,647 &H80000000～&H7FFFFFFF (上位の0は省略可) &M0～&M1342177272F	4バイト
REAL A	4バイト実数型	精度が保証される0に最も遠い値: -1.2E+38～+1.2E+38 精度が保証される0に最も近い値: -1.2E+38～+1.2E-38	4バイト
STRING A$	文字型	0～80バイトの文字列	81バイト
STRING A$*n	文字型	nバイトの文字列 (1≦n≦255)	n+1バイト

詳細はそれぞれの命令語の説明をご覧ください。

注意

FP-BASICの整数は、符号付き整数です。ただし、2・8・16進数表記のバイナリデータの先頭ビットは符号として解釈されません。たとえば、INL()、INH()関数でI/Oの状態を取り出して1バイト型変数に代入すると、最上位ビットが1の場合エラーになります。ただし、IND()関数の値を4バイト型整数に代入する場合はエラーになりません。

したがって、整数型の変数に2・8・16進数表記のバイナリデータを代入する場合は、十分に注意してください。

(3) グローバル変数とローカル変数

FP-BASICの変数は、タスク間でグローバルな変数として使用することも、ローカルな変数として使用することもできます。ただし、ローカル変数に使用した変数名と、グローバル変数に使用した変数名が重複しないように注意してください。なお、ローカル変数では異なるタスクで同一の変数名を使用することができます。

●何も付けずに型宣言した場合は、タスク間でグローバルな変数として使用されます。

```
210 BYTE A
```

●PRIVATE型の型宣言命令を使用した場合は、タスク間でローカルな変数として使用されます。

```
210 PRIVATE BYTE A
```

また、分割コンパイル時にはENTRY,EXTERNの宣言により、ソースファイル間でもグローバルな変数として使用することもできます。
詳細については、ENTRY,EXTERN命令の説明を参照してください。

(4) 配列

FP-BASICでは、任意の配列を変数として使用することができます。配列の次元は、最大5次元まで可能ですが、実際にはメモリ容量により制限されます。
配列名には、8文字までの英数字が使用でき、先頭の1文字は必ず英字で書き始めなければならないという約束があります（変数名と同じ規則）。また、文字列配列は各要素のサイズ（バイト長）を指定できます。
同じような変数をいくつも使用する場合、同じ変数名を付け、たとえば「A(10)」のように配列中の要素の番号を添え字として指示するようにします。このような変数を配列変数と呼びます。配列変数の添え字は、「0」から始まります。したがって、「BYTE A(10)」で宣言した配列変数の添え字は「0」～「10」までです。

●数値配列

```
210 PRIVATE BYTE A(10)
220 PRIVATE BYTE B(5,7)
```

●文字列配列

```
210 PRIVATE STRING A$(10)
220 PRIVATE STRING B$*8(4,2)
```

(5) 型変換

FP-BASICでは、異なった型の数値を代入または演算するとき、次の法則にしたがって型変換を行います。

●代入文による型変換

代入される変数の型に変換されます。ただし、文字型と数値型の変換はできません。

整数型の変数に実数型の値を代入
(小数点以下を四捨五入します)

```
>LIST
100 FUNCTION MAIN
110   BYTE A
120  A=24.51
130 PRINT A
140 FEND
>DWNLD
>XQT
25
>
```

●実数型の変数に有効桁数以上の実数を代入
(有効桁数の範囲に丸められます)

```
>LIST
100 FUNCTION MAIN
110   REAL A
120  A=13.45789123456
130 PRINT A
140 FEND
>DWNLD
>XQT
13.45789
>
```

●算術演算式による型変換

精度の異なる数値演算は、精度の低い値を精度の高い方に変換します。

```
>LIST
100 FUNCTION MAIN
110   BYTE A
120   REAL B
130 A=23
140 B=2.34
150 PRINT (A*B)
160 FEND
>DWNLD
>XQT
53.82
>
```

●論理演算式 (ビット演算式) による型変換

すべて整数に変換 (小数点を四捨五入) して演算します。

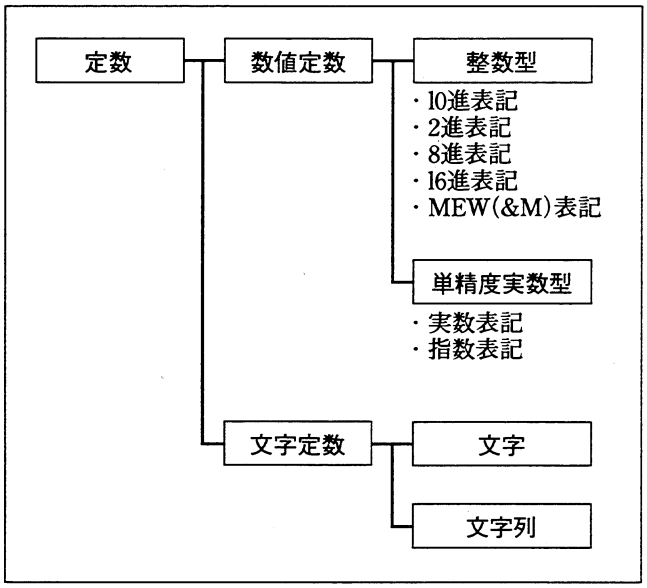
```
>PRINT (2 OR 2)
2
PRINT  (2  OR  2.499)
2
PRINT  (2  OR  2.5)
3
>
```

1-8 定数

(1) 定数の種類

定数は、値の決まっているデータです。たとえば、「1」、「32」という数字や「A」、「ABC」という文字は、それ自身の値を持つ定数です。

定数の種類は、大きく分けて、数値定数と、文字定数（文字列）があります。数値定数には、データの「型」による分類と、「表記」上の分類があります。



(2) 数値定数

数値定数は、通常の10進数での表記の他、2進数、8進数、16進数での表記で記述することができます。さらにFP-BASICでは、MEW表記として「&M」による表記を使用して数値を記述することができます。MEW表記は、末尾1桁だけが16進数表記（0～F）で、その他は10進数表記です。主に、I/O番号の指定に使用します。

・10進表記の例	1　123	6210
・2進表記の例	&B1	&b1011011
・8進表記の例	&O123	&o722
・16進表記の例	&H123	&h12F
・MEW表記の例	&M123	&m12F

注意
2進、8進、16進、MEW表記の数値定数は4バイト整数として扱います。
これらの表記の数値定数を変数に代入するときはオーバーフローに注意してください。

〈例1〉
2バイト整数A（INTEGER A）に32,767を代入する場合の表記
A=32767
A=＆H00007FFF　（上記の0は省略可）

〈例2〉
2バイト整数A（INTEGER A）に－1を代入する場合の表記
A=－1
A=＆HFFFFFFFF

〈例3〉
2バイト整数A（INTEGER A）に－32,768を代入する場合の表記
A=－32768
A=＆HFFFF8000

(3) 文字定数

FP-BASICでは、80バイトまでの文字列を文字定数として使用できます。

プログラム中に文字定数を記述する場合、必ず「"（ダブルクォーテーション）」で囲みます。なお、「"（ダブルクォーテーション）」そのものは、文字定数には使用できません。

・文字定数の例　"ABC"
"123"
"FP-BASIC"
"1+2=3"
"ｱｲｳｴｵ"
"松下電工"

1-9
式と演算子

(1) 式の種類

FP-BASICでは、算術演算、論理演算（ビット演算）、比較演算ができます。
式は、定数、変数、関数および式自体を演算子で結合したものであり、演算子には以下のような特別な記号または文字列を使用します。
FP-BASICの論理演算式（ビット演算式）は、2つの数値の対応するビットごとに、指定された論理演算を実行します。
FP-BASICの比較演算式は、2つの数値の大きさを比較して、式が真であるとき「−1」を、式が偽であるとき「0」を、それぞれ式の値として返します。

演算子			例	変数Aに代入される結果
四則演算	＋ − ＊ ／ MOD	加算 減算 乗算 除算 除算の余り	A＝1＋1 A＝2−1 A＝3＊2 A＝4／2 A＝5 MOD 3	2 1 6 2 2
論理演算 （ビット演算）	OR AND XOR	論理和 論理積 排他的論理和	A＝1 OR 2 A＝1 AND 2 A＝1 XOR 2	3 0 2
比較演算	＝ <＞ ＜ ＞ <＝ ＞＝	等しい 等しくない より大きい より小さい 以上 以下	A＝（B＝C） A＝（B＜＞C） A＝（B＜C） A＝（B＞C） A＝（B＜＝C） A＝（B＞＝C）	真のとき−1 偽のとき　0

(2) 演算式の優先順位

演算子の優先順位は、以下のとおりです。「() カッコ」でくくられた式および関数は最優先で演算を行います。

優先順位

1	()	「() カッコ」で囲まれた式
2	関数	数値関数・文字関数
3	−	マイナス記号
5	MOD	除算の余り
6	＋−	加算・減算
7	＝〈〉	比較演算
8	NOT	否定
9	AND	論理積
10	OR	論理和
11	XOR	排他的論理和

2章 FP-BASICの命令語一覧

2-1
命令語の説明の見方

●コマンド・命令・関数の説明は以下の書式で統一されています。

FIND

オンラインコマンド　オフラインコマンド

概要

編集中のプログラムから文字列を検索します。

書式

FIND〈文字列〉[,[〈検索開始行番号〉][,〈検索終了行番号]]

説明

プログラム中から指定された〈文字列〉を検索し、その行を表示します。
〈検索開始行番号〉と〈検索終了行番号〉が指定されたときはその範囲で検索します。
〈検索開始行番号〉だけが指定されたときは、その行からプログラムの最後までの範囲で検索します。
〈検索終了行番号〉だけが指定されたときは、プログラムの先頭からその行までの範囲で検索します。
〈文字列〉には、ワイルドカード（「？」「＊」）が指定できます。
「？」クエスチョンマークは任意の一文字に対応します。
「＊」アスタリスクは0個以上の文字と対応します。

記述例

```
FIND WAIT
```

用例

```
>LIST
100 FUNCTION SAMPLE
110 INTEGER A
120 ST:
130 FOR A=TO 3 STEP 1
140 ON Y_&M10
150 WAIT 1
160 OFF Y_&M10
170 WAIT 1
180 NEXT A
190 END
200 FEND
>FIND WAIT ……プログラム中から文字列「WAIT」を
150 WAIT 1 ……探索します。
170 WAIT 1
>
```

オンラインコマンド、オフラインコマンド:
FP-BACICの直接実行で使用できるコマンドおよび関数です。オンラインコマンドは、BASICタイプのCPUと接続されている場合にのみ使用できます。オフラインコマンドは、BASICタイプのCPUと接続されていない時でも使用できます。

ステートメント:
FP-BASICの間接実行で使用できる命令および関数です。プログラム中に記述できます。

概要:
コマンド・命令・関数の機能を説明しています。

書式:
コマンド・命令・関数の正しい書式を示します。
[ ] 内の記述は省略可能です。
{A | B} はAまたはBのどちらかを選択します。
( ) はそのままキー入力をする。
␣ はスペースを開けることを意味します。

説明:
コマンド・命令・関数の使用方法を説明します。また、それぞれの引数とオプションについて説明します。

関連命令:
関連するコマンド・命令・関数があればこの項目で示します。

記述例:
コマンド・命令・関数の記述例を示します。

用例:
コマンド・命令・関数の使用方法を具体的に示します。

●このマニュアルで使用されている記号は以下のとおりです。

2-2 命令語一覧

■FP-BASICの準備

●CPUの初期化

●パラメータメモリの設定

●スロット割り付け

●キー定義

■プログラム編集

●ソースプログラムのエディット

●ファイルのセーブとロード

●オブジェクトプログラムの作成と転送

●分割コンパイル

■プログラムの実行と停止

■デバッグ機能

●シンボルの転送と照合

●トレースモード

●テストモード

●実行中モード、ステータスの表示

●I/O、メモリ、変数のモニタ

●I/O、メモリのダンプ

●タイマ経過値の表示および設定

●高機能ユニットの割り込み設定の表示

●I/O、メモリ、その他の内容の一括表示と編集

●強制入出力

●I/O、メモリ、変数のセーブとロード

■制御命令

●実行制御構文

●割り込み制御

●割り込みユニット制御命令

●変数宣言

■一般命令

●BASIC一般命令

●ディスクファイルアクセス命令

●変数操作命令

●I/Oポートアクセス命令

●タイマ命令

●システム変数とシステム関数

■組み込み関数

●数値関数

●三角関数

●対数関数

●文字関数

●数値文字変換関数

■リモートI/Oシステムのスロット割り付け

■MEWNETデータ転送命令

■高機能ユニット制御命令

●通常スロットの高機能ユニット入出力命令

●リモートI/Oシステムの高機能ユニット入出力命令

●A/D変換ユニット専用制御命令

●D/A変換ユニット専用制御命令

●シリアルデータユニット専用制御命令

●高速力ウンタユニット専用制御命令

●パルス出力ユニット専用制御命令

●位置決めユニット専用制御命令

ABS( )

オンラインコマンド　ステートメント

概要

数値の絶対値を返します。

書式

ABS(〈数式〉)

説明

指定した〈数式〉の絶対値を数値として返します。
対象となる〈数式〉の種類は、整数・実数、およびその型の変数・関数です。

記述例

```
A=ABS(-1)
```

用例

「−1」の絶対値を画面表示する場合

```
>PRINT ABS(-1) ……「-1」の絶対値を画面表示させる
1 ……結果の「1」が表示される
>
```

プログラミング例

```
100 FUNCTION SAMPLE
110 INTEGER A,B,C
120 A=INL(WR_0) …… メモリI/Oの内容を読み込み
130 B=A-128 …… その値から「128」を引き
140 C=ABS(B) …… その絶対値を
150 OUTL WY_2,C …… 出力する
160 FEND
```

ADALM( )

オンラインコマンド　ステートメント

概要

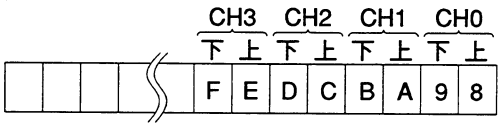
A/D変換ユニットのアラームステータスを返します。

書式

ADALM(〈スロット番号〉)

説明

A/D変換ユニットのアラームステータスを返します。
この値は、A/D変換ユニットの入力ポートX_&M8～X_&MFの値です。
得られる値は、2バイト整数です。
〈スロット番号〉には、0～31を指定します。



注意

この関数はAFP3400に対してのみ使用できます。

参照

入力ポートX_&M8～X_&MFの割り付け内容については、巻末の高機能ユニットパラメーター一覧表をご覧ください。

関連命令

ADASET,ADDATA,ADDATA( ),ADLSET

記述例

```
I=ADALM(4)
```

用例

スロット1のA/D変換ユニットのアラームステータスを表示します。

```
PRINT BIN$(ADALM(1))
101 ……チャンネル0とチャンネル1の
       A/D変換値が上限値を越えてい
       ることを示しています。
```

ADASET

オンラインコマンド　ステートメント

概要

A/D変換ユニットのサンプリング平均回数を設定します。

書式

ADASET␣〈スロット番号〉,〈チャンネル番号〉,〈平均回数〉

説明

A/D変換ユニットのサンプリング平均回数を設定します。この設定値は、ユニットの共有メモリのワードアドレス1～4に書き込まれます。
〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、0～3を指定します。
〈平均回数〉には、3～4,000を指定します。

注意

この命令はAFP3400に対してのみ使用できます。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。

関連命令

ADALM(),ADDATA,ADDATA(),ADLSET

記述例

```
ADASET 4,0,1000
```

ADAVRG

オンラインコマンド　ステートメント

概要

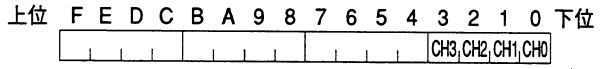
A/D変換ユニットの平均処理を設定します。

書式

ADAVRG␣〈スロット番号〉,〈設定コード〉

説明

A/D変換ユニットの平均処理を〈設定コード〉で設定します。この設定値は、ユニットの共有メモリのワードアドレス0に書き込まれます。
〈スロット番号〉には、0～31を設定します。
〈設定コード〉は、1ワードデータ（16ビット）で、以下のとおりに指定します。



各ビットの値0:サンプリング処理をする
1:平均処理をする

注意

この命令はAFP3400に対してのみ使用できます。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。

関連命令

ADASET,ADDATA,ADDATA(),ADERR(),ADLIMIT,
ADLSET,ADRDY(),ADSCALE,ADSSET

記述例

CH0とCH1に平均処理を設定します。

```
ADAVRG 0,&H3
```

用例

平均処理を行います。

```
100 FUNCTION SAMPLE
110 INTEGER A
120 ADASET 0,1,50 ……CH1の平均回数を"50"に設定
130 ADAVRG 0,&H2 ……平均処理チャンネルを"CH1のみ"に設定
140 WAIT SW(X_&M2) ……CH1のA/D変換準備完了フラグを待つ
150 A=ADDATA(0,1);PRINT A;GOTO 150
160 FEND
```

ADDATA

オンラインコマンド　ステートメント

概要

A/D変換ユニットの変換データを読み出します。

書式

ADDATA_〈スロット番号〉,〈チャンネル番号〉,〈変数〉

説明

A/D変換ユニットの変換データを読み出して、変数に代入します（〈変数〉には実数は使えません）。
なお、この命令を実行すると、ユニットの入力ポートX_&M1～X_&M4の該当チャンネルがONになるまで一時停止します。
〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、0～3を指定します。
注意
この命令はAFP3400に対してのみ使用できます。

関連命令

ADALM(),ADASET,ADDATA(),ADLSET

記述例

```
I=ADDATA 4,1,A
```

用例

```
>PRINT ADDATA 0,1,A
>PRINT A
100
>
```

ADDATA( )

ステートメント

概要

A/D変換ユニットの変換データを返します。

書式

ADDATA(〈スロット番号〉,〈チャンネル番号〉)

説明

A/D変換ユニットの変換データを返します。
このとき変換準備完了フラグ（入力ポートX_&M1～X_&M4）に関係なく変換データを返します。
得られる値は2バイト整数です。
〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、0～3を指定します。
注意
この命令はAFP3400に対してのみ使用できます。

関連命令

ADALM(),ADASET,ADDATA,ADLSET

記述例

```
I=ADDATA(4,1)
```

用例

```
>PRINT ADDATA(0,1)
100
>
```

ADERR( )

オンラインコマンド ステートメント

概要

A/D変換ユニットのエラーステータスを返します。

書式

ADERR(〈スロット番号〉)

説明

A/D変換ユニットのエラーステータスを返します。
この値は、ユニットの共有メモリのワードアドレス27の値です。
得られる値は、2バイト整数です。
〈スロット番号〉には、0～31を指定します。

注意

この命令はAFP3400に対してのみ使用できます。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。

関連命令

ADALM(),ADASET,ADAVRG,ADDATA,ADDATA(),ADLIMIT,ADLSET,ADRDY(),ADSCALE,ADSSET

記述例

```
I=ADERR(4)
```

用例

```
>PRINT BIN$(ADERR(0))
0
>
```

ADLIMIT

オンラインコマンド ステートメント

概要

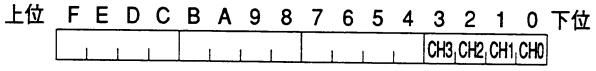
A/D変換ユニットの出力の制限処理を設定します。

書式

ADLIMIT␣〈スロット番号〉,〈指定コード〉

説明

A/D変換ユニットの制限処理を〈指定コード〉を使用して設定します。
この設定値はユニットの共有メモリのワードアドレス5に書き込まれます。
〈スロット番号〉には、0～31を指定します。
〈指定コード〉は、1ワードデータ(16ビット)で、以下のとおり指定します。



各ビットの値0:制限を加えない
1:制限を加える

注意

この命令はAFP3400に対してのみ使用できます。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。

関連命令

ADALM(),ADASET,ADAVRG,ADDATA,ADDATA(),ADERR(),ADLSET,ADRDY(),ADSCALE,ADSSET

記述例

```
ADLIMIT 4,&B1111
```

ADLSET

オンラインコマンド　ステートメント

概要

A/D変換ユニットの出力の上限値、下限値を設定します。

書式

ADLSET_〈スロット番号〉,〈チャンネル番号〉,〈上限値〉,〈下限値〉

説明

A/D変換ユニットのデジタル変換出力の上限値、下限値を設定します。
設定内容は共有メモリに書き込まれます。
〈上限値〉〈下限値〉は、チャンネルのA/D変換値、出力特性にそったデジタル値を設定します。
この設定値は、ユニットの共有メモリの、ワードアドレス6～13に書き込まれます。
上限値、下限値の範囲を超えるとX_&M8～X_&MFに警報信号が出力されます。
また、命令ADARM () を使って、この内容を確認することができます。
〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、0～3を指定します。
〈上限値〉〈下限値〉には、−32,768～＋32,767 (2バイト整数) を指定します。

注意

1. 〈上限値〉〉〈下限値〉となるように設定してください。
2. この命令はAFP3400に対してのみ使用できます。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメータ一覧表をご覧ください。

関連命令

ADALM(),ADDATA,ADDATA()

記述例

```
ADLSET 4,1,3000,500
```

用例

上・下限処理を行います。

```
100 FUNCTION SAMPLE
110 INTEGER A,B
120 ADLSET 0,1,1500,500
130 ADLIMIT 0,&H2
140 WAIT SW(X_&M0)
150 A=ADALM(0);B=ADDATA(0,1);PRINT BIN$(A),B;GOTO 150
160 FEND
```

ADRDY( )

オンラインコマンド　ステートメント

概要

A/D変換ユニットの変換準備完了ステータスを返します。

書式

ADRDY(〈スロット番号〉)

説明

A/D変換ユニットの変換準備完了ステータスを返します。
この値は入力ポートX_&M1～X_&M4の値です。
得られる値は、以下のとおりです。

0:変換準備中
1:変換準備完了

〈スロット番号〉には、0～31を指定します。

関連命令

ADALM(),ADASET,ADAVRG,ADDATA,ADDATA(),
ADERR(),ADLIMIT,ADLSET,ADSCALE,ADSSET

記述例

```
ADRDY(4)
```

ADSCALE

オンラインコマンド　ステートメント

概要

A/D変換ユニットのスケーリング処理をするチャンネルを設定します。

書式

ADSCALE␣〈スロット番号〉,〈指定コード〉

説明

A/D変換ユニットについて、デジタル出力値のスケーリング処理をするチャンネルを〈指定コード〉を使って設定します。
この設定値は、ユニットの共有メモリのワードアドレス14に書き込まれます。
〈スロット番号〉には、0～31を指定します。
〈指定コード〉は、1ワードデータ（16ビット）で、以下のとおり指定します。

上位	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	下位
													CH3	CH2	CH1	CH0	

各ビットの値0：スケーリング処理をしない
1：スケーリング処理をする

注意

この命令はAFP3400に対してのみ使用できます。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。

関連命令

ADSSET

記述例

CH2、CH3をスケーリング処理します。

```
ADSCALE 4,&B1100
```

ADSSET

オンラインコマンド　ステートメント

概要

A/D変換ユニットのスケーリング値を設定します。

書式

ADSCALE␣〈スロット番号〉,〈チャンネル番号〉,〈オフセット変更点〉,〈フルスケール変更点〉

説明

A/D変換ユニットのスケーリング値を、〈オフセット変更点〉〈フルスケール変更点〉を使用して設定します。
この設定値は、ユニットの共有メモリのワードアドレス15～22に書き込まれます。
〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、0～3を指定します。
〈オフセット値〉〈フルスケール値〉には、－32,768～＋32,767（2バイト整数）を指定します。

注意

この命令はAFP3400に対してのみ使用できます。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。

関連命令

ADSCALE

記述例

```
ADDSSET 4,1,0,1000
```

用例

スケーリング処理を行います。

```
100 FUNCTION SAMPLE
110 INTEGER A
120 ADSSET 0,0,500,1500
130 ADSCALE 0,&H1
140 WAIT SW(X_&M0)
150 A=ADDATA(0,0):PRINT A;GOTO 150
160 FEND
```

ASC( )

オンラインコマンド　ステートメント

概要

1バイトの文字に対応するキャラクターコードを返します。

書式

ASC(〈文字列〉)

説明

〈文字列〉で指定した文字に対応するアスキー形式のキャラクターコードを返します。
〈文字列〉に2文字以上の文字列を指定した場合は先頭の1文字だけが変換対象となります。
〈文字列〉が空文字(文字数が0)の場合はエラーとなります。

関連命令

CHR$()

記述例

```
PRINT HEX$(ASC(A$))
```

ATN( )

オンラインコマンド　ステートメント

概要

数値の逆正接(アークタンジェント)を返します。

書式

ATN(〈数式〉)

説明

〈数式〉のATN(逆正接値)をラジアンで返します。
得られる値は、$-\pi/2$から$\pi/2$までの範囲です。
ラジアンを度に変換するには180/πを掛けます。
〈数式〉の型に関係なく単精度(4バイトの実数)の値を返します。
(π=3.14159265358979323846……)

関連命令

COS(),SIN(),TAN()

用例

```
>PRINT ATN(1)*180/3.1415926
 45.
>
```

AUTO

オンラインコマンド　オフラインコマンド

概要

行番号を自動的に発生します。

書式

AUTO␣〈開始行番号〉,[〈増分〉]

説明

プログラムを記述するときの行番号を自動的に発生させます。

行番号は指定した〈開始行番号〉から始まり、指定した〈増分〉の間隔をあけて発生します。

STOP キーを押すか CTRL ＋ C キーを押すと、行番号の自動発生を解除します。

増分を省略した場合は発生させる行番号の間隔は10になります。

注意

行番号として使えるのは、1～32767の範囲の整数です。

記述例

AUTO 1000,10

用例

100で始まり、50ずつ増加させる場合

```
>AUTO 100,50
100 ■ ……行番号が自動的に表示され入力待ちになる
100 FUNCTION MAIN ↵ ……命令文を入力しリターンキーを押す
150 ■ ……次の行番号が自動的に表示さ
 :                    れ入力待ちになる
 :
900 FEND ↵
950 ■ ……最後の行を入力したらSTOP
>■                    キーを押し終了させる
```

BCD( )

オンラインコマンド　ステートメント

概要

バイナリコードを2進化10進コードに変換します。

書式

BCD(〈数式〉)

説明

バイナリコードの〈数式〉(BINデータ)を2進化10進コード(BCDデータ)に変換し、その値を返します。

対象となる〈数式〉の種類は整数・実数およびその型の変数・関数です。

〈数式〉に実数を指定すると、小数点以下が四捨五入され、2バイト整数として扱われます。

〈数式〉の値と返す値は、以下のとおりです。

0～99999999→返す値:&H0～&H99999999

関連命令

BIN()

用例

外部出力に接続された4行の数字表示装置(1桁4ビット)に数値を表示させる

```
>OUTW WY_2,&H1234 ……BCD()を使用しない場合
>■
>OUTW WY_2,BCD(1234) ……BCD()を使用すると10進数のま
>■                      ま指定できる
```

プログラミング例

```
100 FUNCTION SAMPLE
110 INTEGER A,B,C,D
120 A=INW(WX_0) ……デジタルスイッチの値を読み込む
130 B=BIN(A) ……バイナリコードに変換する
140 C=B+500 ……500を加える
150 D=BCD(C) ……2進化10進コード変換する
160 OUTW WY_2,D ……数字表示装置に出力する
170 FEND
```

BIN( )

オンラインコマンド　ステートメント

概要

2進化10進コードをバイナリコードに変換します。

書式

BIN(〈数式〉)

説明

2進化10進コードの〈数式〉(BCDデータ)をバイナリコードに変換し、その値を返します。
対象となる〈数式〉の種類は、整数およびその型の変数·関数です。
〈数式〉に実数を指定すると、小数点以下が四捨五入され、2バイト整数として扱われます。
〈数式〉の値と返す値は、以下のとおりです。
&H0～&H99999999 →返す値:0～9999999

関連命令

BCD()

用例

外部入力に接続された4桁のデジタルスイッチ(0000～9999)の値を表示する

```
>PRINT INW(WX_0) ……BIN( )を使用しない場合
4660
>
```

```
>PRINT BIN(INW(WX_0)) …BIN( )を使用すると10進数に変換される
1234
>
```

プログラミング例

```
100 FUNCTION SAMPLE
110 INTEGER A,B,C,D
120 A=INW(WX_0) ……デジタルスイッチの値を読み込む
130 B=BIN(A) ……バイナリコードに変換する
140 C=B+500 ……500を加える
150 D=BCD(C) ……2進化10進コード変換する
160 OUTW WY_2,D……数字表示装置に出力する
170 FEND
```

BIN$( )

オンラインコマンド　ステートメント

概要

数値を2進表記の文字列に変換します。

書式

BIN$(〈数式〉)

説明

指定した〈数式〉2進表記の文字列に変換して返します。
対象となる〈数式〉の種類は、整数·実数およびその型の変数·関数です。
〈数式〉に実数を指定すると、小数点以下が四捨五入され、2バイト整数として扱われます。

関連命令

HEX$(),MEW$(),OCT$()

記述例

```
A$=BIN$(123)
```

用例

8進数「&O200」を2進表記の文字列に変換する 場合

```
>PRINT BIN$(&O200)
10000000
>
```

10進数「200」を2進数表記の文字列に変換する 場合

```
>PRINT BIN$(200)
11001000
>
```

16進数「&H200」を2進数表記の文字列に変換する 場合

```
>PRINT BIN$(&H200)
100000000
>
```

プログラミング例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 STRING B$
130 A=INL(WX_0) ……「WY_0」ポートの内容を
140 OUTL WY_2,A ……「WY_2」ポートに出力し
150 B$=BIN$(A) ……2進数表記に変換して
160 PRINT B$ ……画面に表示する
170 FEND
```

BLKMOV

オンラインコマンド　ステートメント

概要

I/O間のデータ転送をします。

書式

BLKMOV␣〈ワード指定I/O番号(1)〉,〈ワード指定I/O番号(2)〉,〈ワード数〉

説明

〈ワード指定I/O番号(1)〉で指定したI/Oから〈ワード数〉分だけのデータを、〈ワード指定I/O番号(2)〉で指定したI/Oに転送します。

〈ワード指定I/O番号〉は、以下のとおり指定します。

外部入力 I/O	WX_n	n=0～127
外部出力 I/O	WY_n	n=0～127
メモリ I/O	WR_n	n=0～97
リンクメモリ I/O	WL_n	n=0～127
データメモリ	DT_n	n=0～2047
リンクデータメモリ	LD_n	n=0～255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

記述例

```
BLKMOV DT_0,DT_1024,1024
```

用例

```
100 FUNCTION SAMPLE
110 BLKOUT WY_0,2,-1 ……書き込みデータ-1を
120 PRINT "WY_0",INW(WY_0)      WY_0とWY_1へ書き込みます
130 BLKMOV WY_0,WY_10,2 ……WY_0およびWY_1に格納されて
140 PRINT "WY_10",INW(WY_10)    いるデータをWY_10およびWY_11に
150 FEND                        転送します。
```

BLKOUT

オンラインコマンド　ステートメント

概要

I/Oへの一括書き込みをします。

書式

BLKOUT␣〈ワード指定I/O番号〉[,〈ワード数〉],〈書き込みデータ〉

説明

〈ワード指定I/O番号〉で指定したI/Oから〈書き込みデータ〉を〈ワード数〉だけ出力します。

〈書き込みデータ〉で指定できる範囲は−32,768～+32,767です。

〈ワード指定I/O番号〉は、以下のとおり指定します。

外部入力 I/O	WX_n	n=0～127
外部出力 I/O	WY_n	n=0～127
メモリ I/O	WR_n	n=0～97
リンクメモリ I/O	WL_n	n=0～127
データメモリ	DT_n	n=0～2047
リンクデータメモリ	LD_n	n=0～255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

記述例

```
BLKOUT DT_0,2048,&H1234
```

用例

```
100 FUNCTION SAMPLE
110 BLKOUT WY_0,2,-1 ……書き込みデータ-1を
120 PRINT "WY_0",INW(WY_0)      WY_0とWY_1へ書き込みます
130 BLKMOV WY_0,WY_10,2 ……WY_0およびWY_1に格納されている
140 PRINT "WY_10",INW(WY_10)    データをWY_10およびWY_11に
150 FEND                        転送します。
```

BYTE

ステートメント

概要

変数を1バイト整数型として使用するよう宣言します。

書式

BYTE␣〈変数名〉[,〈変数名〉･･･]
BYTE␣〈変数名〉(〈添字の最大値〉[,〈添字の最大値〉･･･])

説明

1バイト整数型の変数の使用を宣言します。
変数の値の範囲:
･−128～+127
･&B 1111 1111 1111 1111 1111 1111 1000 0000
～&B 0000 0000 0000 0000 0000 0000 0111 1111
･&O 37777777600～&O 00000000177
･&H FFFFFF80～&H 0000007F
･&M 0～&M 7F
〈変数名〉には、8文字までの英数字と「_(アンダースコア)」が使用できますが、先頭の1文字は必ず英字で書き始めなければならないという約束があります。また、Pで始まる変数名は使用できません。

注意

1.予約語は、変数名として使用することができません。
2.数値型の変数の場合末尾に記号は付けません。
3.値が代入される前の変数は、数値変数では「0」として扱われます。
4.変数宣言は、プログラム中の最初のタスクブロック(FUNCTION～FEND)のはじめにまとめて記述します(タスクブロックの途中で変数宣言をすると正しくアップロードできない場合があります)。
5.配列変数の場合、添字の最小値は0になります。

関連命令

INTEGER,LONG,REAL,STRING

用例

```
100 FUNCTION SAMPLE
110 BYTE A
120 INTEGER B
130 LONG C
140 REAL D
150 STRING E$
 :
 :
300 FEND
```

CALL～FUNCTION

ステートメント

概要

他のタスクをサブルーチンとして呼び出します。

書式(呼出側:)

CALL_〈サブルーチンタスク名〉(〈引数〉[,〈引数〉･･･])

書式(受け側:)

FUNCTION_〈サブルーチンタスク名〉(〈引数〉[,〈引数〉･･･])

説明

他のタスクをサブルーチンとして実行します。
CALL文を使用すれば、他のタスクをサブルーチンとして呼び出せ、さらにサブルーチンとの間で変数の値の受け渡しができます。

注意

1. 呼出側の〈引数〉には、変数または定数が使用できます。
   但し、配列変数は使用できません。
2. 呼出側の〈引数〉に定数を使用した場合、受け側の〈引数〉の変数型は下記のようになります。
   呼出側の定数
   2バイト以下の整数 → 2バイト整数(INTEGER)
   4バイト以下の整数 → 4バイト整数(LONG)
   4バイト以下の実数 → 4バイト実数(REAL)
3. 呼出側の〈引数〉に定数を使用した場合、受け側の〈引数〉の変数型は呼出側で宣言された変数型と同じ型になります。
4. 受け側の〈引数〉には、変数のみ使用できます。
   但し、変数の型は宣言しないでください。また、配列変数は使用できません。
5. 呼出側と受け側の〈引数〉の数が違う場合、呼出側の引数の数を受け側の〈引数〉の数にあわせてください。
   (呼び出し側にダミーの〈引数〉をつけ加えてください。)
6. 実行時にBASICスタックオーバーフロー(ERROR2004)が発生した場合は、MAP命令を参照し、このエラーの発生したタスクのスタック領域を増やしてください。

用例

1.2つのタスクから同じサブルーチンを呼出します。
(引数は定数を使っています)

```
100 FUNCTION SAMPLE1
110 CALL SUB(100,200) ……受け側の〈引数〉であるA,Bはそれぞれ
120 XQT !2,SAMPLE2          2バイト整数(INTEGER)になります。
130 FEND
140 '
150 FUNCTION SAMPLE2
160 CALL SUB(300000,400000) ……受け側の〈引数〉であるA,Bはそれぞれ
170 FEND                    4バイト整数(LONG)になります。
180'
190 FUNCTION SUB (A,B)
200 A=A*10:B=B*100
210 PRINT A,B
220 FEND
>XQT [↵] ←打込み
1000   20000
3000000   40000000←表示
```

2.2つのタスクから同じサブルーチンを呼出します。
(引数は変数を2つずつ使っています)

```
100 FUNCTION SAMPLE1
110 INTEGER A,B
120 LONG C,D
130XQT!2,SAMPLE2
140 A=10:B=20
150 CALL SUB(A,B) ……受け側の〈引数〉であるX,Yはそれぞれ A,Bの型
160 PRINT A,B          (INTEGER)になります。
170 FEND
180 '
190 FUNCTION SAMPLE2
200 C=30:D=40
210 CALL SUB(C,D) ……受け側の〈引数〉であるX,Yはそれぞれ C,Dの型
220 PRINT C,D          (LONG)になります。
230 FEND
240 '
250 FUNCTION SUB(X,Y)
260 X=X*10:Y=Y*100
270 FEND
>XQT [↵] ←打込み
100 2000
300 4000←表示
```

CHR$( )

オンラインコマンド　ステートメント

概要

キャラクターコードに対応する1バイトの文字を返します。

書式

CHR$(〈数式〉)

説明

〈数式〉で指定したキャラクタコードに対応する1バイトの文字を返します。
指定できる〈数式〉の範囲は0～255です。範囲を越えるとエラーとなります。

関連命令

ASC()

記述例

```
A$=CHR$(&H42)
```

用例

キャラクターコード「66」に対応する文字を調べる

```
>PRINT CHR$(66)
B
>
```

プログラミング例

```
100 FUNCTION SAMPLE
120 INTEGER A
130 STRING B$
140 A=66 ……数値「66」の
150 B$=CHR$(A) ……キャラクター(文字)を
160 PRINT B$ ……画面に表示する
170 FEND
```

CLOSE

ステートメント

概要

ファイルを閉じます。

書式

CLOSE␣[[#]〈ファイル番号〉[,#]〈ファイル番号〉]･･･]

説明

〈ファイル番号〉で指定したディスクファイルを閉じます。〈ファイル番号〉を複数指定することにより、一度に複数のファイルを閉じることができます。また、〈ファイル番号〉を省略した場合、現在開かれている全てのファイルを閉じます。ファイルを出力用に開いたときは、バッファに残っていたデータを書き出しますので、ファイルの出力処理を正しく終了するには、「CLOSE」命令の実行が必要です。

関連命令

OPEN

記述例

```
CLOSE #1,#2
```

用例

```
100 FUNCTION SAMPLE
110 INTEGER A,B,C
 ･
 ･
 ･
200 OPEN "INITDATA" FOR OUTPUT AS #1
210 A=1;B=2;C=3
220 PRINT #1,A,B,C ……変数の初期値を「INITDATA」に
230 CLOSE #1 設定する
 ･
 ･
 ･
300 OPEN "INITDATA" FOR INPUT AS #1
310 INPUT #1,A,B,C ……変数の初期値を「INITDATA」より
320 CLOSE #1 読み出す
 ･
 ･
 ･
420 OPEN "TEST.DAT" FOR APPEND AS #1
430 PRINT #1,DATE$(),TIME$(),A,B,C ……変数の値を「TEST.DAT」に記録
440 CLOSE #1 する
 ･
 ･
 ･
500 FEND
```

CLRVAL

オンラインコマンド　ステートメント

概要

グローバル変数を初期化します。

書式

CLRVAL

説明

グローバル変数を初期化します。
初期化後の値は数値変数は0、文字変数は""（ヌルストリング）となります。

関連命令

CLRVAR,VERINIT

記述例

CLRVAL

CLRVAR

オンラインコマンド

概要

BASICタイプCPUユニットのプログラムと変数を初期化します。

書式

CLRVAR

説明

BASICタイプCPUユニットのメモリのプログラムと変数を初期化します。
なお、パラメータメモリで設定されたI/O、メモリI/Oおよびデータメモリの保持・非保持は初期化されません。これらを初期化したい場合は「VERINIT」で初期化してください。

関連命令

CLRVAL,VERINIT

記述例

CLRVAR

CLS

オンラインコマンド　オフラインコマンド　ステートメント

概要

パソコンの画面をクリア（表示を消去）します。

書式

CLS

説明

パソコンの表示画面、およびエディット画面に表示された内容を消去します。消去されるのは画面上の表示だけで、プログラムメモリ内のプログラム、変数は消去されません。
画面を消去したあと、カーソルは画面の最上行の左端に移動します。

記述例

CLS

用例

```
AUTO
>10 FUNCTION SAMPLE
>20 INTEGER A
>
```

```
>CLS
>
```

COLOR

オンラインコマンド　ステートメント

概要

パソコンの画面に表示される文字の色を指定します。

書式

COLOR␣〈カラーコード〉

説明

パソコンの画面に表示される文字の色を〈カラーコード〉の色番号で指定します。
〈カラーコード〉の色番号は、以下のとおり指定します。

0:黒
1:青
2:赤
3:紫
4:緑
5:水色
6:黄色
7:白

COMPILE

オンラインコマンド　オフラインコマンド

概要

ソースプログラムをオブジェクトプログラムに変換します。

書式

COMPILE␣["[〈ドライブ番号〉:][〈パス名〉]〈ファイル名〉"][,H][,L]

説明

パソコンのプログラムメモリ上のプログラムから、CPUで実行できるオブジェクトプログラムを生成し、ディスク上のファイルにセーブします。

〈ドライブ番号〉を指定することにより、フロッピーディスク上のプログラムファイルから直接オブジェクトファイルを生成することもできます。パソコン上で編集したプログラムをダウンロードする際には、この「COMPILE」コマンドでソースプログラムをオブジェクトプログラムに変換してからCPUユニットに転送します。

対象となるソースファイルは、拡張子「.PRG」のASCII形式（テキスト）ファイルです。得られるファイルは、ソースプログラムの〈ファイル名〉に拡張子「.OBJ」を付けたオブジェクトファイルと、同じく拡張子「.SYM」を付けたシンボリックファイルです。シンボリックファイルは、変数名、ラベル名を管理するためのファイルです。

〈ファイル名〉には、数字で始まるものは使えません。

〈ドライブ番号〉〈パス名〉〈ファイル名〉を省略すると、パソコンのプログラムメモリ内でエディット中のソースファイルをコンパイルし、「SYSTMP.OBJ」「SYSTMP.SYM」の2つのファイルを作成し、ディスクに書き込みます。

参考

「H」オプションをつけてコンパイルすると、拡張子「.HEX」のファイルを作成します。これはROMライタ用のインテルHEX方式のプログラムファイルです。

「L」オプションをつけると、コンパイルするときに、コメントのみの行を削除します。（ソースファイルには残ります）これにより、オブジェクトプログラムのサイズを削除した分だけ小さくすることができます。

注意

FP-BASIC編集ソフトVer.2で作成したHEXファイルは、FP3-BASIC CPUユニットVer.1では使用できません。

FP3-BASIC CPUユニットVer.1用のHEXファイルは、FP-BASIC編集ソフトVer.1で作成してください。

記述例

```
>COMPILE "C:¥TEST"
>COM "C:¥TEST"
>COM
```

用例

```
>FILES ……ファイルを画面に一覧表示する
TEST.PRG ……ソースプログラムしかない
>COMPILE "TEST" ……コンパイルする
COMPILE END
>FILES ……ファイルの内容を再表示する
TEST.PRG TEST.OBJ TEST.SYM
  ……新しいファイルが生成されている
>
```

オプションを指定したとき

```
COMPILE,L ……コメント行を削除してコンパイルする
COMPILE,H ……コメント行もコンパイルし、HEXファイルを作成する
COMPILE,L,H ……コメント行を削除してコンパイルし、HEXファイルを作成する
```

CONT

オンラインコマンド

概要

PAUSE命令で一時停止しているタスクの実行を再開します。

書式

CONT␣[!〈タスク番号〉]

説明

「PAUSE」命令により一時停止しているタスクの実行を再開します。
〈タスク番号〉を省略すると、一時停止しているすべてのタスクが実行を再開します。
「PAUSE」状態のタスクがないときは、この命令は無視されます。
注意
「HALT」命令で一時停止しているタスクの実行は再開できません。

関連命令

PAUSE,STEP,TEST

記述例

```
CONT !1
CONT
```

用例

タスク2の一時停止(PAUSE)を解除し、実行を再開する場合

```
>CONT !2
```

すべての一時停止(PAUSE)状態のタスクの実行 を再開する場合

```
>CONT
```

COS( )

オンラインコマンド　ステートメント

概要

数値の余弦(コサイン)を返します。

書式

COS(〈数式〉)

説明

〈数式〉の値に対する余弦(コサイン)の値を返します。
〈数式〉の単位はラジアンです。
角度が度で与えられている場合は、ラジアンに変換するためにπ/180を掛けます。
〈数式〉の型には関係なく単精度の値を返します。
(π=3.14159265358979323846……)

関連命令

ATN(),SIN(),TAN()

用例

```
>PRINT COS(3.1415926/180*60)
0.5
>
```

CVB( )

オンラインコマンド　ステートメント

概要

1バイトの文字を1バイト整数に変換します。

書式

CVB(〈文字列〉)

説明

1バイトの文字を1バイトの整数に変換して返します。
1バイトに満たない文字列データのときは0 (NULL) を返します。

関連命令

CVL(),CVS(),CVW()

用例

```
>PRINT CVB("A")
65
>
```

CVL( )

オンラインコマンド　ステートメント

概要

4バイトの文字列を4バイト整数に変換します。

書式

CVL(〈文字列〉)

説明

4バイトの文字列を4バイト整数に変換して返します。
4バイトに満たない文字列データのときは、上位バイト側に「0」を与えます。

関連命令

CVB(),CVS(),CVW()

用例

```
>PRINT CVL("1234")
875770417
>PRINT HEX$(CVL("1234"))
34 33 32 31
```

CVS( )

オンラインコマンド　ステートメント

概要

4バイトの文字列を4バイト実数に変換します。

書式

CVS(〈文字列〉)

説明

4バイトの文字列を4バイト実数に変換して返します。

関連命令

CVB(),CVL(),CVW()

用例

```
>PRINT CVS("ABCD")
781.0352
```

CVW( )

オンラインコマンド　ステートメント

概要

2バイトの文字列を2バイト整数に変換します。

書式

CVW(〈文字列〉)

説明

2バイトの文字列を2バイト整数に変換して返します。
2バイトに満たない文字列データのときは、上位バイト側に「0」を与えます。

関連命令

CVB(),CVL(),CVS()

用例

```
>PRINT CVW("AB")
16961
>PRINT HEX$(CVW("AB")
4241
```

DAALM( )

オンラインコマンド　ステートメント

概要

D/A変換ユニットのアラームステータスを返します。

書式

DAALM(〈スロット番号〉)

説明

D/A変換ユニットのアラームステータスを返します。
この値は、ユニットの入力ポートX_&M1～X_&M6の値です。
得られる値は2バイト整数です。
〈スロット番号〉には、0～31を指定します。

注意

この命令はAFP3410,AFP3411に対してのみ使用できます。

参照

入力ポートX_&M1～X_&M6の割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。

関連命令

DAERR(),DALIMIT,DALSET,DARDY(),DASET

記述例

```
I=DAALM(4)
```

DAERR( )

オンラインコマンド　ステートメント

概要

D/A変換ユニットのエラーステータスを返します。

書式

DAERR(〈スロット番号〉)

説明

D/A変換ユニットのエラーステータスを返します。
この値は、ユニットの共有メモリのワードアドレス7の値です。
得られる値は、2バイト整数です。
〈スロット番号〉には、0～31を指定します。

注意

この命令はAFP3410,AFP3411に対してのみ使用できます。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。

関連命令

DAALM(),DALIMIT,DALSET,DARDY(),DASET

記述例

```
I=DAERR(4)
```

DALIMIT

オンラインコマンド ステートメント

概要

D/A変換ユニットのアナログ出力の制限処理の設定をします。

書式

DALIMIT␣〈スロット番号〉,〈設定コード〉

説明

D/A変換ユニットのアナログ出力の制限処理の設定をします。
制限処理の設定は〈設定コード〉によります。
D/A変換ユニットの出荷時はCH0,CH1とも、アナログ出力に制限を加えない設定になっています。
この設定はユニットの共有メモリのワードアドレス0に書き込まれます。
〈スロット番号〉には、0～31に指定します。
〈設定コード〉は、以下のとおり指定します。

0:チャンネル0、1の両方に制限を加えません。
1:チャンネル0に制限を加え、チャンネル1には制限を加えません。
2:チャンネル1に制限を加え、チャンネル0には制限を加えません。
3:チャンネル0、1の両方に制限を加えます。

注意

この命令はAFP3410,AFP3411に対してのみ使用できます。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。

関連命令

DAALM(),DAERR(),DALSET,DARDY(),DASET

記述例

```
DALIMIT 4,1
```

用例

出力制限を行いデータをセットします。

```
100 FUNCTION SAMPLE
110 DALSET 1,0,1500,300 ····· 上限値を"1500",下限値を"300"に設定
120 DALIMIT 1,&H1 ····· アナログ出力制限チャンネ
130 '                    ルを"CH0のみ"に設定
140 IF DARDY(1)=0 THEN GOTO 140  D/A変換準備完了を待つ
150 '
160 DASET 1,0,1000 ·····
                                 D/A変換入力値を"1000"に設定
170 FEND
```

注意

アナログ出力制限を行う場合はD/A変換準備完了が立ってからD/A変換入力値を設定してください。

DALSET

オンラインコマンド　ステートメント

概要

D/A変換ユニットの上限値、下限値を設定します。

書式

DALSET␣〈スロット番号〉,〈チャンネル番号〉,〈上限値〉,〈下限値〉

説明

D/A変換ユニットのアナログ出力の〈上限値〉、〈下限値〉を設定します。

〈上限値〉、〈下限値〉はユニットの入出力特性に合わせて設定してください。

この設定値は、ユニットの共有メモリのワードアドレス1～4に書き込まれます。

〈上限値〉、〈下限値〉の範囲を超えるとX_&M3～X_&M6に警報信号が出力されます。

〈スロット番号〉には、0～31を指定します。

〈チャンネル番号〉には、0または1を指定します。

〈上限値〉、〈下限値〉は以下のとおり指定します。

－10～＋10V (－20～20mA) レンジユニット

データ範囲 －2000～＋2000

1～5V (4～20mA) レンジユニット

データ範囲 0～＋4000

注意

この命令はAFP3410,AFP3411に対してのみ使用できます。

参照

共有メモリおよびX_&M3～X_&M6の割り付けの内容については、巻末の高機能ユニットパラメータ一覧表をご覧ください。

関連命令

DAALM(),DAERR(),DALIMIT,DARDY(),DASET

記述例

DALSET 4,1,1000,-1000

用例

出力制限を行いデータをセットします。

```
100 FUNCTION SAMPLE
110 DALSET 1,0,1500,30 ……上限値を"1500"、下限値を"300"に設定
120 DALIMIT 1,&H1 ……アナログ出力制限チャンネルを"CH0のみ"に設定
130 '
140 IF DARDY(1)=0 THEN GOTO 140 ……D/A変換準備完了を待つ
150 '
160 DASET 1,0,1000 ……D/A変換入力値を"1000"に設定
170 FEND
```

注意

アナログ出力制限を行う場合はD/A変換準備完了が立ってからD/A変換入力値を設定してください。

DARDY( )

オンラインコマンド　ステートメント

概要

D/A変換ユニットの設定完了ステータスを返します。

書式

DARDY(〈スロット番号〉)

説明

D/A変換ユニットの設定の完了を確認し、結果を返します。この値は、D/A変換ユニットの入力ポートX_&M0の値です。得られえる値は、以下のとおりです。

0:変換準備中

1:変換準備完了

〈スロット番号〉には、0～31を指定します。

注意

この命令はAFP3410,AFP3411に対してのみ使用できます。

関連命令

DAALM(),DAERR(),DALIMIT,DALSET,DARDY(),DASET

用例

出力制限を行いデータをセットします。

```
100 FUNCTION SAMPLE
110 DALSET 1,0,1500,30 ……上限値を"1500"、下限値を"300"に設定
120 DALIMIT 1,&H1 ……アナログ出力制限チャンネルを"CH0のみ"に設定
130 '
140 IF DARDY(1)=0 THEN GOTO 140 ……D/A変換準備完了を待つ
150 '
160 DASET 1,0,1000 ……D/A変換入力値を"1000"に設定
170 FEND
```

注意

アナログ出力制限を行う場合はD/A変換準備完了が立ってからD/A変換入力値を設定してください。

DASET

オンラインコマンド　ステートメント

概要

D/A変換ユニットの入力値を設定します。

書式

DASET␣〈スロット番号〉,〈チャンネル番号〉,〈変換設定値〉

説明

D/A変換ユニットのデジタル入力値を設定します。〈変換設定値〉はユニットの入出力特性に合わせて設定してください。

この設定は、ユニットの共有メモリのワードアドレス5,6に書き込まれます。

〈スロット番号〉には、0～31を指定します。

〈チャンネル番号〉には、0または1を指定します。

〈変換設定値〉は、以下のとおり指定します。

－10～＋10V (－20～20mA) レンジの場合

データ範囲 －2000～＋2000

1～5V (4～20mA) レンジの場合

データ範囲 0～＋4000

注意

この命令はAFP3410,AFP3411に対してのみ使用できます。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。

関連命令

DAALM(),DAERR(),DALIMIT,DALSET,DARDY()

記述例

```
DASET 4,1,500
```

用例

出力制限を行わずデータをセットします。

```
100 FUNCTION SAMPLE
110 DASET 1,0,500 ……D/A変換入力値を"500"に設定
120 FEND
```

注意

アナログ出力制限を行わない場合はD/A変換準備完了は出ませんのでそれを待つ必要はありません。

DATA

ステートメント

概要

DREAD文で読み出されるデータを定義します。

書式

DATA␣〈定数〉[,〈定数〉]

説明

DREAD文で読み出される数値、文字定数を定義します。
DATA文は非実行文でファンクション内のどこにでも置くことができ、任意の行数だけ書くことができます。
DATA文はいくつ定義してもかまいません。
DREAD文は同一ファンクション内の行番号の小さい方から順番に、DATA文中のデータを読み出していきます。
注意
〈定数〉にMEW表記(&M)を使う時は2桁以上で記述してください。例:&M00,&M01等

関連命令

DREAD,RESTORE

用例

```
100 FUNCTION SAMPLE
110 INTEGER A,B,C,D
120 DREAD A,B,C,D ……DATA文で定義されたデータを変数A,B,C,Dに
130 PRINT A          それぞれ読み出します。
140 PRINT B
150 PRINT C
160 PRINT D
170 DATA 10,20,30,40
180 FEND
```

DATE$

オンラインコマンド　ステートメント

概要

CPUユニットの日付を設定します。

書式

DATE$␣"〈年〉/〈月〉/〈日〉"

説明

CPUユニットの〈年〉〈月〉〈日〉の日付を設定します。
〈年〉は西暦の下2桁を指定します。
〈月〉は、その月が1桁の場合、先頭に0を付けて指定します。(例:4月=04)。
〈日〉は、その日が1桁の場合、先頭に0を付けて指定します。(例:8日=08)。
1993年4月8日の場合は、DATE$"93/04/08"となります。

関連命令

DATE$(),TIME$

記述例

```
DATE$ "92/10/08"
```

DATE$( )

オンラインコマンド　ステートメント

概要

CPUユニットの日付を返します。

書式

DATE$()

説明

CPUユニットに設定されている日付を返します。
返される日付は"年/月/日"です（1993年4月8日の場合は"93/04/08"が返ります）。

関連命令

DATA$,TIME$()

記述例

```
PRINT DATE$()
```

DEBUG Ver2.3以降

オンラインコマンド

概要

シンボル情報を読み込み、デバッグを可能にします。

書式

DEBUG␣["〈ファイル名〉"]

説明

CPU本体またはディスクより、シンボル情報を読み込み、変数を使用したデバッグを可能にします。(FP-BASICを起動した直後はシンボル情報がないため変数を使用したデバッグができません。)
〈ファイル名〉を省略すると、CPU本体よりシンボル情報をアップロードします。
〈ファイル名〉を指定すると、その〈ファイル名〉.SYMをディスクよりロードします。

記述例

```
DEBUG
DEBUG "TEST"
```

DEC

ステートメント

概要

変数をディクリメントします。(カウントDOWN)

書式

DEC␣〈数値変数〉[,〈数式〉]

説明

〈数値変数〉の値を、〈数式〉の値分減算(ディクリメント)します。

〈数式〉が省略されたときは1を減算します。

関連命令

INC

記述例

```
DEC A
```

用例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 A=20
130 WHILE A>10
140 PRINT A
150 DEC A ……変数Aから1減算します。
160 WEND
170 FEDD
```

DECO( )

オンラインコマンド　ステートメント

概要

数値をデコード変換します。

書式

DECO(〈数式〉)

説明

指定した〈数式〉をデコード変換し(2のn乗を求め)、その数値を返します。

対象となる〈数式〉の種類は、整数・実数およびその型の変数・関数です。

〈数式〉に実数を指定すると、小数点以下が四捨五入され、2バイト整数として扱われます。

指定した〈数式〉の「型」と変換後の「型」は同じものになります。

〈数式〉の値と返す値は、以下のとおりです。

数式:0～31 (&H0～&H1F)

→返す値:&B1～&B100 0000 0000 0000 0000 0000 0000

関連命令

ENCO()

用例

指定した出力ポートのみをONする

```
>OUTW WY_2,DECO(1) ……WY_2の1番ポートをONする
>
```

プログラミング例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 FOR A=0 TO 15
130 OUTW WY_2,DECO(A) ……WY_2の0番ポートから15番
140 WAIT 1                までを順番にONする
150 NEXT
160 OUTW WY_2,0
170 FEND
```

DIST( )

オンラインコマンド　ステートメント

概要

数値を4ビット単位で分離します。

書式

DIST(〈数式〉,〈分離位置〉)

説明

〈数式〉を4ビット単位で区切り、指定した〈分離位置〉の4ビットを2バイト整数の下位4ビットとし、その数値を返します。

対象となる〈数式〉の種類は、整数・実数とその変数・関数です。

〈数式〉に実数が指定された場合は、小数点以下が四捨五入され、2バイト整数として処理されます。

〈分離位置〉は1～8の数値で指定します。

指定した〈分離位置〉より〈数式〉のビットが少ない場合は「0」を返します。

注意

〈分離位置〉は、以下のとおりです。

8	7	6	5	4	3	2	1
31 30 26 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0

用例

0ビット～3ビットを分離した場合

```
>PRINT BIN$(DIST(&HF3,1))
11
>
```

4ビット～7ビットを分離した場合

```
>PRINT BIN$(DIST(&HF3,2))
1111
>
```

DO～LOOP

ステートメント

概要

条件式が満たされている間、命令文を繰り返し実行します。(DO条件ループ)

書式

DO_ [WHILE | UNTIL_ 〈条件式〉]～LOOP [WHILE | UNTIL〈条件式〉]

説明

WHILEは条件が真の間、DOからLOOPまでを繰り返し実行します。

UNTILは条件式が真になるまでDOからLOOPまでを繰り返し実行します。

注意

1.DO、LOOPのどちらにもWHILE、UNTILがないとき、無限ループになります。

2.「DO～LOOP」のネスティングは最高10段まで可能です。

関連命令

WHILE～WEND

用例

```
100 FUNCTION SAMPLE
110 INTEGER I
120 I=1
130 DO UNTIL I>3 ……条件式Iの値がI>3となるまで、LOOP間での命令を繰り返し実行します。
140 ON Y_I
150 WAIT 1
160 INC I
170 LOOP ……I>3の時次に進みます。
180 DO WHILE I>=3 ……条件式Iの値がI>=3(I≧3)となっている間、LOOPまでの命令をくり返し実行します。
190 ON Y_I
200 WAIT 1
210 IF I=6 THEN EXIT
220 INC I
230 LOOP
240 END
250 FEND
```

DREAD

ステートメント

概要

DATA文で定義したデータを読み出し、変数に代入します。

書式

DREAD␣〈変数〉[,〈変数〉･･･]

説明

DATA文で定義した数値、文字、定数を読み出し、変数に代入します。
DREAD文は同一ファンクション内のDATA文と組み合わせて使用する必要があります。
DREAD文は同一ファンクション内のDATA文のデータを、行番号の小さい方から順番に1対1の対応で読み込まれて変数に割り当てていきます。

関連命令

DATA,RESTORE

用例

```
100 FUNCTION SAMPLE
110 INTEGER A,B,C,D
120 DREAD A,B,C,D ……DATA文で定義されたデータを変数A,B,C,Dに
130 PRINT A          それぞれ読み出します。
140 PRINT B
150 PRINT C
160 PRINT D
170 DATA 10,20,30,40
180 FEND
```

DUMP

オンラインコマンド

概要

I/Oおよびメモリの内容を読み出して、画面に表示します。

書式

DUMP_ [〈開始I/O番号〉][,〈終了I/O番号〉][,{B | W}]

説明

〈開始I/O番号〉、〈終了I/O番号〉で指定したI/O、メモリI/O、データメモリ、リンクメモリI/O、リンクデータメモリ、ファイルメモリの内容を画面に表示します。
〈開始I/O番号〉を省略すると、前回読み出したI/O番号の次の番号から表示します。
〈終了I/O番号〉を省略すると、〈開始I/O番号〉がワード指定のときは8ワード、バイト指定のときは16バイト、ビット指定のときは16ビットが表示されます。
また、〈終了I/O番号〉のI/Oの種類は省略することができます。
〈I/O番号〉の指定

ビット単位　X_ Y_ R_ L_

ワード単位　WX_ WY_ WR_ WL_ DT_ LD_ FL_

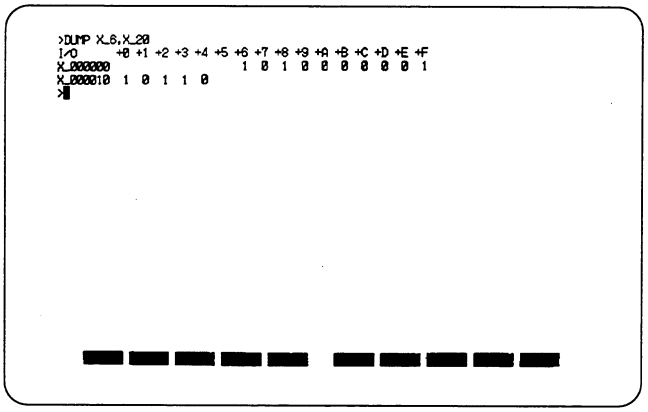
〈I/O番号〉をワード単位で指定した場合は、オプション〈B〉〈W〉を使用して、バイト単位での表示とワード単位の表示ができます。
オプション〈B〉〈W〉を省略するとワード表示となります。

記述例

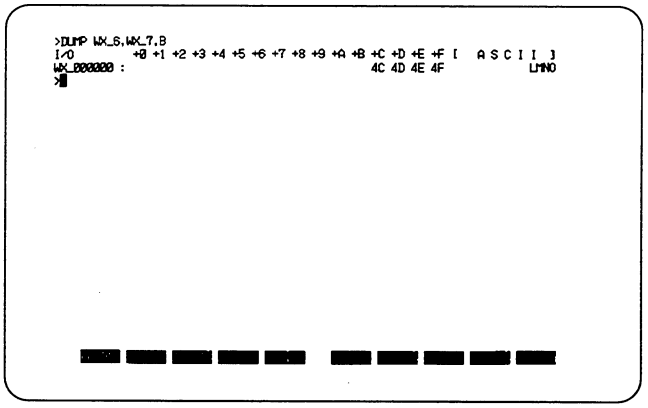
DUMP X_&M6,X_&M20
DUMP WX_6,WX_7

用例

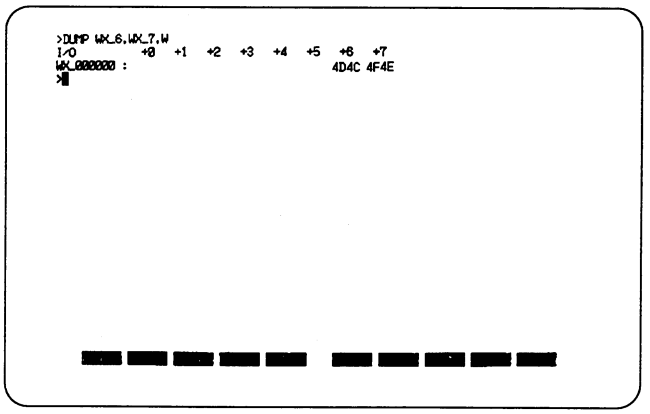
ビット単位の表示X_Y_R_L_を指定



バイト単位の表示WX_WY_WR_WL_DT_LD_FL_を指定



ワード単位の表示WX_WY_WR_WL_DT_LD_FL_を指定



DUMPC

オンラインコマンド

概要

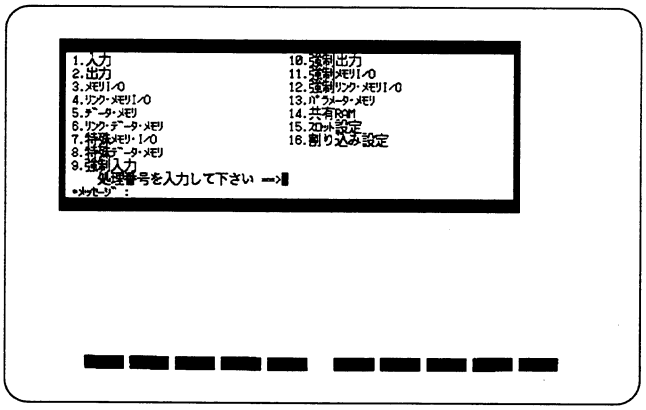
I/O、メモリ、その他の内容を一括して読み出して、スクリーン表示をします。

書式

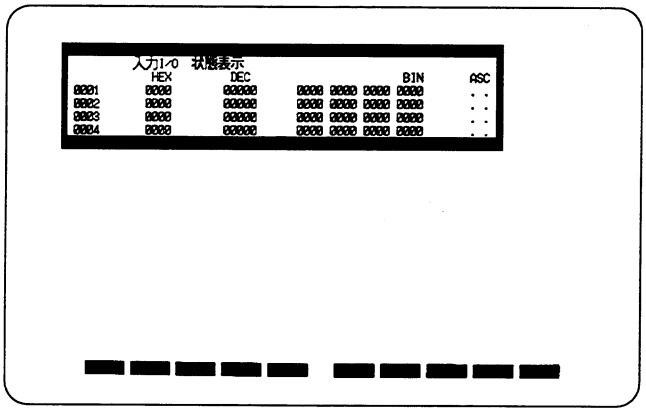
DUMPC

説明

DUMPCを実行すると、〈状態表示番号〉の選択ウィンドウが表示されます。



つづいて処理番号を入力すると、I/O、メモリ、その他の内容を一括して読み出して、スクロールが可能なスクリーン表示をします。下図は入力I/Oの状態表示を選択した場合のスクリーン表示の状態を示します。



↑ ↓ TAB BS キーでスクロールできます。
ESC キーでBASICのプロンプト（入力待ち状態）に戻ります。

関連命令

DUMPP

記述例

DUMPC

DUMPINT

オンラインコマンド

概要

割り込み設定状況の一覧を表示します。

書式

DUMPINT

説明

割り込みを発生する高機能ユニットについて、割り込み設定状況の一覧を表示します。
表示内容は、定義されている割り込み番号、割り込み処理ルーチンの行番号、現在の割り込み許可・禁止（ON/OFF）状態です。

記述例

DUMPINT

用例

```
>DUMPINT
Interrupt No. 0 Disable No req.
>
```

```
>DUMPINT
Interrupt No. 0 Enable No req.
>
```

DUMPP

オンラインコマンド

概要

I/O、メモリ、その他の内容を一括して読み出して、スクリーン表示および編集をします。

書式

DUMPP

説明

DUMPPを実行すると、〈状態表示番号〉の選択ウィンドウが表示されます。
つづいて処理番号を入力すると、I/O、メモリ、その他の内容を一括して読み出して、スクロールが可能なスクリーン表示、および編集をします。
ただし、「7.特殊メモリI/O」「8.特殊データメモリ」に対しては、実行できません。
↑ ↓ キーでスクロールできます。
← → TAB BS キーで訂正個所にカーソルを移動し、任意のキー入力ができます。
ESC キーでBASICのプロンプト（入力待ち状態）に戻ります。

関連命令

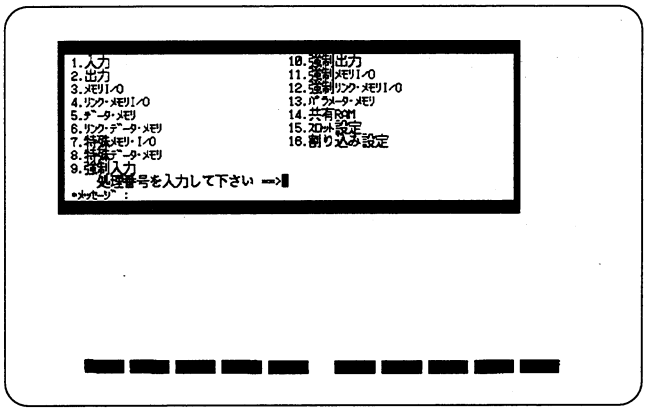
DUMPC

記述例

DUMPP

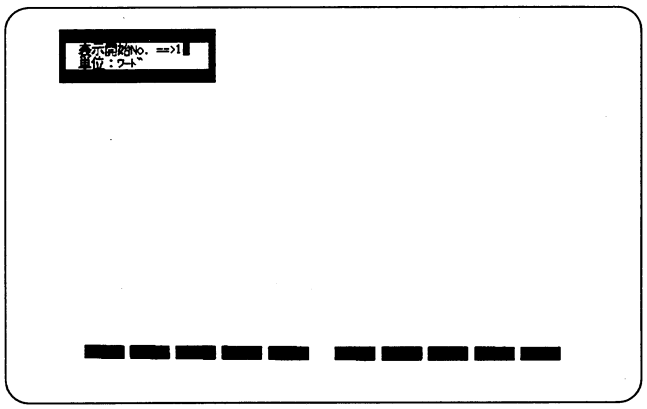
用例

〈状態表示番号〉の選択ウィンドウ

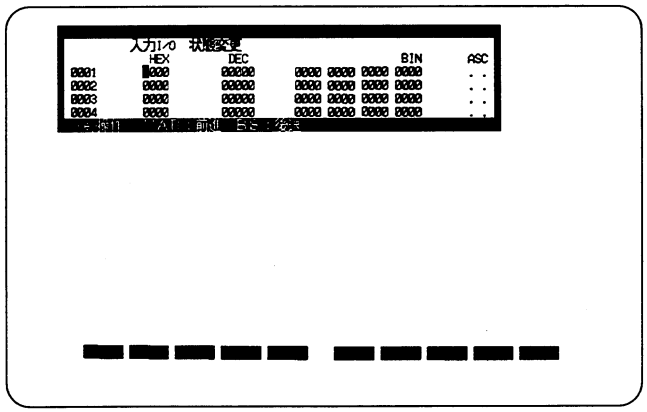


●1.入力

処理番号に1を入力する



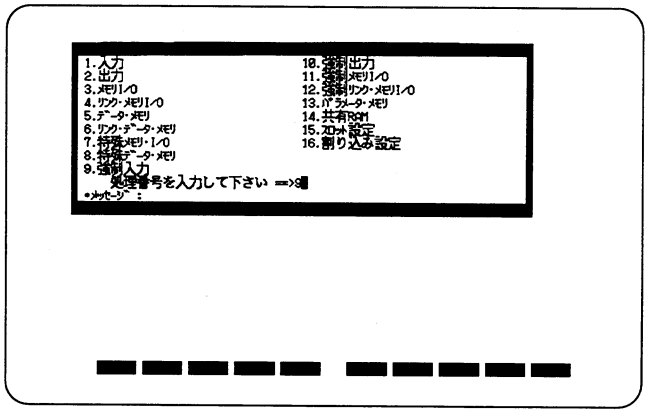
表示開始No.に0を入力する



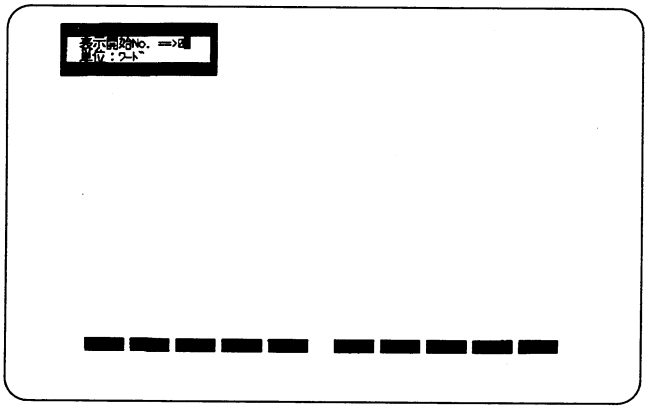
処理番号2～8も同様に行ないます。

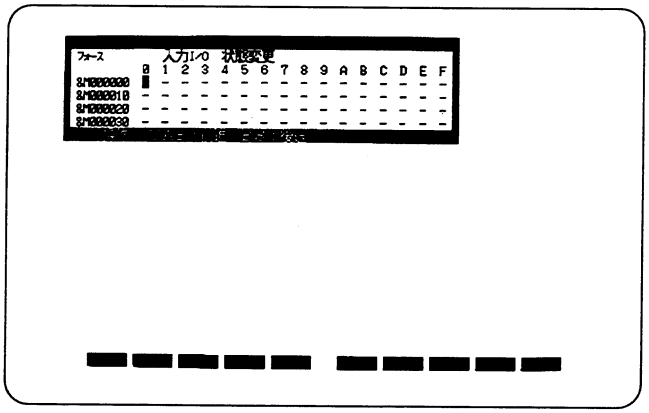
●9.強制入力

処理番号に9を入力する



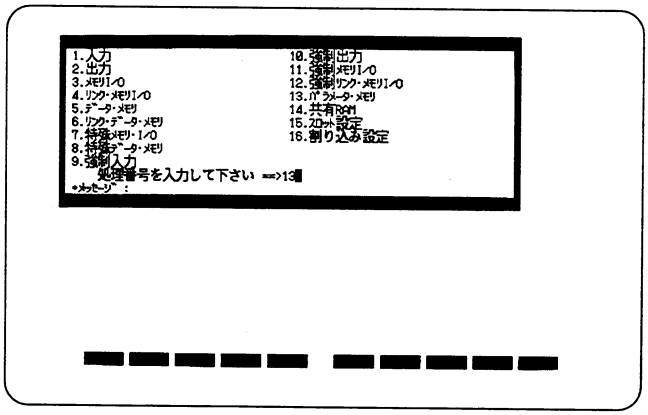
表示開始No.に0を入力する



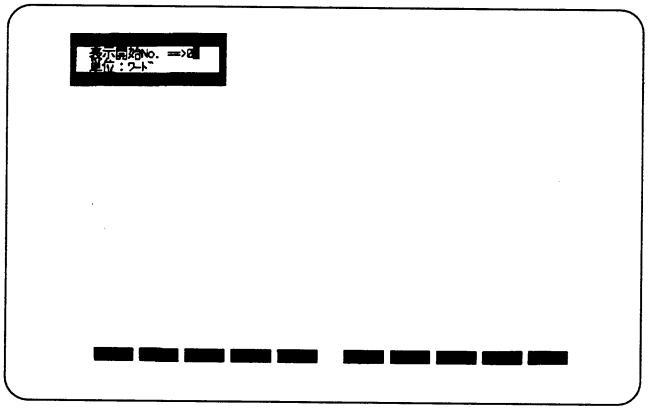


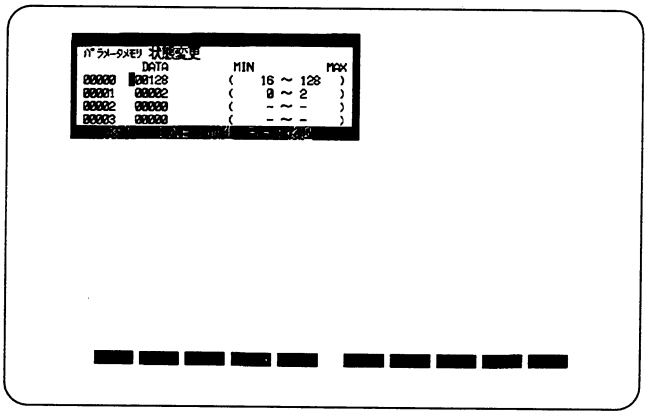
●13.パラメータメモリ

処理番号に13を入力する



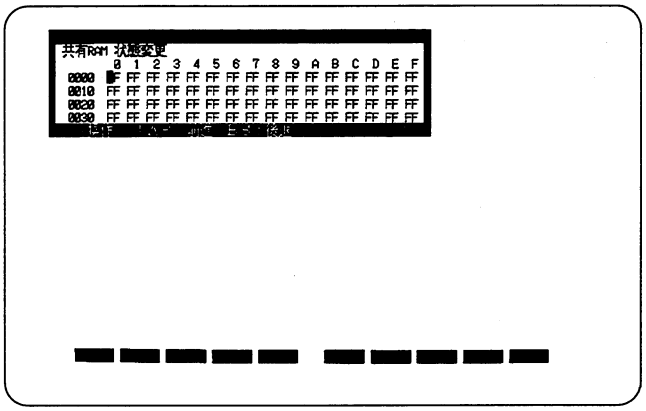
表示開始No.に0を入力する



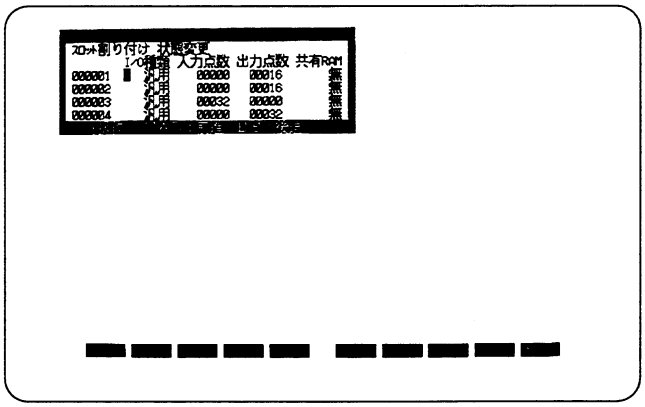


処理番号2～8も同様に行ないます。

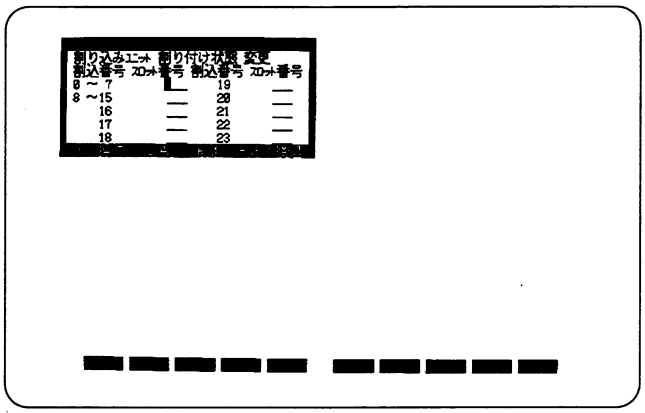
●14.共有メモリのダンプ



●15.スロット割り付けの状態表示



●16.割り込み設定
表示開始No.に16を入力する。



DUMPR

オンラインコマンド

概要

リモートI/O子局の占有スロット数とスロット割り付けの設定値を画面に表示します。

書式

DUMPR_〈親局番号〉[,〈子局番号〉]

説明

〈親局番号〉、〈子局番号〉で指定されたリモートI/O子局の占有スロット数とI/O割り付けの設定状態を画面に表示します。

〈子局番号〉が省略されたときは、指定された親局に接続されている各子局の占有スロット数を表示します。

〈親局番号〉と〈子局番号〉が指定されたときは、指定された子局の各スロットのスロット割り付けの設定値(機能番号)を表示します。

〈親局番号〉には、1～4を指定します。

〈子局番号〉には、1～32を指定します。

関連命令

PRMR,PRMR(),RIOCLR,RIOSET,SLOTR,SLOTR()

記述例

DUMPR1

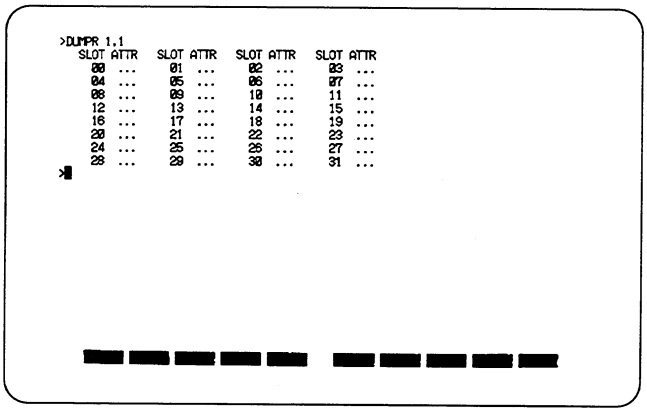
用例

```
>PRMR 1,1,3 ……親局番号1、子局番号1のリモートI/O子局に占有
                スロット数3を割りつけます
>SLOTR1,1,0,120 ……親局番号1、子局番号1のリモートI/O子局のス
                ロット0に16点入力ユニットを割り付けます
>RIOSET ……PRMR、SLOTRで設定された登録マップをリ
>PRINT PRMR(1,1)  モートI/Oの現在値マップに転送します。
3
>PRINT SLOTR(1,1,0)
120
>
```

```
>PRINT SLOTR(1,1,0)
120
>SLOTR 1,1,0,,220 ……親局番号1、子局番号1のリモートI/O子局のス
>RIOSET           ロット0にA/D、D/A変換ユニットを割り付け
>PRINT SLOTR(1,1,0)  ます
220
>RIOCLR ……リモートI/Oのスロット割り付けを初期化
>PRINT SLOTR(1,1,0)  します。
120
>
```

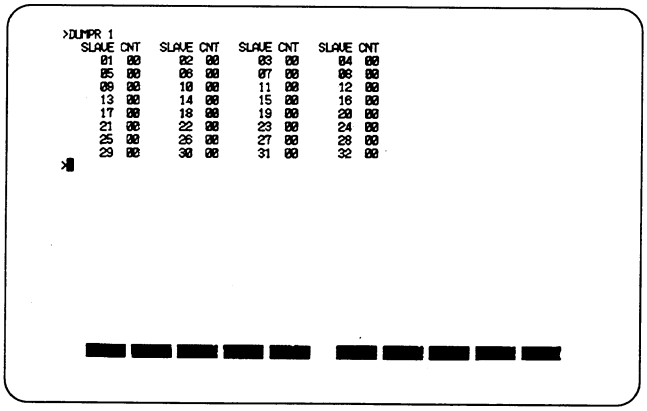
>DUMPR 1,1

親局番号1、子局番号1のリモートI/O子局の割り付け状態(占有スロット数とユニットの機能番号)を表示します。子局の割り付け状態(占有スロットと機能番号)を示します。



>DUMPR 1

親局番号1のリモートI/O親局に接続されている各子局の占有スロット数を表示します。子局番号1の占有スロット数が3であることを示します。



DWNLD

オンラインコマンド

概要

オブジェクトプログラムをパソコンのプログラムメモリからCPUユニットに転送します。

書式

DWNLD_ ["〈ドライブ番号〉:][〈パス名〉]〈ファイル名〉"]

説明

〈ファイル名〉で指定したオブジェクトプログラムを、BASICタイプCPUユニットに転送します。
対象となるファイルは、ファイル名に拡張子「.OBJ」が付いたオブジェクトファイルと、同じく拡張子「.SYM」が付いたシンボリックファイルです。
〈ファイル名〉には数字で始まるものは使えません。
これらのファイルは、あらかじめソースプログラムをコンパイルして作成しておきます。
〈ドライブ番号〉〈パス名〉〈ファイル名〉を省略するとカレントディレクトリ内の「SYSTMP.OBJ」と「SYSTMP.SYM」の2つのファイルを転送します。

関連命令

UPLD

記述例

```
>DWNLD "C:¥TEST"
>DWNLD
```

ECLR

ステートメント

概要

エラーステータスを初期化（クリア）します。

書式

ECLR

説明

「ONERR」命令により呼び出されたエラー処理ルーチンを終了し、エラーステータスを初期化します。
「RETURN」命令でエラー発生文の次の命令文に戻ります。

注意

エラーステータスを初期化せずにエラー処理ルーチンから戻ると、戻り先から再度、エラーを発生します。プログラムを記述するときは、エラー処理ルーチンの中で、エラーステータスを必ず初期化してください。

用例

「WAIT」命令によりエラー（タイムアウト）が発生したときに、画面に「"Time out error"」の文字列を表示する場合

```
100 FUNCTION SAMPLE
110 TMOUT 20
120 ONERR ER_SUB ……エラー処理ルーチンの宣言
 :
 :
190 WAIT SW(X_0)=1 ……「WAIT」中にエラーが発生すると
 :
 :
300 ER_SUB: ……プログラム「ER_SUB」に分岐する
310 IF ERR(0)〈〉1004 THEN END ……タイムアウトエラー以外ならば終了する
320 PRINT "Time out error" ……タイムアウトエラーならばメッセージを表示する
330 ECLR ……エラーを解除して
340 RETURN ……元のプログラムに復帰する
 :
 :
400 FEND
```

ENCO ( )

オンラインコマンド　ステートメント

概要

数値をエンコード変換します。

書式

ENCO(〈数式〉)

説明

指定した〈数式〉をエンコード変換し(log2 nを求め)、その数値を返します(「DECO」の逆関数)。
対象となる〈数式〉の種類は、整数・実数およびその型の変数・関数です。
〈数式〉に実数を指定すると、小数点以下が四捨五入され、2バイト整数として扱われます。
指定した〈数式〉の「型」と変換後の「型」は同じものになります。
〈数式〉の値と返す値は、以下のとおりです。
&B1～&B1000 0000 0000 0000 0000 0000 0000 0000
　→返す値:0～31 (&H0～&H1F)

関連命令

DECO()

用例

どのポートがONしているか調べる

```
>PRINT ENCO(INW(WY_0))
3 ……3番ポートがONしている
>
```

プログラミング例

```
100 FUNCTION SAMPLE
120 INTEGER A
130WAIT INW(WX_0)〈〉0
140 A=ENCO(INW(WX_0))
150 IF A=0 THEN GOTO SUB_X_0 ……X_0がONのときSUB_X_0に飛ぶ
160 IF A=1 THEN GOTO SUB_X_1 ……X_1がONのときSUB_X_1に飛ぶ
170 IF A=2 THEN GOTO SUB_X_2 ……X_2がONのときSUB_X_2に飛ぶ
 :
200 SUB_X_0: ……X_0がONのときの処理
 :
300 SUB_X_1: ……X_1がONのときの処理
 :
400 SUB_X_2: ……X_2がONのときの処理
 :
500 FEND
```

END

ステートメント

概要

プログラムを終了します。

書式

END

説明

実行中のプログラムを終了し、全ファイルを閉じます。

用例

「WAIT」命令によりエラー(タイムアウト)が発生したときに、画面に「"Time out error"」の文字列を表示する場合

```
100 FUNCTION SAMPLE
110 TMOUT 20
120 ONERR ER_SUB …… エラー処理ルーチンの宣言
 :
190 WAIT SW(X_0)=1 ……「WAIT」中にエラーが発生すると
 :
300 ER_SUB: ……プログラム「ER_SUB」に分岐する
310 IF ERR(0)〈〉1004 THEN END ……タイムアウトエラー以外ならば終了する
320 PRINT "Time out error" ……タイムアウトエラーならばメッセージを表示する
340 RETURN ……元のプログラムに復帰する
 :
400 FEND
```

ENTRY

ステートメント

概要

分割コンパイル時のソースファイル間グローバル変数を宣言します。

書式

ENTRY␣{BYTE | INTEGER | LONG | REAL | STRING}␣〈変数名〉[,〈変数名〉･･･]

説明

分割コンパイルで複数ソースファイルを作成する際に、ソースファイル間のグローバル変数を宣言（定義）します。

注意

1．〈変数名〉宣言する型にかかわらず、複数ソースファイルで重複して宣言しないでください。

2．1ソースファイル中またはLINK後の実行オブジェクトファイル中でのシンボルの総数はEXTERN宣言するシンボルの数を除いて1024個までです。 ただし、配列変数を使用している場合やFUNCTIONに引数を指定している場合は、配列の次元や引数の数だけ少なくなります。

関連命令

EXTERN

記述例

```
ENTRY INTEGER A,B,C
```

EOF( )

ステートメント

概要

ファイルの終了コードを返します。

書式

EOF([#]〈ファイル番号〉)

説明

〈ファイル番号〉で指定されたディスクファイルが終わりに達したかどうかを調べる関数です。

終わりに達していれば真（－1）、そうでなければ偽（0）を返します。

記述例

```
IF EOF(1) THEN GOTO SUB1
```

用例

```
100 FUNCTION SAMPLE
110 INTEGER A,B,C
 :
200 OPEN "INITDATA" FOR OUTPUT AS #1
210 A=1 : B=2 : C=3
220 PRINT #1,A,3,C  ……変数の初期値を「INITDATA」に設定する
230 CLOSE #1
 :
300 OPEN "INITDATA" FOR INPUT AS #1
310 INPUT #1,A  ……変数の初期値を「INITDATA」から読み出す
320 IF EOF(1) THEN GOSUB SUB ELSE GOTO 310
330 CLOSE #1
 :
400 END
410 SUB:
420 PRINT "読み出し終了"
430 RETURN
440 FEND
```

ERR( )

ステートメント

概要

エラーが発生したときにエラー番号を返します。

書式

ERR(〈ダミー引数〉)

説明

エラーが発生したときに、そのエラーコード番号を数値で返します。
発生したエラーをエラー処理ルーチンの中で調べるために使います。
〈ダミー引数〉には、一般に「0」を指定します。

用例

「WAIT」命令によりエラー（タイムアウト）が発生したときに、画面に「"Time out error"」の文字列を表示する場合

```
100 FUNCTION SAMPLE
110 TMOUT 20
120 ONERR ER_SUB ····· エラー処理ルーチンの宣言
 :
190 WAIT SW(X_0)=1 ····· 「WAIT」中にエラーが発生すると
 :
300 ER_SUB: ····· プログラム「ER_SUB」に分岐する
310 IF ERR(0)〈〉1004 THEN END ····· タイムアウトエラー以外ならば終了する
320 PRINT "Time out error" ····· タイムアウトエラーならばらばメッセージを表示する
340 RETURN ····· 元のプログラムに復帰する
 :
400 FEND
```

EXIT

ステートメント

概要

ループから強制的に脱出します。

書式

EXIT

説明

「FOR～NEXT」「DO～LOOP」「WHILE～WEND」のループ文から強制的に脱出します。
多重ループのときは、一番浅いループを脱出します。

注意

1つのグループに対してEXITを記述できるのは1つだけです。

EXIT

オフラインコマンド

概要

プログラム編集を終了します。

書式

EXIT

説明

プログラム編集を終了し、MS-DOSのコマンド入力待ち状態になります。

EXIT命令を実行すると、プログラムが保存されている場合はそのまま終了されます。プログラムが保存されていない場合は終了の確認メッセージが表示されますので、終了の場合は Y キーを押してください。N キーを押しますと再びFP-BASICに戻ります。

EXP( )

オンラインコマンド　ステートメント

概要

eを底にする指数関数の値を返します。

書式

EXP(〈数式〉)

説明

〈数式〉の指数関数の結果の値を返します。

〈数式〉の型の関係なく単精度の値を返します。

関連命令

LOG()

記述例

```
>PRINT EXP(2)
7.389056
>
```

EXTERN

ステートメント

概要

分割コンパイル時のソースファイル間グローバル変数またはタスク名の参照を宣言します。

書式

EXTERN␣ {BYTE | INTEGER | LONG | REAL | STRING | FUNCTION} {〈変数名〉 | 〈タスク名〉} [, {〈変数名〉 | 〈タスク名〉} ･･･]

説明

分割コンパイルで複数ソースファイルを作成する際に、外部変数（ソースファイル間グローバル変数）またはタスク名の参照を宣言します。

関連命令

ENTRY

注意

1ソースファイル中でEXTERN宣言できるシンボルの数は400個までです。 ただし、400個以上を越えてもエラーにはなりませんので注意してください。 また、1ソースファイル中またはLINK後の実行オブジェクトファイル中でのシンボルの総数はEXTERN宣言するシンボルの数を除いて1024個までです。 ただし、配列変数を使用している場合やFUNCTIONに引数を指定している場合は、配列の次元や引数の数だけ少なくなります。

記述例

```
EXTERN REAL A,B,C
EXTERN FUNCTION SUB1
```

用例

```
>LOAD "MAIN.PRG"
>LIST
100 FUNCTION MAIN
110 EXTERN FUNCTION INIT ･･･････････参照タスクの宣言
120 INTEGER I
130 ENTRY INTEGER WORK(100) ･･･････グローバル変数定義
140 ENTRY REAL A_SIN,A_COS ･･･････グローバル変数定義
150 XQT !2,INIT
160 FEND
>LOAD "INIT.PRG"
>LIST
100 FUNCTION INIT
110 INTEGER I
120 EXTERN INTEGER WORK(100) ･･･････参照変数の宣言
130 EXTERN REAL A_SIN,A_COS ･･･････参照変数の宣言
140 EXTERN FUNCTION RRI ･･･････････参照タスクの宣言
150 FOR I=0 TO 100
160 WORK(I)=I
170 NEXT
180 A_SIN=SIN(1):A_COS=COS(1)
190 XQT !3,RRI
200 FEND
>LOAD"RRI.PRG"
>LIST
100 FUNCTION RRI
110 INTEGER I
120 EXTERN INTEGER WORK(100) ･･･････参照変数の宣言
130 EXTERN REAL Z_SIN,A_COS ･･･････参照変数の宣言
140 FOR I=0 TO 100
150 PRINT WORK(I)
160 NEXT
170 PRINT A_SIN,A_COS
180 FEND
>COMPILE "MAIN"
COMPILE END
>COMPILE "INIT"
COMPILE END
>COMPILE "RRI"
COMPILE END
>FILES
FPB.EXE MAIN.PRG MAIN.OBJ
MAIN.SYM INIT.PRG INIT.OBJ INIT.SYM
RRI.PRG RRI.OBJ RRI.SYM
>LINK MAIN+INIT+RRI,OBJ
>FILES
FPB.EXE MAIN.PRG MAIN.OBJ
MAIN.SYM INIT.PRG INIT.OBJ INIT.SYM
RRI.PRG RRI.OBJ RRI.SYM
OBJ.OBJ OBJ.SYM
>DWNLD"OBJ"
>XQT
```

FILES

オンラインコマンド　オフラインコマンド

概要

ディスクに書き込まれているファイルを一覧表示します。

書式

FILES_ ["[〈ドライブ番号〉:][〈パス名〉][〈ファイル名〉.[〈拡張子〉]]["]

説明

〈ドライブ番号〉、〈パス名〉、〈ファイル名〉を指定して、ディスク上に書き込まれているファイルの一覧を画面に表示します。
〈ドライブ番号〉を省略するとカレントドライブが、〈パス名〉を省略するとカレントディレクトリが、それぞれ指定されます。
また、〈ファイル名〉には、ワイルドカード「＊」「?」を指定することもできます。

関連命令

LOAD,SAVE

記述例

```
FILES
FILES "A:¥"
FILES "A:¥PC"
```

用例

現在のディレクトリに登録されているファイルの名前を一覧表示する場合
（ワイルドカードを使用しない一覧表示の例）

```
>FILES
COMAND.COM  AUTOEXEC.BAT RSDRV.SYS  SPEED.EXE
FPBASIC.EXE PRINT.SYS   CONFIG.BAK  CONFIG.SYS
SYSTMP.PRG  FKEY.DEF    SYSMP.BIN   SYSTEMP.PRM
>
```

上記のファイルの中から「S･･･」で始まるファイル名だけを探索する場合（ワイルドカードを使用した一覧表示の例）

```
>FILES "S＊
SPEED.EXE SYSTMP.PRG SYSTEMP.BIN SYSTEMP.PRM
>
```

「B」ドライブの「RROG」ディレクトリの内容を表示する。

```
>FILES "B:¥PROG¥"
A0206.PRG  A0312.PRG  A0301.PRG  A0302.PRG
A0305.PRG  A0306.PRG  A0307.PRG  A0308.PRG
A0309.PRG  A0310.PRG  A0401.PRG  A0402.PRG
A0311.PRG  A0313.PRG  A0314.PRG  A0315.PRG
A0316.PRG  A0319.PRG  A0205.PRG
>
```

FIND

オンラインコマンドオフラインコマンド

概要

編集中のプログラムから文字列を検索します。

書式

FIND_ 〈文字列〉[,[〈検索開始行番号〉][,〈検索終了行番号]]

説明

プログラム中から指定された〈文字列〉を検索し、その行を表示します。
〈検索開始行番号〉と〈検索終了行番号〉が指定されたときはその範囲で検索します。
〈検索開始行番号〉だけが指定されたときは、その行からプログラムの最後までの範囲で検索します。
〈検索終了行番号〉だけが指定されたときは、プログラムの先頭からその行までの範囲で検索します。
〈文字列〉には、ワイルドカード（「?」「＊」）が指定できます。
「?」クエスチョンマークは任意の一文字に対応します。
「＊」アスタリスクは0個以上の任意の文字と対応します。

記述例

```
FIND WAIT
```

用例

```
>LIST
100 FUNCTION SAMPLE
110 INTEGER A
120 ST:
130 FOR A=TO 3 STEP 1
140 ON Y_&M10
150 WAIT 1
160 OFF Y_&M10
170 WAIT 1
180 NEXT A
190 END
200 FEND
>FIND WAIT

150 WAIT 1
170 WAIT 1
>
```

プログラム中から文字列「WAIT」を探索します。

FOR～NEXT

ステートメント

概要

命令文を繰り返し実行します。

書式

FOR␣〈変数〉=〈初期値〉␣TO␣〈終了値〉␣[STEP〈差分〉]～NEXT[〈変数〉]

説明

「FOR」文と「NEXT」文で囲まれた行の「命令文」を繰り返し実行します。

「命令文」は、指定した〈変数〉が〈初期値〉から〈差分〉の割合で〈終了値〉に達するまで繰り返されます。

〈差分〉に負の数を指定するときは、〈初期値〉を〈終了値〉より大きくします。〈差分〉は省略すると「1」が自動設定されます。

注意

「FOR～NEXT」の中に、さらに「FOR～NEXT」を記述（ネスティング）することもできます。

内側の「FOR～NEXT」は、外側の「FOR～NEXT」に完全に含まれている必要があります。

ネスティングは、最高10段まで可能です。

用例

プログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 WAIT SW(X_0)=1 ……ポート「X_0」がONになるまで待つ
130 FOR A=47 TO 32 STEP -1
140 ON Y_A ……ポート「Y_47～Y_32」までを順にON
150 NEXT                にする
160 GOTO 120
170 FEND
```

ネスティングの例

```
100 FUNCTION SAMPLE
110 INTEGER A,B,C
120 WAIT SW(X_0)=1 ……ポート「X_0」がONになるまで待つ
130 FOR A=2 TO 4 STEP 2 ……変数「A」は「2・4」と変化する
140 FOR B=0 TO 15 ……変数「B」は「0～15」と変化する
150 C=A＊16＋B
160 ON Y_C ……結果、指定されるポートは「Y_32」～
170 NEXT                「Y_47」と「Y_64」～「Y_79」を順にON
180 NEXT                にする
190 GOTO 120
200 FEND
```

FORCE

オンラインコマンド

概要

I/O、メモリの強制セットおよびリセットを設定します。

書式

FORCE␣[〈I/O番号〉=〈設定値〉]

説明

プログラムとは無関係に〈I/O番号〉で指定した外部入力、メモリI/Oを強制的に〈設定値〉の内容に設定（「ON（セット）・OFF（リセット）」）します。

〈I/O番号〉を省略すると、現在強制モードに設定されている〈I/O番号〉を画面に表示します。

〈I/O番号〉は、以下のとおり指定します。

外部入力 I/O	X_n	n=&M0～&M127F（0～2047）
外部出力 I/O	Y_n	n=&M0～&M127F（0～2047）
メモリ I/O	R_n	n=&M0～&M97F （0～1567）
リンクメモリ I/O	L_n	n=&M0～&M127F（0～2047）

〈設定値〉には、「1（ON）」または「0（OFF）」を指定します。

関連命令

RFORCE

記述例

```
FORCE Y_&M0=1
```

外部出力ユニット「Y_&M0」を強制的にONする場合

```
>FORCE Y_&M0=1
>
```

強制出力が設定されているI/Oを調べる場合

```
>FORCE
Y_: 0
>
```

FRE( )

オンラインコマンド　ステートメント

概要

ローカル変数領域の空エリアサイズを返します。

書式

FRE(!〈タスク番号〉)

説明

ローカル変数領域の空きエリアサイズを16バイト単位で返します。
(1=16バイト、2=32バイト、3=48バイト・・・)

記述例

```
PRINT FRE(!1)
```

FREV

オンラインコマンド　ステートメント

概要

グローバル変数領域の空エリアサイズを返します。

書式

FREV

説明

グローバル変数領域の空きエリアサイズをKバイト単位で返します
(1=1Kバイト、2=2Kバイト、3=3Kバイト・・・)

記述例

```
PRINT FREV
```

FUNCTION～FEND

ステートメント

概要

プログラムの開始および終了を宣言します。

書式

FUNCTION␣〈タスク名〉～FEND

説明

「FUNCTION」と「FEND」で囲まれたプログラムを〈タスク名〉で定義（宣言）します。

FP-BASICでは、1ファイル中に複数のプログラムを記述できます。また、記述されたプログラムは、マルチタスクで実行することができます。

〈タスク名〉は、それぞれのプログラムを呼び出す（実行する）ための名前です。

〈タスク名〉には、予約語と同じ文字列、Pで始まる文字列は使えません。

なお、英字の大文字と小文字を区別しませんので注意してください。

注意

1.最初に記述されたプログラムをタスク1で実行し、他のタスク1の中から呼び出され（実行され）るように記述します。

2.FP-BASIC上で1ファイル内で記述できるFUNCTION数は、Ver2.3以降は100個まで、Ver2.3未満は50個までです。

用例

マルチタスクプログラムの例

```
100 FUNCTION MAIN ……タスク1でメインプログラムを実行する
110 INTEGER A,B,C
120 XQT !2,SUB_1 ……タスク2で「SUB_1」を実行する
130 XQT !3,SUB_2 ……タスク3で「SUB_2」を実行する
140 XQT !4,SUB_2 ……タスク4で「SUB_2」を実行する
(このように同じプログラムを異なるタスクで実行することも可能)
 ⋮
200 FEND
210 '
220 FUNCTION SUB_1 ……プログラム名「SUB_1」(タスク2で実行される)
230 WAIT SW(R_0)=1
 ⋮
300 FEND
310 '
320 FUNCTION SUB_2 ……プログラム名「SUB_2」(タスク3、4で実行
330 A=MYTASK(0)          される)
 ⋮
400 FEND
```

GOSUB

ステートメント

概要

サブルーチンに実行を移します。

書式

GOSUB␣{〈ラベル名〉|〈行番号〉}

説明

指定した〈ラベル名〉または〈行番号〉の「行」から始まり、「RETURN」命令で終わるサブルーチンを実行します。
サブルーチンを実行したあとは、GOSUB文の次の行からプログラムを続行します。
サブルーチンの中から、さらにサブルーチンを呼び出すこと（ネスティング）もできます。

注意

1.分岐先は、行番号またはラベルで指定しますが、同一タスクブロック内でなければなりません（タスクブロックを越える分岐は不可）。
2.指定できる〈行番号〉の範囲は「1～32767」の数値です。
3.分岐先行にラベルを使用する場合、分岐先のラベルの後に「:（コロン）」を付けて記述します。
4.GOSUB文のネスティングは、最高8段まで可能です。
5.実行時にBASICスタックオーバーフロー（ERROR2004）が発生した場合は、MAP命令を参照し、このエラーの発生したタスクのスタック領域を増やしてください。
6.1ファイル内でGOTO,GOSUB,ON GOTO,ON GOSUB, ON ERRを合わせて600個まで記述できます。
また指定できるラベルの数は400個までです。

関連命令

ON～GOSUB,RETURN

記述例

```
GOSUB 160
GOSUB SUB
```

用例

プログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 MAIN:
130 GOSUB 160 ……160行へ飛ぶ
140 GOTO MAIN
150 '
160 A=INW(WX_0) ……「WX_0」の内容を調べる
170 IF A=0 THEN GOSUB SUB 0ならばラベル「SUB」へ飛ぶ
180 OUTW WY_2,A
190 RETURN ……元のルーチンに復帰する
200 '
210 SUB: ……ラベル「SUB」はこの行
220 A=INW(WX_1)
230 RETURN ……元のルーチンに復帰する
240 FEND
```

GOTO

ステートメント

概要

プログラムの実行を分岐します。

書式

GOTO␣{〈ラベル名〉|行番号〉}

説明

指定した〈ラベル名〉または〈行番号〉へジャンプして、プログラムを実行します。
指定した〈ラベル名〉または〈行番号〉がプログラム中に存在しない場合は、エラーになります。

注意

1.GOTO命令により分岐できる分岐先は、同一のタスクブロック内（FUNCTION～FENDの中）でなければなりません（タスクブロックを越える分岐は不可）。
2.指定できる〈行番号〉の範囲は、1～32767の整数です。
3.分岐先行にラベルを使用する場合、分岐先のラベルの後に「:（コロン）」に付けて記述します。
4.1ファイル内でGOTO,GOSUB,ONGOTO,ON GOSUB,ONERRを合わせて600個まで記述できます。また指定できるラベルの数は400個までです。

関連命令

ON～GOTO

記述例

```
GOTO 180
GOTO START
```

用例

プログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 START: ……ラベル「START」はこの行
130 A=INW(WX_0) ……「WX_0」の内容を調べる
140 IF A=0 THEN GOTO 180 ……0ならば180行に分岐する
150 OUTW WY_2,A
160 GOTO START ……ラベル「START」に分岐する
170 '
180 A=INW(WX_1) ……「WX_1」の内容を調べる
190 OUTW WY_2,A ……WX_1の内容をWY_2へ出力する
200 GOTO START ……ラベル「START」に分岐する
210 FEND
```

HALT

ステートメント

概要

実行中のタスクを一時停止します。

書式

HALT␣[!〈タスク番号〉]

説明

現在実行中のタスクを一時停止します。
〈タスク番号〉を省略したときは、タスク1を一時停止します。プログラム中でステートメントとして使用するときは、常に自機が対象となります。
一時停止を解除するには、「RESUME」命令を使います。

関連命令

RESUME

記述例

```
HALT !2←タスク2を一時停止する。
```

用例

実行中のプログラムのタスク1を一時停止する

```
>HALT
```

実行中のプログラムのタスク2を一時停止する

```
>HALT !2
```

HCDATA( )

オンラインコマンド　ステートメント

概要

高速カウンタユニットの経過値を返します。

書式

HCDATA(〈スロット番号〉,〈チャンネル番号〉)

説明

高速カウンタユニットの経過値を読み出し返します。
この値は、ユニットの共有メモリのワードアドレス0、1または8、9の値です。
〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、0または1を指定します。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。

記述例

```
A=HCDATA(2,1)
```

用例

1000パルスカウントするとC=PがONし、経過値を画面に表示します。

```
100 FUNCTION SAMPLE
110 HCRESET 1,0
120 HCISET 1,0,0
130 HCPSET 1,0,1000
140 HCOUTEN 1,0,1
150 PRINT HCDATA(1,0) ……スロット番号1,チャンネル番号0の経過値
160 GOTO 150                を画面に表示します
170 FEND
```

HCFSET

オンラインコマンド　ステートメント

概要

高速カウンタユニットの入力フィルタ時定数を設定します。

書式

HCFSET␣〈スロット番号〉,〈チャンネル番号〉,〈入力フィルタ時定数〉

説明

高速カウンタユニットの入力フィルタ時定数を設定します。
この設定値は、ユニットの共有メモリのワードアドレス4または12に書き込まれます。
〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、0または1を指定します。
〈入力フィルタ時定数〉は、以下のとおり指定します。

0:100kcps
1:50kcps
2:25kcps
3:8kcps

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。

記述例

```
HCFSET 4,1,1
```

HCISET

オンラインコマンド　ステートメント

概要

高速カウンタユニットの初期値を書き込みます。

書式

HCISET_〈スロット番号〉,〈チャンネル番号〉,〈初期値〉

説明

高速カウンタユニットの〈チャンネル番号〉の初期値を設定します。
初期値は24ビット整数で−16,777,216～＋16,777,215の範囲で設定できます。この設定は、ユニットの共有メモリのワードアドレス0、1または8、9に書き込まれます。
〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、0または1を指定します。
〈初期値〉には、24ビット整数（−16,777,216～＋16,777,215）を指定します。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター一覧表をご覧ください。

記述例

```
HCISET 2,1,−1000
```

用例

1000パルスカウントするとC=PがONし、経過値を画面に表示します。

```
100 FUNCTION SAMPLE
110 HCRESET 1,0
120 HCISET 1,0,0 ····· スロット番号1,チャンネル番号0の初期値
130 HCPSET 1,0,1000        を0にします
140 HCOUTEN 1,0,1
150 PRINT HCDATA(1,0)
160 GOTO 150
170 FEND
```

HCOUTEN

オンラインコマンド　ステートメント

概要

高速カウンタユニットの制御出力（C=P）（C>P）を許可します。

書式

HCOUTEN_〈スロット番号〉,〈チャンネル番号〉,〈許可/禁止〉

説明

高速カウンタユニットの出力ポートY_&M11またはY_&M13をONして、〈チャンネル番号〉の制御出力を許可します。
〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、0または1を指定します。
〈許可/禁止〉は、以下のとおり指定します。
　0:禁止
　1:許可

参照

出力ポートY_&M11およびY_&M13の割り付け内容については、巻末の高機能ユニットパラメーター一覧表をご覧ください。

記述例

```
HCOUTEN 4,0,1
```

用例

1000パルスカウントするとC=PがONし、経過値を画面に表示します。

```
100 FUNCTION SAMPLE
110 HCRESET 1,0
120 HCISET 1,0,0
130 HCPSET 1,0,1000
140 HCOUTEN 1,0,1 ····· スロット番号1,チャンネル番号0の
150 PRINT HCDATA(1,0)       制御出力を許可します
160 GOTO 150
170 FEND
```

HCPSET

オンラインコマンド　ステートメント

概要

高速カウンタユニットの目標値を書き込みます。

書式

HCPSET_〈スロット番号〉,〈チャンネル番号〉,〈目標値〉

説明

高速カウンタユニットの〈チャンネル番号〉の目標値を設定します。目標値は24ビット整数で−16,777,216〜＋16,777,215の範囲で設定できます。
この設定値は、ユニットの共有メモリのワードアドレス2、3または10、11に書き込まれます。
〈スロット番号〉には、0〜31を指定します。
〈チャンネル番号〉には、0または1を指定します。
〈目標値〉には、24ビット整数（−16,777,216〜＋16,777,215）を指定してください。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメータ一覧表をご覧ください。

記述例

```
HCPSET 2,1,5000
```

用例

1000パルスカウントするとC=PがONします。経過値を画面に表示します。

```
100 FUNCTION SAMPLE
110 HCRESET 1,0
120 HCISET 1,0,0
130 HCPSET 1,0,1000 ……スロット番号1,チャンネル番号0の目標値を1000にします
140 HCOUTEN 1,0,1
150 PRINT HCDATA(1,0)
160 GOTO 150
170 FEND
```

HCRESET

オンラインコマンド　ステートメント

概要

高速カウンタユニットのカウンタリセットをします。

書式

HCRESET_〈スロット番号〉,〈チャンネル番号〉

説明

高速カウンタユニットの出力ポートY_&M10またはY_&M12をONして、カウンタの経過値、目標値、入力フィルタ時定数をクリアします。
またこのとき、入力ポートX_&M0〜X_&M6は、すべてOFFします。
〈スロット番号〉には、0〜31を指定します。
〈チャンネル番号〉には、0または1を指定します。

参照

出力ポートY_&M10〜Y_&M12および入力ポートX_&M0〜X_&M6の割り付け内容については、巻末の高機能ユニットパラメータ一覧表をご覧ください。

記述例

```
HCRESET 4,1
```

用例

1000パルスカウントするとC=PがONし、経過値を画面に表示します。

```
100 FUNCTION SAMPLE
110 HCRESET 1,0 ……スロット番号1,チャンネル番号0をリセットします
120 HCISET 1,0,0
130 HCPSET 1,0,1000
140 HCOUTEN 1,0,1
150 PRINT HCDATA(1,0)
160 GOTO 150
170 FEND
```

HCSTAT( )

オンラインコマンド　ステートメント

概要

高速カウンタユニットの動作状態を返します。

書式

HCSTAT(〈スロット番号〉,〈チャンネル番号〉)

説明

高速カウンタユニットの現在の動作状態を返します。
この値は、ユニットの入力ポートX_&M0～X_&M2またはX_&M4～X_&M6の値です。
返す値は、以下のとおりです。

ビット0:C=P一致フラグのとき1
ビット1:C>P比較フラグのとき1
ビット2:オーバー、アンダーフローフラグ経過値が範囲外のとき1

〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、0または1を指定します。

参照

入力ポートX_&M0～X_&M6の割り付け内容については、巻末の高機能ユニットパラメータ一覧表をご覧ください。

記述例

```
I=HCSTAT(2,1)
```

用例

```
PRINT HCSTAT(2,1)
```

HEX$( )

オンラインコマンド　ステートメント

概要

数値を16進表記の文字列に変換します。

書式

HEX$(〈数式〉)

説明

指定した〈数式〉を16進表記の文字列に変換して返します。
対象となる〈数式〉の種類は、整数・実数とその変数・関数です。
〈数式〉に実数が指定された場合は、小数点以下が四捨五入され、2バイト整数として扱われます。

関連命令

BIN$(),MEW$(),OCT$()

記述例

```
A$=HEX$(123)
```

用例

2進数「&B1000000000」を16進表記の文字列に変換する場合

```
>PRINT HEX$(&B1000000000)
200
>
```

8進数「&O200」を16進表記の文字列に変換する場合

```
>PRINT HEX$(&O200)
80
>
```

10進数「200」を16進表記の文字列に変換する場合

```
>PRINT HEX$(200)
CB
```

プログラミング例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 STRING B$
130 A=INL(WX_0) ……「WX_0」ポートの内容を
140 OUTL WY_2,A ……「WX_2」ポートに出力し
150 B$=HEX$(A)  ……16進表記に変換して
160 PRINT B$ ……画面に表示する
170 FEND
```

IF～THEN～ELSE

ステートメント

概要

条件式の結果により、プログラムの実行を制御します。

書式

IF␣〈条件式〉␣THEN␣〈実行文(1)〉␣[ELSE␣〈実行文(2)〉]

説明

〈条件式〉の結果が真のとき、〈実行文(1)〉を実行し、偽のとき〈実行文(2)〉を実行します。
ELSE以降の記述を省略したとき、〈条件式〉が「偽」になると、次の行の命令文を実行します。
「IF THEN ELSE」は、必ず1行に記述しなければなりません。
注意
1.「IF」命令のネスティングはできません。ネスティングが必要な複雑な条件判断には、「IFB」命令を使用してください。
2.条件式には、関数および演算子(四則演算・論理演算・比較演算)が記述できます。

関連命令

IFB～THEN～ELSE

用例

A=1のときWY_1に1を出力、A≠1のときWY_1に2を出力します。

```
100 FUNCTION SAMPLE
110 INTEGER A
120 ST:
130 A=INW(WX_0)
140 JUDGE:
150 IF A=1 THEN OUTW WY_1,1 ELSE OUTW WY_1,2
160 GOTO ST
170 FEND
```

IFB～THEN～ELSE

ステートメント

概要

条件式の結果により、プログラムの実行を制御します。

書式

IFB␣〈条件式〉␣THEN～〈実行文(1)〉～[ELSEIF～]
ELSE～〈実行文(2)〉～ENDIF

説明

〈条件式〉の結果が真(「0」以外)のとき、〈実行文(1)〉を実行し、偽(「0」)のとき〈実行文(2)〉を実行します。
「IFB」命令はネスティングが可能なので、複雑な条件判断をわかりやすく記述できます(用例参照)。
それぞれの実行文は、複数行での記述、およびマルチステートメントでの記述も可能です。
注意
「IFB」のネスティングは、最高9段まで可能です。
(ただし、ELSEIFも1段と数えます。)

関連命令

IF～THEN～ELSE

用例

```
100 FUNCTION SAMPLE
110 INTEGER I
120 FOR I=1 TO 5
130 IFB I=1 THEN
140 ON Y_1
150 WAIT 1 ……Iの値が1の時実行します。
160 ELSEIF I=2 THEN
170 ON Y_2
180 WAIT 1 ……Iの値が2の時に実行します。
190 ELSE
200 ON Y_7
210 WAIT 1 ……Iの値が1と2でない時に実行します。
220 ENDIF
230 NEXT I
240 FEND
```

INC

ステートメント

概要

変数をインクリメント（カウントUP）します。

書式

INC_〈数値変数〉[,〈数式〉]

説明

〈数値変数〉の値を、〈数式〉の値分加算（インクリメント）します。
〈数式〉が省略されたときは、1を加算します。

関連命令

DEC

記述例

INC A

用例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 A=1
130 WHILE A<11
140 PRINT A
150 INC A ……変数Aに1を加算します。
160 WEND
170 FEND
```

IND( )

オンラインコマンド　ステートメント

概要

ワード指定したI/O、メモリの内容を32ビット単位で返します。

書式

IND(〈ワード指定I/O番号〉)

説明

〈ワード指定I/O番号〉で指定した外部入力I/O、外部出力I/O、メモリI/O、リンクメモリI/O、データメモリ、リンクデータメモリの内容を32ビットの数値で返します。
〈ワード指定I/O番号〉は1ワード単位で記述しますので、返される数値の上位16ビットは、〈ワード指定I/O番号〉で指定した次の外部入力I/O、メモリI/O、リンクメモリI/O、データメモリ、リンクデータメモリを示します。
〈ワード指定I/O番号〉は、以下のとおり指定します。

外部入力 I/O	WX_n	n=0～127
外部出力 I/O	WY_n	n=0～127
メモリ I/O	WR_n	n=0～97
リンクメモリ I/O	WL_n	n=0～127
データメモリ	DT_n	n=0～2047
リンクデータメモリ	LD_n	n=0～255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

注意

〈ワード指定I/O番号〉は、1ワード単位で指定します。したがって、「IND」命令でWX_0を指定すると、X_&M0～X_&MFの16個ビットに、次の16ビット（X_&M10～X_&M1F）を合わせて32ビットのデータとして処理します。

関連命令

INH(),INL(),INW()

記述例

A=IND(DT_100)

用例

ポート「WX_0」の内容をポート「WY_4」に出力するプログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 A=IND(WX_0) ……「WX_0」と「WX_1」の内容をAに代入
130 OUTD WY_4,A ……「WY_4」と「WY_5」に出力する
140 GOTO 120
150 FEND
```

INH( )

オンラインコマンド　ステートメント

概要

ワード指定したI/O、メモリの上位8ビットの内容を返します。

書式

INH(〈ワード指定I/O番号〉)

説明

〈ワード指定I/O番号〉で指定した外部入力I/O、外部出力I/O、メモリI/O、リンクメモリI/O、データメモリ、リンクデータメモリの上位8ビットの内容を数値(1バイト整数)で返します。

〈ワード指定I/O番号〉は、以下のとおり指定します。

外部入力 I/O	WX_n	n=0～127
外部出力 I/O	WY_n	n=0～127
メモリ I/O	WR_n	n=0～97
リンクメモリ I/O	WL_n	n=0～127
データメモリ	DT_n	n=0～2047
リンクデータメモリ	LD_n	n=0～255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

注意

〈ワード指定I/O番号〉は、1ワード単位で指定します。例えば、WX_0の上位8ビットは、X_&M0～X_&MF (X_0～X_15) の16個のポートの上位8個、X_&M8～X_&MF (X_8～X_15) を示します。

関連命令

IND(),INL(),INW()

記述例

```
A=INH(DT_100)
```

用例

ポート「WX_0」の上位8ビットの内容をポート「WY_2」の上位8ビットに出力するプログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 A=INH(WX_0) ……「WX_0」の上位8ビットの内容をAに代入
130 OUTD WY_2,A ……「WY_2」の上位8ビットに出力する
140 GOTO 120
150 FEND
```

INL( )

オンラインコマンド　ステートメント

概要

ワード指定したI/O、メモリの下位8ビットの内容を返します。

書式

INL(〈ワード指定I/O番号〉)

説明

〈ワード指定I/O番号〉で指定した外部入力I/O、外部出力I/O、メモリI/O、リンクメモリI/O、データメモリ、リンクデータメモリの下位8ビットの内容を数値(1バイト整数)で返します。

〈ワード指定I/O番号〉は、以下のとおり指定します。

外部入力 I/O	WX_n	n=0～127
外部出力 I/O	WY_n	n=0～127
メモリ I/O	WR_n	n=0～97
リンクメモリ I/O	WL_n	n=0～127
データメモリ	DT_n	n=0～2047
リンクデータメモリ	LD_n	n=0～255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

注意

〈ワード指定I/O番号〉は、1ワード単位で指定します。例えば、WX_0の下位8ビットは、X_&M0～X_&MF (X_0～X_15) の16個のポートの下位8個、X_&M0～X_&M7 (X_0～X_7) を示します。

関連命令

IND(),INH(), INW()

記述例

```
A=INL(WX_2)
```

用例

ポート「WX_0」の下位8ビットの内容をポート「WY_2」の下位8ビットに出力するプログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 A=INL(WX_0) ……「WX_0」の下位8ビットの内容をAに代入
130 OUTD WY_2,A ……「WY_2」の下位8ビットに出力する
140 GOTO 120
150 FEND
```

INPUT

ステートメント

概要

ファイルからデータを入力し、変数に代入します。

書式

INPUT␣[#]〈ファイル番号〉,〈変数〉,[,〈変数〉･･･]

説明

〈ファイル番号〉で指定されたディスクファイルからデータを入力し、変数に代入します。
〈変数〉が複数指定されているときは、ファイルに書き込まれているデータが数値変数に代入される場合、
「,」「空白」「キャリッジリターン (CHR$ (13))」が区切り記号になります。

注意

使用できる文字変数は、127文字以下です。

関連命令

PRINT #

記述例

```
INPUT #1,B,B$
```

用例

```
100 FUNCTION SAMPLE
110 INTEGER A,B,C
 :
 :
200 OPEN "INITDATA" FOR OUTPUT AS #1
210 A=1;B=2;C=3
220 PRINT #1,A,B,C ･･･････････････････････変数の初期値を「INITDATA」に
230 CLOSE #1                              設定する
 :
 :
300 OPEN "INITDATA" FOR INPUT AS #1
310 INPUT #1,A,B,C ･･･････････････････････変数の初期値を「INITDATA」よ
320 CLOSE #1                              り読み出す
 :
 :
420 OPEN "TEST.DAT" FOR APPEND AS #1
430 PRINT #1,DATE$(),TIME$(),A,B,C ･･･････変数の値を「TEST.DAT」に記録
440 CLOSE #1                              する
 :
 :
500 FEND
```

INPUT$( )

ステートメント

概要

ファイルから指定された文字数分の文字列を取り出して返します。

書式

INPUT$(〈文字数〉[,[#]〈ファイル番号〉])

説明

〈ファイル番号〉で指定されたディスクファイルから (文字数) 分の文字列を読み出してその値を返します。
〈文字数〉には、文字列の長さをバイト数で指定します。

記述例

```
A$=INPUT$(5,1)
```

INPUT%

オンラインコマンド　ステートメント

概要

高機能ユニットの共有メモリからデータを読み出し、〈変数〉に代入します。

書式

INPUT%_〈スロット番号〉,〈高機能ユニットの先頭アドレス〉,〈変数〉,・・・

説明

〈スロット番号〉で指定された高機能ユニットの共有メモリの〈先頭アドレス〉からデータを読み出し、〈変数〉に代入します。
〈変数〉が数値変数のときは、そのデータ型で代入されます。
〈変数〉が文字変数のときは、80バイト（文字）単位で代入されます。
〈スロット番号〉には、0～31を指定します。
〈先頭アドレス〉には、0～2047を指定します（バイト単位）。

注意

INPUT%命令は共有メモリのアドレスをバイト単位で指定します。

バイトアドレス	0	1	2	3	4	5	6	7
ワードアドレス	0		1		2		3	

参照

高機能ユニットの共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。

関連命令

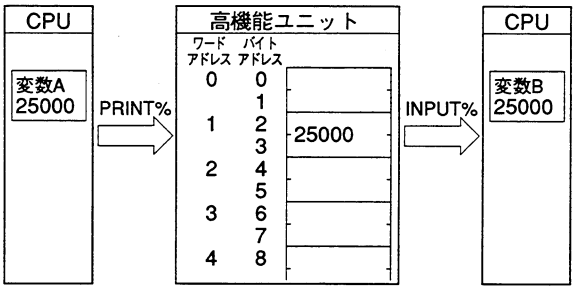
PRINT%,READ%,WRITE%

記述例

```
INPUT% 4,0,A
```

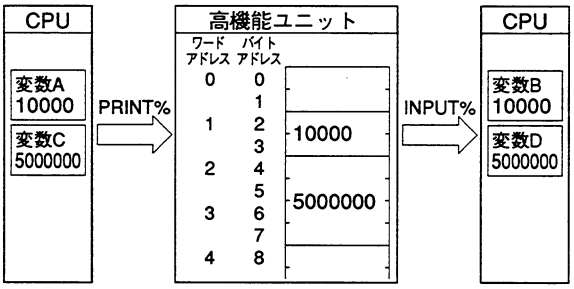
用例

変数の内容を高機能ユニットの共有メモリに対し書き込みを行い、そのデータを別の変数に読み出しを行います。




```
100 FUNCTION SAMPLE
110'
120'** PRINT%-INPUT%-1 **
130'
140 INTEGER A,B
150 A=25000
160 PRINT% 2,2,A
170'
180 INPUT% 2,2,B
190 PRINT B
200'
210 FEND
```

2つの変数の内容を高機能ユニットの共有メモリに対し書き込みを行い、そのデータを別の変数に読み出しを行います。




```
100 FUNCTION SAMPLE
110'
120'** PRINT%-INPUT%-2 **
130 INTEGER A,C
140 LONG B,D
150 A=10000 ;B=5000000
160 PRINT% 2,2,A,B
170'
180 INPUT% 2,2,C,D
190 PRINT C
200 PRINT
210'
220 FEND
```

INPUTR

オンラインコマンド　ステートメント

概要

リモートI/O子局の高機能ユニットの共有メモリからデータを読み出し、変数に代入します。

書式

INPUTR␣〈親局番号〉,〈子局番号〉,〈スロット番号〉,〈先頭アドレス〉,〈変数〉・・

説明

〈親局番号〉、〈子局番号〉、〈スロット番号〉で指定した高機能ユニットの共有メモリの〈先頭アドレス〉からデータを読み出し、〈変数〉に代入します。
〈変数〉が整数型または実数型のときは1、2または4バイト単位で読み出されます。
〈変数〉が文字型のときは80バイト単位で読み出されます。
〈親局番号〉には、1～4を指定します。
〈子局番号〉には、1～32を指定します。
〈スロット番号〉には、0～31を指定します。
〈先頭アドレス〉には、0～2047を指定します（バイト単位）。

参照

高機能ユニットの共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。

関連命令

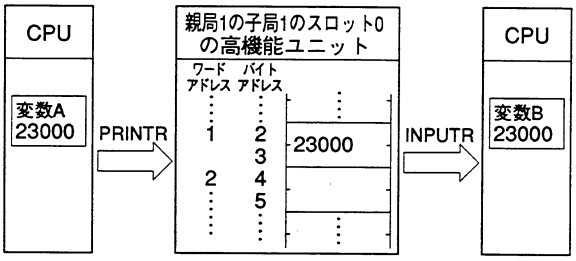
PRINTR,READR,WRITER

記述例

```
INPUTR 1,1,0,0,A
```

用例

変数の内容を親局1の子局1のスロット0の高機能ユニットの共有メモリに対し書き込み、別の変数に読み込みます。




```
100 FUNCTION SAMPLE
110'
120'** PRINT-INPUTR-1 **
130'
140 INTEGER A,B
150 A=23000
160 PRINTR 1,1,0,2,A
170'
180 INPUTR 1,1,0,2,B
190 PRINT B
200 FEND
```

注意

INPUTR命令は共有メモリのアドレスをバイト単位で指定します。

バイトアドレス	0	1	2	3	4	5	6	7
ワードアドレス	0		1		2		3	

INSTR( )

オンラインコマンド　ステートメント

概要

文字列中の何文字目に指定した文字列があるかを返します。

書式

INSTR([〈開始位置〉],〈検索される文字列〉,〈見つける文字列〉)

説明

〈検索される文字列〉の〈開始位置〉から〈見つける文字列〉を検索し、その位置を〈検索される文字列〉の先頭からのバイト数で返します。

注意

検索できる文字列は、127文字以下です。

記述例

```
A=INSTR(1,"ABCDEFG","C")
```

用例

```
100 FUNCTION SAMPLE
120 PRINT ASC("A")
130 PRINT CHR$(65)
140 PRINT LEN("ABCDE")
150 PRINT RIGHT$("ABCDE",1)
160 PRINT LEFT$("ABCDE",1)
170 PRINT MID$("ABCDE",3,1)
180 PRINT "X",SPACE$(3),"Y"
190 PRINT INSTR(1,"ABCDE","B") ……文字列"ABCDE"から文字列"B"
200 END                          が何バイト目にあるかを画面に
210 FEND                         表示します。
```

INT( )

オンラインコマンド　ステートメント

概要

数値の小数点以下を切り捨てます。

書式

INT(〈数式〉)

説明

指定した〈数式〉の小数点以下を切り捨て、その数値を返します。

対象となる〈数式〉の種類は実数とその型の変数・関数です。また整数を代入してもエラーにはなりません。

記述例

```
INT(1) CLR
```

用例

「12.3」の小数点以下を切り捨てた値を画面表示する場合

```
>PRINT INT(12.3) ……「12.3」の小数点以下を切り捨てて出力させる
12 ……結果の「12」が表示される
>
```

「−12.99」の小数点以下を切り捨てた値を画面表示する場合

```
>PRINT INT(-12.99) ……「-12.99」の小数点以下を切り捨てて出力させる
-12 ……結果の「-12」が表示される
>
```

プログラミング例

```
100 FUNCTION SAMPLE
120 INTEGER A,C
130 REAL B
140 A=INW(WR_0) …… メモリI/Oの内容を読み込み
150 B=A/2 …… その値を「2」で割り
160 C=INT(B) …… その値の小数点以下を切り捨てて
170 OUTW WY_2,C …… 出力する
180 FEND
```

INT( ) CLR

ステートメント

概要

割り込み要求を解除します。

書式

INT(〈割り込み番号〉)_ CLR

説明

〈割り込み番号〉で指定した高機能ユニットからの割り込み要求を解除します。

〈割り込み番号〉は、以下のとおり指定します。

0～15 ：割り込みユニット

16～23：高速カウンタユニット、パルス出力ユニット

関連命令

ON INT( ) INT( ) ON,INT( ) OFF,MY INT

記述例

```
INT(1) CLR
```

用例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 ON INT(0) SUB_INT ……割り込み番号INT(0)に対応するプログラムをSUB_INTに定義します
130 INT(0) ON ……割り込みINT(0)を許可します
140 FOR A=1 TO 7 STEP 1
150 OUTW WY_1,A
160 WAIT 1
170 OUTW WY_1,0
180 WAIT 1
190 NEXT
200 INT(0) OFF ……割り込みINT(0)を禁止します
210 FEND
220'
230 FUNCTION SUB_INT ……割り込みサブルーチン SUB_INT
240 PRINT MYINT ……実行中の割り込み番号を表示します
250 OUTW WY_1,&B1111111111111111
260 INT(0) CLR ……割り込み要求を解除します
270 FEND
```

INT( ) OFF

ステートメント

概要

高機能ユニットからの割り込みを禁止します。

書式

INT(〈割り込み番号〉)_ OFF

説明

〈割り込み番号〉で指定した高機能ユニットからの割り込みを禁止します。

〈割り込み番号〉は、以下のとおり指定します。

0～15 ：割り込みユニット

16～23：高速カウンタユニット、パルス出力ユニット

注意

禁止中は割り込みが発生しても無視します。(記憶しません)。

関連命令

ON INT( ) INT( ) ON,INT( ) CLR,MY INT

記述例

```
INT(1) OFF
```

用例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 ON INT(0) SUB_INT ……割り込み番号INT(0)に対応するプログラムをSUB_INTに定義します
130 INT(0) ON ……割り込みINT(0)を許可します
140 FOR A=1 TO 7 STEP 1
150 OUTW WY_1,A
160 WAIT 1
170 OUTW WY_1,0
180 WAIT 1
190 NEXT
200 INT(0) OFF ……割り込みINT(0)を禁止します
210 FEND
220'
230 FUNCTION SUB_INT ……割り込みサブルーチン SUB_INT
240 PRINT MYINT ……実行中の割り込み番号を表示します
250 OUTW WY_1,&B1111111111111111
260 INT(0) CLR ……割り込み要求を解除します
270 FEND
```

INT( ) ON

ステートメント

概要

高機能ユニットからの割り込みを許可します。

書式

INT(〈割り込み番号〉)_ ON

説明

〈割り込み番号〉で指定した高機能ユニットからの割り込みを許可します。

〈割り込み番号〉は、以下のとおり指定します。

0　15:割り込みユニット

16　23:高速カウンタユニット、パルス出力ユニット

関連命令

ON INT(),INT()CLR,INT()ON,MY INT

記述例

```
INT(0) ON
```

用例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 ON INT(0) SUB_INT ……割り込み番号INT(0)に対応するプログラムをSUB_INTに定義します
130 INT(0) ON ……割り込みINT(0)を許可します
140 FOR A=1 TO 7 STEP 1
150 OUTW WY_1,A
160 WAIT 1
170 OUTW WY_1,0
180 WAIT 1
190 NEXT
200 INT(0) OFF ……割り込みINT(0)を禁止します
210 FEND
220'
230 FUNCTION SUB_INT ……割り込みサブルーチンSUB_INT
240 PRINT MYINT ……実行中の割り込み番号を表示します
250 OUTW WY_1,&B1111111111111111
260 INT(0) CLR ……割り込み要求を解除します
270 FEND
```

INTEGER

ステートメント

概要

変数を2バイト整数として使用するよう宣言します。

書式

INTEGER_ 〈変数名〉[,〈変数名〉・・・]

INTEGER_ 〈変数名〉(〈添字の最大値〉[,〈添字の最大値〉・・・])

説明

2バイト整数型の変数の使用を宣言します。

変数の値の範囲:

- −32,768～+32,767
- &B1111 1111 1111 1111 1000 0000 0000 0000 ～&B0000 0000 0000 0000 0111 1111 1111 1111
- &O 37777700000 ～&O 00000077777
- &H FFFF8000 ～&H 00007FFF
- &M 0 ～&M 2047F

〈変数名〉には、8文字までの英数字と「_(アンダースコア)」が使用できますが、先頭の1文字は必ず英字で書き始めなければならないという約束があります。また、Pで始まる変数名は使用できません。

注意

1.予約語は、変数名として使用することができません。

2.数値型の変数の場合末尾に記号は付けません。

3.値が代入される前の変数は、数値変数では「0」として扱われます。

4.変数宣言は、プログラム中の最初のタスクブロック(FUNCTION～FEND)のはじめにまとめて記述します(タスクブロックの途中で変数宣言をすると正しくアップロードされない場合があります)。

5.配列変数の場合、添字の最大値は0になります。

関連命令

BYTE,LONG,REAL,STRING

用例

```
100 FUNCTION SAMPLE
110 BYTE A
120 INTEGER B
130 LONG C
140 REAL D
150 STRING E$
 :
 :
300 FEND
```

INV( )

オンラインコマンド　ステートメント

概要

数値をビット反転します。

書式

INV(〈数式〉)

説明

指定した〈数式〉をビット反転し、その数値を返します。
対象となる〈数式〉の種類は、整数・実数とその変数・関数です。
〈数式〉に実数が指定された場合は、小数点以下が四捨五入され2バイト整数として扱われます。

注意

2進表記の「&B0001001000110100」(10進の「4660」)をビット反転すると、「&B1110110111001011」(10進の「−4661」になります。

用例

プログラム例

```
100 FUNCTION SAMPLE
120 INTEGER A
130 A=IND(WX_0) ……「WX_0」の16ビットの内部をAに代入
140 A=INV(A) ……Aの内容を反転
150 OUTD WY_2,A ……反転した内容を「WY_2」に出力
160 FEND
```

INW( )

オンラインコマンド　ステートメント

概要

ワード指定したI/O、メモリの内容を16ビット単位で返します。

書式

INW(〈ワード指定I/O番号〉)

説明

〈ワード指定I/O番号〉で指定した外部入力I/O、外部出力I/O、メモリI/O、リンクメモリI/O、データメモリ、リンクデータメモリの内容を16ビットの数値で返します。
〈ワード指定I/O番号〉は、以下のとおり指定します。

外部入力 I/O	WX_n	n=0～127
外部出力 I/O	WY_n	n=0～127
メモリ I/O	WR_n	n=0～97
リンクメモリ I/O	WL_n	n=0～127
データメモリ	DT_n	n=0～2047
リンクデータメモリ	LD_n	n=0～255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

注意

〈ワード指定I/O番号〉は、1ワード単位で指定します。例えば、WY_0はY_&M0～Y_&MF (Y_0～Y_15) の16個のポートを、WY_1はY_&M10～Y_&M1F (Y_16～Y_31) の16個のポートをそれぞれ示します。

関連命令

IND(),INH(),INL()

用例

ポート「WX_0」の内容をポート「WY_2」に出力するプログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 A=INW(WX_0) ……「WX_0」の16ビットの内容をAに代入
130 OUTW WY_2,A ……「WY_2」の16ビットに出力する
140 GOTO 120
150 FEND
```

KEY

オンラインコマンド　オフラインコマンド

概要

ファンクションキーにコマンドを割り付けます。

書式

KEY␣〈キー番号〉,〈文字列〉

説明

〈キー番号〉で指定したファンクションキーに、任意の〈文字列〉を指定することによりコマンドを割り付けます。
〈キー番号〉には、1～20の数値を指定します。
〈文字列〉は最大15文字まで指定できます。また、特殊文字（制御コード）の指定も可能です。
コマンド割り付けの内容は、FP-BASICを終了するときに、カレントディレクトリの「FKEY.DEF」ファイルに保存されます。
「FKEY.DEF」がカレントディレクトリに存在しないときは、「FKEY.DEF」を作成します。
したがって、カレントディレクトリから「FKEY.DEF」を削除または、移動させることによって、ファンクションキーの内容を初期化できます。

【特殊文字の指定】

¥a	ベルを鳴らす
¥t	水平タブ
¥r	改行「リターン」
¥xn（16進数）	16進キャラクタコード
¥"	ダブルクォテーション
¥	¥

（a、t、r、xは大文字でも表記できます）

関連命令

KEY LIST

用例

ファンクションキーに「TON」＋を設定する場合
（その1）

```
KEY 3,TON¥r
```

（その2）

```
KEY 3,TON¥X0D
```

KEY LIST

オンラインコマンド　オフラインコマンド

概要

ファンクションキーへのコマンド割り付けの内容を表示します。

書式

KEY␣ LIST

説明

現在のファンクションキーへのコマンド割り付けの設定内容を画面に表示します。

関連命令

KEY

記述例

```
>KEY LIST
 …1  ,"LOAD    ¥""
 …2  ,"AUTO"
 …3  ,"TSTAT¥r"
 …4  ,"LIST"
 …5  ,"XQT¥r"
 …6  ,"COMPILE¥r"
 …7  ,"DWNLD¥r"
 …8  ,"PRINT"
 …9  ,"FILES¥r"
 …10 ,"QUIT ALL¥r"
 …11 ,"SAVE ¥""
 …12 ,"TOFF¥r"
 …13 ,"FORCE"
 …14 ,"RFORCE"
 …15 ,"QUIT¥r"
 …16 ,"COMPILE  ¥""
 …17 ,"UPLD¥r"
 …18 ,"CONT¥r"
 …19 ,"STEP¥r"
 …20 ,"EXIT"
>
```

KILL

オンラインコマンド　オフラインコマンド

概要

ディスクに書き込まれているファイルを削除します。

書式

KILL␣ ["][〈ドライブ番号〉:][〈パス名〉]〈ファイル名〉[.〈拡張子〉]["]

説明

〈ドライブ番号〉、〈パス名〉、〈ファイル名〉で指定したディスク上のファイルを削除します。
削除するファイルの〈拡張子〉を省略した場合は、同じ〈ファイル名〉を持つすべてのファイルが削除の対象となります。このとき、削除する前に確認のメッセージ (Sure (Y/N)) が画面に表示されます。

注意

カレントディスクが書き込み禁止の場合、指定したファイルが書き込み禁止の場合、エラーメッセージが表示されます。

記述例

```
KILL "C:¥TEST"
```

用例

ディスクの内容が以下の場合

```
>FILES
TEST_1.PRG TEST_2.PRG TEST_3.OBJ TEST_3.SYM
>
```

・「TEST_2.PRG」ファイルだけを削除する

```
>KILL "TEST_2.PRG"
>FILES
TEST_1.PRG TEST_3.PRG TEST_3.OBJ TEST_3.SYM
>
```

・「TEST_3」ファイルを拡張子を無視して削除する

```
>KILL "TEST_3"
Sure(Y/N):
```

「N」を押した場合は削除処理をキャンセルする

```
>FILES
TEST_1.PRG TEST_3.PRG TEST_3.OBJ TEST_3.SYM
>
```

「Y」を押すと削除される

```
>FILES
TEST_1.PRG
>
```

LDCR

オンラインコマンド　ステートメント

概要

共有メモリデータをディスクからロードします。

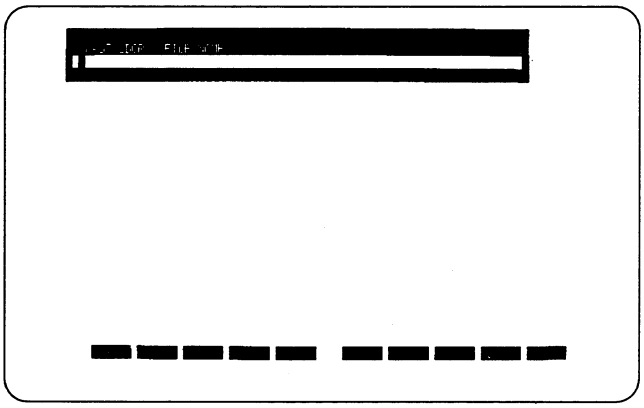
書式

LDCR_ ["〈ファイル名〉"]

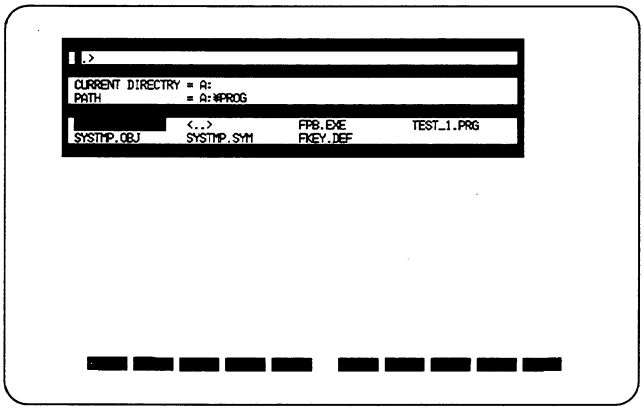
説明

〈ファイル名〉で指定したファイルから共有メモリデータをロードします。

ロードする高機能ユニットのスロット指定と共有メモリのアドレス指定は、SVCR文でファイルを作成する際にファイルに書き込まれています。



〈ファイル名〉が省略されたときは、次のようなウィンドウでの選択になります。



〈ファイル名〉の拡張子には「.CMN」を指定しますが、拡張子を省略しても自動的に「.CMN」が付けられます。

関連命令

LDIO,SVCR,SVIO

記述例

LDCR "TEST2"

用例

```
>SVCR 0,4,"TEST.CMN"  ……「TEST.CMN」というファイル
                         名で共有メモリをセーブする
Over Write(Y/N):Y  …… すでに同名ファイルが存在
>FILES                  するときに表示されます
FPB.EXE SPEED.COM RSDRV.SYS PRINT.SYS
CONFIG.SYS AUTOEXEC.BAT TEST.PRG TEST.OBJ
TEST.SYM TEST.CMN
>
```

```
>FILES
FPB.EXE SPEED.COM RSDRV.SYS PRINT.SYS
CONFIG.SYS AUTOEXEC.BAT TEST.PRG TEST.OBJ
TEST.SYM TEST.CMN
>LDIO "TEST.CMN"  ……「TEST.CMN」というファイル
>                    名で共有メモリデータを
                     ロードする
```

LDIO

オンラインコマンド　ステートメント

概要

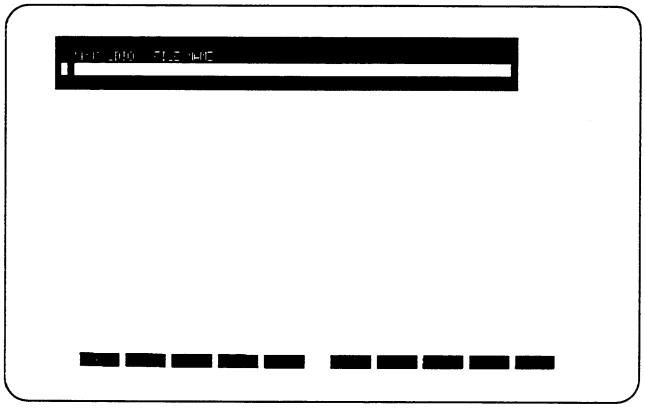
I/Oデータをディスクからロードします。

書式

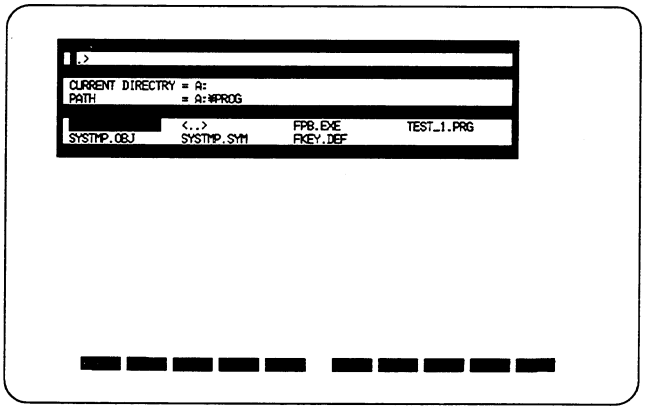
LDIO␣ ["〈ファイル名〉"]

説明

〈ファイル名〉で指定されたファイルを読みだし、その内容をI/Oに割り付けます。



〈ファイル名〉が省略されたときは、次のようなウィンドウでの選択になります。



〈ファイル名〉には必ず以下の拡張子を指定します。

出力I/O	.Y
メモリI/O	.R
データメモリ	.DT
リンクメモリ	.L

関連命令

SVIO,LDCR,SVCR

記述例

LDIO "TEST1.Y"

用例

```
>SVIO 1,2,"TEST.Y"  ……「TEST.Y」というファイル名で
                        出力I/Oデータをセーブする
Over Write(Y/N):Y   ……すでに同名ファイルが存在す
>FILES                  るときに表示されます
FPB.EXE SPEED.COM RSDRV.SYS PRINT.SYS
CONFIG.SYS AUTOEXDC.BAT TEST.PRG TEST.OBJ
TEST.SYM TEST.Y
>

>FILES
FPB.EXE SPEED.COM RSDRV.SYS PRINT.SYS
CONFIG.SYS AUTOEXDC.BAT TEST.PRG TEST.OBJ
TEST.SYM TEST.Y
>LDIO "TEST.Y"      ……出力I/Oファイル「TEST.Y」を
>                       ロードする
```

LDVAL

オンラインコマンド

概要

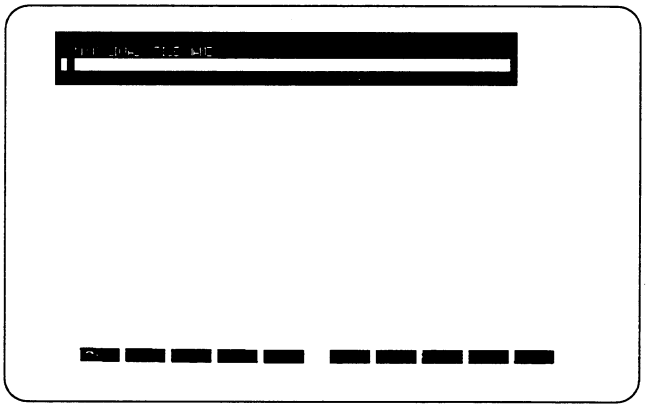
グローバル変数をディスクからロードします。

書式

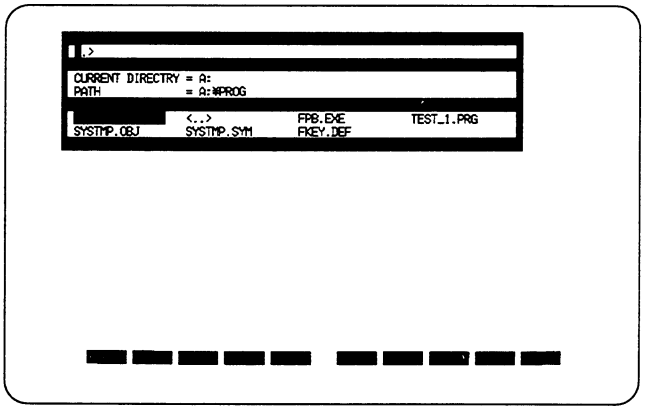
LDVAL␣["〈ファイル名〉"]

説明

すべてのグローバル変数データをファイルからロードします。〈ファイル名〉を省略した場合は、次のような画面が表示されます。



入力したファイル名にディレクトリ名またはワイルドカード(*,?)を使用した場合はさらに次のウィンドウが表示されます。また、[↵]キーのみを押した場合も、"*.*"が入力されたものとして次のウィンドウが表示されます。



[BS]、[TAB]キーによりファイルを選択し、[↵]キーで上部(選択ファイル名表示エリア)に表示されます。また、ファイル名を直接キーボードから入力することも可能です。

選択ファイル名表示エリアにファイル名を表示させ、[↵]キーを押すとそのファイル名をロードのファイル名として処理します。

[ESC]キーにより処理を中断し、ウィンドウを閉じます。

注意

CLRVAL命令を実行するとLRVAL命令が使用できなくなります。このときはもう一度プログラムをダウンロードしてください。

関連命令

SVVAL

記述例

```
LDVAL "TEST"
```

用例

```
>PRINT I  ……変数Ⅰの値を読み出す
1234
>SVVAL "TEST"  ……グローバル変数をディスクに保存する
>I =0  ……変数Ⅰの値を変更する
>PRINT I
0
>LDVAL "TEST"  ……グローバル変数をディスクから読み出す
>PRINT I  ……変数Ⅰの値が復活している
1234
>
```

LEFT$( )

オンラインコマンド　ステートメント

概要

文字列の左端から指定した文字数分の文字列を返します。

書式

LEFT$(〈文字列〉,〈文字数〉)

説明

〈文字列〉で指定した文字列の左端から〈文字数〉分の文字列を取り出し、その文字列を返します。
〈文字数〉が0の場合はエラーになります。
指定できる〈文字列〉の文字数の範囲は1～255です。

注意

〈文字列〉の文字数より大きい〈文字数〉を指定した場合、〈文字数〉は〈文字列〉の文字数になります。

関連命令

LEN(),MID$(),RIGHT$()

記述例

```
A$=LEFT$("ABCDEFG",2)
```

用例

文字列「"FP-BASIC"」の左から5文字を取り出す場合

```
>PRINT LEFT$("FP-BASIC",5)
FP-BA
>
```

〈文字データ〉の文字数より大きな〈文字数〉を指定した場合

```
>PRINT LEFT$("FP-BASIC",80)
FP-BASIC ……80文字を指定しても「FP-BASIC」の
>                8文字だけが表示される
```

プログラミング例

```
100 FUNCTION SAMPLE
120 STRING A$,B$
130 A$="FP-BASIC"  ……文字列「FP-BASIC」の
140 B$=LEFT$(A$,4) ……左から4文字分を
150 PRINT B$       ……画面に表示する
160 FEND
```

LEN( )

オンラインコマンド　ステートメント

概要

文字列の文字数を返します。

書式

LEN(〈文字列〉)

説明

〈文字列〉で指定した文字列の文字数を数値で返します。
画面に表示されない文字（コントールコード）や空白（スペース）も1文字として返します。
指定できる〈文字列〉の文字数の範囲は0～255です。

関連命令

LEFT$(),MID$(),RIGHT$()

記述例

```
A=LEN("1234")
```

用例

文字列「"FP-BASIC"」の長さを調べる場合

```
>PRINT LEN("FP-BASIC")
8
```

文字列「"ABCD"」の長さを調べる場合

```
>PRINT LEN("A B C D")
7…空白も1文字として数えられる
>
```

プログラミング例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 STRING B$
130 B$="FP-BASIC"  ……文字列「FP-BASIC」の
140 A=LEN(B$)      ……文字数を
150 PRINT A        ……画面に表示する
160 FEND
```

LINE INPUT

ステートメント

概要

ファイルから入力されるデータを区切らず一括して文字変数に代入します。

書式

LINE␣ INPUT␣ [#]〈ファイル番号〉,〈文字変数〉

説明

〈ファイル番号〉で指定されたディスクファイルから入力されるデータを、区切らず一括して文字変数に代入します。入力されたすべてのデータがコンマ「,」などの区切り記号も含めて〈文字変数〉に代入されます。

注意

使用できる文字変数は、127文字以下です。

記述例

```
LINE INPUT #1,A$
```

LINEMV

オンラインコマンド　オフラインコマンド

概要

行を移動します。

書式

LINEMV␣〈旧開始番号〉,〈旧終了行番号〉,〈移動先行番号〉[,〈増分〉]

説明

〈旧開始番号〉、〈旧終了行番号〉で指定した行を、〈移動先行番号〉に移し変えます。

〈増分〉が指定されたときは、移し替える前の行番号に関係なく〈移動先行番号〉から〈増分〉づつ増やされた行番号に移し変えられます。

記述例

```
LINEMV 1020,1060,5000,10
```

用例

```
>LIST
100 FUNCTION SAMPLE
110 ST:
120 INTEGER A
130 FOR A=1 TO 3 STEP 1
140 ON Y_&M10
150 WAIT 1
160 OFF Y_&M10
170 WAIT 1
180 NEXT A
190 END
200 FEND
>LINEMV 120,120,110 ……120行目を110行目に移動
>LIST
100 FUNCTION SAMPLE
110 INTEGER A
120 ST:
130 FOR A=1 TO 3 STEP 1
140 ON Y_&M10
150 WAIT 1
160 OFF Y_&M10
170 WAIT 1
180 NEXT A
190 END
200 FEND
```

LINK

オンラインコマンド

概要

分割コンパイルのオブジェクトをリンクし、実行オブジェクトを作成します。

書式

LINK␣〈分割オブジェクトファイル名〉[+〈分割オブジェクトファイル名〉･･･],〈実行オブジェクトファイル名〉[,H]

説明

分割ファイルのすべてのオブジェクトファイルをリンクすることにより実行オブジェクトファイルを作成します。

参考

「H」オプションをつけてLINKすると、拡張子「.HEX」のファイルを作成します。 これはROMライタ用のインテルHEX方式のプログラムファイルです。

注意

分割ソースファイルは、あらかじめCOMPILEコマンドでコンパイルしておきます。

実行するときにタスク番号、ファンクション名を指定しなかった場合は、LINK命令で最初に記述した〈分割オブジェクトファイル〉の一番最初のファンクションがタスク１で実行されます。

LINK命令は80字以内で記述してください。

LINKできるソースファイルの数は12個までです。

記述例

```
LINK MAIN+SUB1,OBJECT
```

用例

```
>NEW ↵
>100 FUNCTION MAIN ↵
>110 EXTERN FUNCTION SUB1 ↵
>120 XQT !2,SUB1 ↵
>130 DO ↵
>140 ON Y_&M10;WAIT 0.5 ↵
>150 OFF Y_&M10;WAIT 0.5 ↵
>160 LOOP ↵
>170 FEND ↵
>SAVE "MAIN" ↵ ..... ファイル名「MAIN」で保
>COMPILE "MAIN" ↵      存し、コンパイルします
COMPILE END
>NEW ↵
>100 FUNCTION SUB1 ↵
>110 DO ↵
>120 ON Y_&M11;WAIT 1 ↵
>130 OFF Y_&M11;WAIT 1 ↵
>140 LOOP ↵
>150 FEND ↵
>SAVE "SUB1" ↵ ..... ファイル名「SUB1」で保
>COMPILE "SUB1" ↵      存し、コンパイルします
COMPILE END
>LINK MAIN+SUB1,OBJECT ↵ ..... 「MAIN」と「SUB1」を結合
>DWNLD "OBJECT" ↵              して「OBJECT」を作ります
>XQT ↵ ..... プログラム「MAIN」をタ
                              スク１で実行します
```

LIST

オンラインコマンド オフラインコマンド

概要

編集中のプログラムを画面に表示します。

書式

LIST␣ [〈開始行番号〉][ー[〈終了行番号〉]]

説明

パソコンのプログラムメモリ内のプログラムを、〈開始行番号〉、〈終了行番号〉で指定した範囲で、画面に表示します。
〈開始行番号〉を省略すると、プログラムの先頭行から〈終了行番号〉で指定した行までを表示します。
〈終了行番号〉を省略すると、〈開始行番号〉で指定した行から、プログラムの終了行までを表示します。
〈開始行番号〉、〈終了行番号〉の両方を省略すると、プログラムの全体を表示します。
ESC キーを押すと、表示を一時停止します。一時停止を解除するには、もう一度ESC キーを押します。
また、SPACE キーでも一時停止を解除できます。SPACE キーでは、キーを押すたびに一行ずつ表示されます。
画面からスクロールアップして消えてしまった行の表示は、画面のいちばん上の行で、さらにカーソルキーの↑ キーを押すことで、逆にスクロールさせ再表示することができます。

注意

大きなプログラムでは、すべてを表示できない（逆スクロールしても表示されない）場合があります。

記述例

```
LIST
LIST 1020-
LIST -1060
LIST 1020-1060
```

用例

すべての行番号を表示する場合

```
>LIST
100 FUNCTION SAMPLE
110 WAIT SW(R_0)=1
120 OFF R_0=1
130 OUTW WY_2,&HFF
140 GOTO 110
150 FEND
>
```

範囲を指定した場合

• 120行以降すべてを表示

```
>LEST 120-
120 OFF R_0
130 OUTW WY_2,&HFF
140 GOTO 110
150 FEND
>
```

• 先頭から130行までを表示

```
>LIST-130
100 FUNCTION SAMPLE
110 WAIT SW(R_0)=1
120 OFF R_0
130 OUTW WY_2,&HFF
>
```

• 110行から140行までを表示

```
>LIST 110-140
110 WAIT SW(R_0)=1
120 OFF R_0
130 OUTW WY_2,&HFF
140 GOTO 110
>
```

LLIST

オンラインコマンド　オフラインコマンド

概要

編集中のプログラムをプリンタに印字します。

書式

LLIST␣[〈開始行番号〉][−[〈終了行番号〉]]

説明

パソコンのプログラムメモリ内のプログラムを、〈開始行番号〉、〈終了行番号〉で指定した範囲で、プリンタから印字します。

〈開始行番号〉を省略すると、プログラムの先頭行から〈終了行番号〉で指定した行までを印字します。

〈終了行番号〉を省略すると、〈開始行番号〉で指定した行からプログラムの最終行までを印字します。

〈開始行番号〉、〈終了行番号〉の両方を省略すると、プログラムの全体を印字します。

注意

MS-DOSを起動する際に、プリンタドライブ(PRINT.SYS)が組み込まれていないと、正常に印字されないことがあります。

記述例

```
LLIST 2000-3000
```

用例

すべての行番号を印字する場合

```
>LLIST

100 FUNCTION SAMPLE
110 WAIT SW(R_0)=1
120 OFF R_0
130 OUTW WY_2,&HFF
140 GOTO 110
150 FEND
```

範囲を指定した場合

120行以降すべてを印字

```
>LLIST 120-

120 OFF R_0
130 OUTW WY_2,&HFF
140 GOTO 110
150 FEND
```

先頭から130行までを印字

```
>LLIST -130

100 FUNCTION SAMPLE
110 WAIT SW(R_0)=1
120 OFF R_0
130 OUTW WY_2,&HFF
```

110行から140行までを印字

```
>LLIST 110-140

110 WAIT SW(R_0)=1
120 OFF R_0
130 OUTW WY_2,&HFF
140 GOTO 110
```

LOAD

オンラインコマンド　オフラインコマンド

概要

ディスクからパソコンのプログラムメモリにプログラムを読み出します。

書式

LOAD_ "[〈ドライブ番号〉:][〈パス名〉]["〈ファイル名〉[,PRG]]["]

説明

〈ファイル名〉で指定したファイルを、ディスクからパソコンのプログラムメモリ上に読み出します。
読み出されるファイルの内容は、ASCII形式（テキストファイル）のソースプログラムです。
〈ドライブ番号〉を省略するとカレントドライブが、〈パス名〉を省略するとカレントディレクトリが、それぞれ指定されます。
〈ファイル名〉は、英数字および「_（アンダースコア）」8文字以内（数字で始まるものは使えません）と、拡張子3文字で記述します。
ソースプログラムのファイル名の拡張子は、省略すると自動的に「.PRG」が指定されます。
〈ファイル名〉を省略するとウィンドウが開きます。
ウインドウ上で参照できるファイル数は50個までです。

注意

読み出す前にパソコンのプログラムメモリ上に存在したソースプログラムは消去されます。ファイルサイズが64Kバイトを越えるプログラムは読み出せません。

関連命令

FILES,SAVE

記述例

```
LOAD "C:¥
LOAD "C:¥PC¥USER¥TEST"
```

用例

ディスクの内容が以下の場合

```
>FILES
TEST.PRG TEST_2.PRG TEST_3.PRG
>
```

プログラム「TEST_2.PRG」を読み出す

```
>LOAD "TEST_2"
>
```

```
>LOAD "TEST_2
>
```

どちらの場合も同じ結果になります。

LOC( )

ステートメント

概要

ファイル中の論理的な現在位置を返します。

書式

LOC([#]〈ファイル番号〉)

説明

〈ファイル番号〉で指定されたファイルの次の読みだし位置を返します。ファイルをオープンしてから、現在までに読み出したバイト数の次のバイト数が得られます。

記述例

```
A=LOC(#1)
```

LOCATE

オンラインコマンド　ステートメント

概要

テキスト画面のカーソルの位置を制御します。

書式

LOCATE␣〈X〉,〈Y〉

説明

画面の左上を基点(0,0)として、カーソルを任意の座標位置(X,Y)に移動します。
〈X〉は水平座標を示し、常に「0」を指定します。
〈Y〉は垂直座標を示します。

記述例

```
>LOCATE 0,10
```

LOF( )

ステートメント

概要

ファイルの大きさを返します。

書式

LOF([#]〈ファイル番号〉)

説明

〈ファイル番号〉で指定されたファイルの大きさをバイト数で返します。

記述例

```
A=LOF(#1)
```

LOG( )

オンラインコマンド　ステートメント

概要

数値の自然対数を返します。

書式

LOG(〈数式〉)

説明

〈数式〉によって与えられた値の自然対数（eを底とした対数）を返します。
〈数式〉の型に関係なく単精度の値を返します。

関連命令

EXP()

用例

```
>PRINT LOG(2)
 0.6931472
>
```

LONG

ステートメント

概要

変数を4バイト整数として使用するよう宣言します。

書式

LONG_〈変数名〉[,〈変数名〉･･･]
LONG_〈変数名〉(〈添字の最大値〉[,〈添字の最大値〉･･･])

説明

4バイト整数型の変数の使用を宣言します。
変数の値の範囲：

- －2,147,483,648 ～ ＋2,147,483,647
- &B 1000 0000 0000 0000 0000 0000 0000 0000 ～ &B 0111 1111 1111 1111 1111 1111 1111 1111
- &O 20000000000 ～ &O 17777777777
- &H 8000 0000 ～ &H 7FFF FFFF
- &M 0～&M 134217727F

〈変数名〉には、8文字までの英数字と「_（アンダースコア）」が使用できますが、先頭の1文字は必ず英字で書き始めなければならないという約束があります。また、Pで始まる変数名は使用できません。

注意

1.予約語は、変数名として使用することができません。
2.数値型の変数の場合末尾に記号は付けません。
3.値が代入される前の変数は、数値変数では「0」として扱われます。
4.変数宣言は、プログラム中の最初のタスクブロック（FUNCTION～FEND）のはじめにまとめて記述します（タスクブロックの途中で変数宣言をすると正しくアップロードされない場合があります）。
5.配列変数に場合、添字の最小値は0になります。

関連命令

BYTE,INTEGER,REAL,STRING

用例

```
100 FUNCTION SAMPLE
110 BYTE A
120 INTEGER B
130 LONG C
140 REAL D
150 STRING E$
 :
 :
300 FEND
```

LROLL( )

オンラインコマンド　ステートメント

概要

数値をビット単位で左ローテート（回転）します。

書式

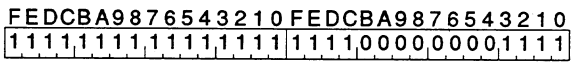
LROLL(〈数式〉,〈ビット数〉)

説明

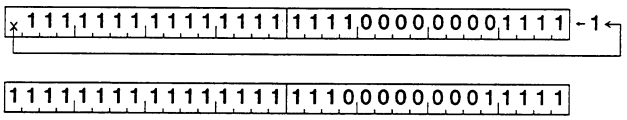
〈数式〉を指定した〈ビット数〉だけ左にローテートし、その数値を返します。
対象となる〈数式〉の種類は、整数・実数およびその型の変数・関数です。指定した〈数式〉にデータ型とローテート後のデータ型は同じになります。
〈数式〉に実数が指定された場合は、小数点以下が切り捨てられ、2バイト整数として処理されます。

32ビットの〈数式〉「&HFFFFF00F」を1ビット、左にローテートしたとき、結果は「&HFFFFE01F」になります。

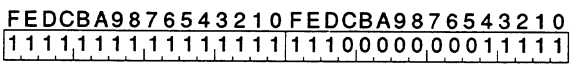
元のデータ



1ビット左へローテートします



結果



最上位ビットの情報を最下位ビットに移します。

関連命令

LSHIFT(),RROLL(),RSHIFT()

用例

メモリI/Oの内容を読み込み、3ビット左ローテートして出力する場合

```
100 FUNCTION SAMPLE
110 INTEGER A,B
120 A=INW(WR_0) ……メモリI/Oの内容を読み込み
130 B=LROLL(A,3) ……その値を3ビット左ローテートして
140 OUTW WY_2,B ……出力する
150 FEND
```

LSHIFT( )

オンラインコマンド　ステートメント

概要

数値をビット単位で左シフトします。

書式

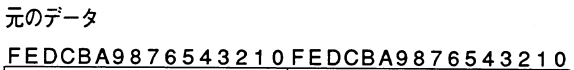
LSHIFT(〈数式〉,〈ビット数〉)

説明

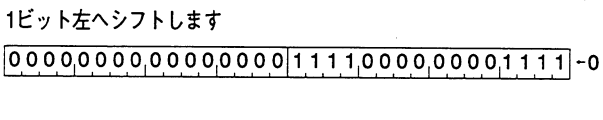
指定した〈数式〉を、指定した〈ビット数〉だけ左にシフトし、その数値を返します。
対象となる〈数式〉の種類は、整数・実数およびその型の変数・関数です。指定した〈数式〉のデータ型とシフト後のデータ型は同じになります。
〈数式〉に実数が指定された場合は、小数点以下が切り捨てられ、2バイト整数として処理されます。また、シフトして空いたビット（最下位ビット）には0が入ります。

32ビットの〈数式〉「&HF00F」を1ビット、左にシフトしたとき、結果は「&H1E01E」になります。

元のデータ



1ビット左へシフトします



結果

最上位1ビットは捨てられ、最下位1ビットには「0」が入ります。

関連命令

LROOL(),RROOL(),RSHIFT()

用例

メモリI/Oの内容を読み込み、3ビット左シフトして出力する場合

```
100 FUNCTION SAMPLE
110 INTEGER A,B
120 A=INW(WR_0) ……メモリI/Oの内容を読み込み
130 B=LSHIFT(A,3) ……その値を3ビット左シフトして
140 OUTW WY_2,B ……出力する
150 FEND
```

MAP

オンラインコマンド

概要

タスクごとにローカル変数領域を割り付けます。

書式

MAP_!〈タスク番号〉,〈サイズ〉

説明

タスクごとにローカル変数領域を割り付けます。
〈タスク番号〉と〈サイズ〉を省略すると、全タスク(1～16)の割り付け内容を画面に表示します。
〈サイズ〉は、Kバイト単位で指定します。
(0=0Kバイト、1=1Kバイト、2=2Kバイト...)

注意

1. ローカル変数領域は最大32Kバイトあり、デフォルトで各タスクに2KバイトずつでFree Memoryは0Kバイト割り付けられています。 あるタスクのローカル変数領域のサイズを3Kバイト以上割り付けたい場合は、他のタスクのサイズを小さくしてできたFree Memoryを割り付けることができます。しかし、ローカル変数領域は、タスク稼動のためのスタックメモリとしても使用されていますので、タスクごとのローカル変数領域の設定にMAP!n,0 (n=1～16)と入力しても実際の割り付けサイズは0Kバイトにはならず、タスク稼動の最低サイズ(スタックメモリ)として0.75Kバイト確保しています。MAPの表示では小数点以下切り捨てられていますので、ローカル変数領域は、実際0.75Kバイトですが、0Kバイトと表示され、Free memoryは実際1.25Kバイトですが1Kバイトと表示されています。
2. ERROR 2004が発生した場合はこの命令で発生したタスクのスタック領増やしてください。

参照

グローバル変数、ローカル変数の違いについては巻末の変数の頁をご覧ください。

記述例

```
MAP !1,1
MAP
```

用例

```
>MAP !1,1……タスク1のローカル変数領域を1Kbyteに設定します。
>MAP !2,1……タスク2のローカル変数領域を1Kbyteに設定します。
>MAP ……タスクごとのローカル変数領域を表示します。
task No.  1 : 1Kbytes
task No.  2 : 1Kbytes
task No.  3 : 2Kbytes
task No.  4 : 2Kbytes
task No.  5 : 2Kbytes
task No.  6 : 2Kbytes
task No.  7 : 2Kbytes
task No.  8 : 2Kbytes
task No.  9 : 2Kbytes
task No.  10 : 2Kbytes
task No.  11 : 2Kbytes
task No.  12 : 2Kbytes
task No.  13 : 2Kbytes
task No.  14 : 2Kbytes
task No.  15 : 2Kbytes
task No.  16 : 2Kbytes
Groval valiavle  41Kbytes
Free memory  : 2Kbytes
>
```

MEW$( )

オンラインコマンド　ステートメント

概要

数値をMEW表記の文字列に変換します。

書式

MEW$(〈数式〉)

説明

指定した〈数式〉をMEW表記（最下位桁が16進表記で最下位桁以外が10進表記）の文字列に変換して返します。
対象となる〈数式〉の種類は、整数・実数とその変数・関数です。
〈数式〉に実数が指定された場合は、小数点以下が四捨五入され、2バイト整数として扱われます。

関連命令

BIN$(),HEX$(),OCT$()

記述例

```
A$=MEW$(2047)
```

用例

2進数「&B101000000」をMEW表記の文字列に変換する場合

```
>PRINT MEW$(&B101000000)
200
>
```

8進数「&O200」をMEW表記の文字列に変換する場合

```
>PRINT MEW$(&O200)
80
>
```

10進数「20」をMEW表記の文字列に変換する場合

```
>PRINT MEW$(20)
14
>
```

16進数「&H200」をMEW表記の文字列に変換する場合

```
>PRINT MEW$(&H200)
320
>
```

プログラミング例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 STRING B$
130 A=INL(WX_0) ……「WX_0」ポートの内容を
140 OUTL WY_2,A ……「WY_2」ポートに出力し
150 B$=MEW$(A) ……MEW表記に変換して
160 PRINT B$ ……画面に表示する
170 FEND
```

MID$( )

オンラインコマンド　ステートメント

概要

文字列の指定した位置から指定した文字数分の文字列を返します。

書式

MID$(〈文字列〉,〈文字位置〉,〈文字数〉)

説明

〈文字列〉で指定した文字列の〈文字位置〉から〈文字数〉分の文字列を取り出し、その文字列を返します。
〈文字数〉が0のときはエラーになります。
指定できる〈文字列〉の文字数の範囲は1～255です。

注意

〈文字列〉の文字数より大きい〈文字数〉を指定したときは、〈文字数〉は〈文字列〉の文字数になります。〈文字列〉の文字数より大きい〈文字位置〉を指定したときは、空文字を返します。

関連命令

LEFT$(),LEN(),RIGHT$()

記述例

```
PRINT MID$("ABCDEFG",2,2)
```

用例

文字列「"FP3H-BASIC TYPE"」の6文字目から5文字分を取り出す場合

```
>PRINT MID$("FP3H-BASIC TYPE",6,5)
BASIC
>
```

〈文字データ〉の文字数より大きな〈文字数〉を指定した場合

```
>PRINT MID$("FP3H-BASIC TYPE",12,80)
TYPE ……12文字目から80文字を指定しても
>                    TYPEの4文字だけが表示される
```

プログラミング例

```
100 FUNCTION SAMPLE
110 STRING A$,B$
120 A$="FP3H-BASIC TYPE" ……文字列「FP3H-BASIC TYPE」の
130 B$=MID$(A$,6,5)          (左から)6文字目から5文字分までを
140 PRINT B$ ……画面に表示する
150 FEND
```

MID$( )

オンラインコマンド　ステートメント

概要

文字列の一部分を別の文字列で書き換えます。

書式

MID$(〈文字列〉,〈文字位置〉,〈文字数〉)=〈置換文字列〉

説明

〈文字列〉で指定した文字列の〈文字位置〉から〈文字数〉分の文字列を、〈置換文字列〉に置き換えます。
〈文字数〉が0の場合はエラーになります。
指定できる〈文字列〉の文字数の範囲は1～255です。

注意

〈文字列〉の文字数より大きい〈文字数〉を指定した場合、〈文字数〉は〈文字列〉の文字数になります。

記述例

```
MID$(A$,3,1)="4"
```

用例

```
100 FUNCTION SAMPLE
110 STRING A$
120 A$="ABCDEFGHIJ"
130 PRINT A$
140 MID$(A$,4,4)=".OK."
150 PRINT A$
160 FEND
```

MKB$( )

オンラインコマンド　ステートメント

概要

1バイト整数を1バイトの文字に変換します。

書式

MKB$(〈数式〉)

説明

1バイトの整数（数値、－128～127）を1バイトの文字に変換して返します。

関連命令

MKL$(),MKS$(),MKW$()

用例

```
>PRINT MKB$(65)
A
>PRINT MKB$(&H41)
A
>
```

MKL$( )

オンラインコマンド　ステートメント

概要

4バイト整数を4バイトの文字列に変換します。

書式

MKL$(〈数式〉)

説明

4バイト整数(−2147483648〜2147483647)を4バイトの文字列に変換して返します。

関連命令

MKB$(),MKS$(),MKW$()

記述例

```
>PRINT MKL$(&H41424344)
DCBA
>
```

MKS$( )

オンラインコマンド　ステートメント

概要

4バイト実数を4バイトの文字列に変換します。

書式

MKS$(〈数式〉)

説明

4バイトの実数を4バイトの文字列に変換して返します。

関連命令

MKB$(),MKL$(),MKW$()

記述例

```
>PRINT MKS$(123.456)
y鉋B
>
```

MKW$( )

オンラインコマンド　ステートメント

概要

2バイト整数を2バイトの文字列に変換します。

書式

MKW$(〈数式〉)

説明

2バイト整数(−32768〜32767)を2バイトの文字列に変換して返します。

関連命令

MKB$(),MKL$(),MKS$()

記述例

```
>PRINT MKW$(&H3031)
10
>
```

MODE

オンラインコマンド

概要

BASICタイプCPUの現在のモードを表示します。

書式

MODE

説明

BASICタイプCPUの現在の動作モードを表示します。
表示される動作モードは、以下のとおりです。

PROG	プログラムモード
RUN	ランモード
TEST RUN	テストランモード
REM PROG	リモート制御のプログラムモード
REM RUN	リモート制御のランモード
REM TEST RUN	リモート制御のテストランモード

注意

それぞれの動作モードの表示の最後には、そのプログラムの格納場所により「(ROM)」または「(RAM)」が表示されます。

記述例

```
MODE
```

用例

```
>MODE
TEST RUN(RAM)  ……テストランモード(RAM上のプログラムが実行対象)
>
```

MON

オンラインコマンド

概要

I/O、メモリおよび変数の値をリアルタイムに表示します。

書式

MON〈式〉

説明

〈式〉で指定したI/O、メモリおよび変数の値を、リアルタイムで画面に表示（モニタ）します。
〈式〉には一般に変数を指定し、I/Oまたはメモリの値をモニタする場合はSW ()、INL ()、INH ()、INW ()、IND ()の関数を使用します。
[ESC] キーまたは [↵] キーを押すと、表示を中止します。
表示中に、[B] [O] [H] [D] キのーを押すと、それぞれ2、8、16、10進表記を切り替えることができます。
〈I/O番号〉の指定は、以下のとおりです。

ビット単位	X_	Y_	R_	L_			
ワード単位	WX_	WY_	WR_	WL_	DT_	LD_	FL_

注意

変数をモニタする場合は、プログラム中で使用している変数に限ります。

関連命令

DUMP,DUMPC,DUMPP,MONDISP,MONENT

記述例

```
MON A ……プログラム中の変数Aをモニタします
MON SW(X_&M0)
MON INW(WX_0)
MON IND(DT_100)
```

用例

```
>MON A ……変数を指定
MONITOR START
    0
MONITOR END ……[ESC]キーまたはリターンキーを
>                  押してモニタ終了

>MON SW(Y_&M10) ……ビット単位で指定
MONITOR START
    0 ……表示はリアルタイムに変化する
MONITOR END
>MON INW(WY_1) ……ワード単位で指定
MONITOR START
 0000 H 「H」キーを押すと16進表示
MONITOR END
>MON IND(WY_1) ……ダブルワード単位で指定
MONITOR START
 00000000 H
MONITOR END
>
```

MONDISP

オンラインコマンド

概要

複数のI/Oおよびメモリをリアルタイムにモニタします。

書式

MONDISP

説明

MONENTコマンドにより、登録された複数のI/Oおよびメモリをウィンドウでリアルタイムにモニタします。
[ESC] キーにより表示を終了し、ウィンドウを閉じます。
表示中に、[B] [O] [H] [D] のキーを押すと、それぞれ2、8、16、10進表記を切り替えることができます。

注意

表示中に2、8、16、10進表記を切り替える場合、表記が切り替わるまでそれぞれ [B] [O] [H] [D] のキーを押し続けてください。
表示を終了する場合、ウインドウが閉じるまで[ESC] を押し続けてください。

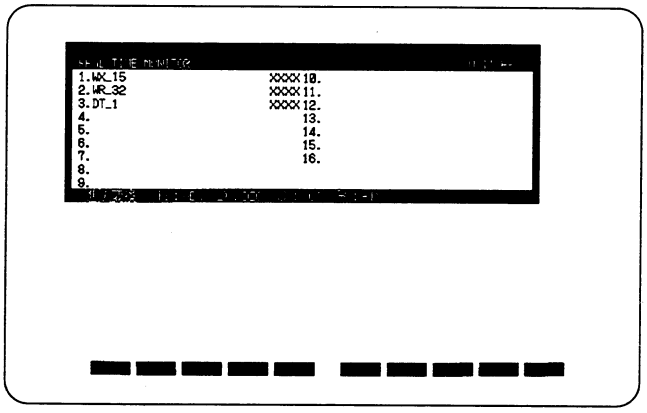
関連命令

DUMP,DUMPC,DUMPP,MON,MONENT

記述例

MONDISP

用例



MONENT

オンラインコマンド

概要

リアルタイムでモニタする複数I/Oおよびメモリを登録します。

書式

MONENT

説明

ウィンドウでリアルタイムにモニタする複数のI/Oおよびメモリを登録します。
最大登録数は16です。
↑ ↓ TAB BS キーで移動し、キー入力の後 ↵ キーで登録します。
ESC キーにより登録を終了し、ウィンドウを閉じます。

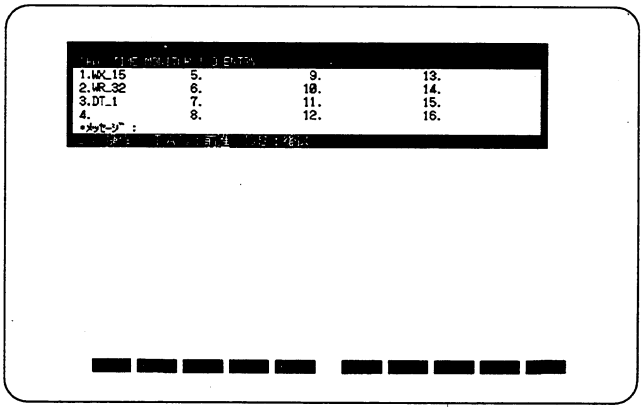
関連命令

DUMP,DUMPC,DUMPP,MON,MONDISP

記述例

MONENT

用例



MONTS

オンラインコマンド

概要

タスクごとのステータスをリアルタイムに表示します。

書式

MONTS

説明

各タスクのステータス(動作状態)を、ウィンドウでリアルタイムに表示(モニタ)します。
ESC キーにより表示を終了し、ウィンドウを閉じます。

注意

表示を終了する場合、ウインドウが閉じるまでESC キーを押し続けてください。

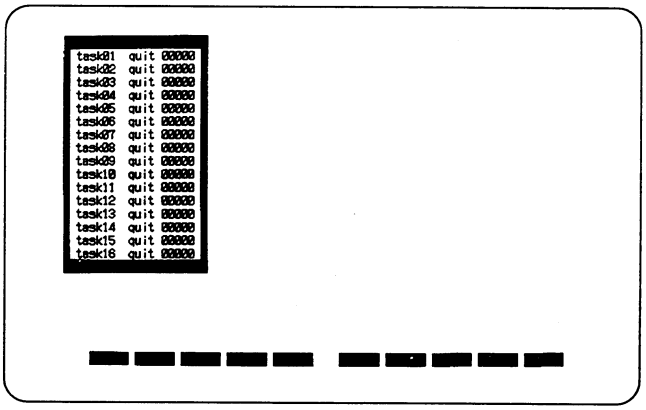
関連命令

TSTAT

記述例

MONTS

用例



MYINT

ステートメント

概要

割り込み処理ルーチンで、実行中の割り込み番号を返します。

書式

MYINT

説明

高機能ユニットからの割り込みにより割り込み処理ルーチンが実行されているとき、実行中の割り込み番号を返します。したがって、この関数は、割り込み処理ルーチンの中に記述します。

記述例

```
PRINT MYINT
```

用例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 ON INT(0) SUB_INT ……割り込み番号INT(0)に対応する
                        プログラムをSUB_INTに定義します
130 INT(0) ON ……割り込みINT(0)を許可します
140 FOR A=1 TO 7 STEP 1
150 OUTW WY_1,A
160 WAIT
170 OUTW WY_1,0
180 WAIT 1
190 NEXT
200 INT(0) OFF ……割り込みINT(0)を禁止します
210 FEND
220 '
230 FUNCTION SUB_INT ……割り込みサブルーチンSUB_INT
240 PRINT MYINT          実行中の割り込み番号を表示します
250 OUTW WY_1,&B1111111111111111
260 INT(0) CLR ……割り込み要求を解除します
270 FEND
```

MYTASK( )

ステートメント

概要

この命令を実行しているタスク番号を返します。

書式

MYTASK(〈ダミー引数〉)

説明

この命令を実行しているプログラムのタスク番号を数値で返します。

1つのプログラムを複数のタスクで使用する場合、タスクごとに入出力ポートを割り当てるのに有効です。例えば、この命令を使用したプログラムをタスク2で実行 (XQT !2) すると、MYTASK (0) は「2」を、タスク3で実行 (XQT !3) すると「3」を返しますので、プログラムを1つだけ記述し、異なるタスク番号によって実行することで、それぞれの処理内容を変更することができます。

〈ダミー引数〉には、一般に「0」を指定します。

記述例

```
PRINT MYTASK(0)
```

用例

プログラム「SUB」を、タスクで実行すると「WY_2」ポートに、16ビットデータ「&HFF」をタスク3で実行すると「WY_4」ポートに16ビットデータ「&HFF」を出力する場合のプログラム例。

```
100 FUNCTION SAMPLE
110 INTEGER A,B,C
120 XQT !2,SUB
130 XQT !3,SUB
140 FEND
150 FUNCTION SUB ……「SUB」は両方のタスクから実行される
160 A=MYTASK(0)          タスク2で実行する場合は「B=2」タス
170 B=(A-1)*             ク3で実行する場合は「B=4」が代入さ
180 C=&HFFFF             れるしたがって同じ「WY_B」というポー
190 OUTW WY_B,C          ト指定でも異なったポートにデータを
200 GOTO 160             出力できる
210 FEND
```

NAME

オンラインコマンド　オフラインコマンド

概要

ディスクに書き込まれているファイルの名前を変更します。

書式

NAME␣
["][〈ドライブ番号〉:][〈パス名〉]〈旧ファイル名〉[.〈旧拡張子〉]["]␣ AS␣ ["]〈新ファイル名〉[.〈新拡張子〉]["]

説明

〈旧ファイル名〉で指定されたディスク上のファイルの名前を、〈新ファイル名〉で指定されたファイル名に変更します。〈旧拡張子〉を省略すると、同じ〈ファイル名〉を持つすべてのファイルが書き換えの対象となります（拡張子は無視します）。また、そのときは〈新拡張子〉の指定は無効となります。〈旧拡張子〉を指定し〈新拡張子〉を省略すると、〈新拡張子〉には〈旧拡張子〉と同じ拡張子名が自動設定されます。ディスク上に同名のファイルがすでに存在するときは、エラーメッセージを画面に表示して処理を中止します。
〈ドライブ番号〉を省略するとカレントドライブが、〈パス名〉を省略するとカレントディレクトリが、それぞれ指定されます。

注意

1.「"（ダブルクォーテーション）」の代わりに、空白（スペース）でも記述できます。
2.カレントディスクが書き込み禁止の場合、および旧ファイルが書き込み禁止の場合、エラーメッセージが表示されます。

記述例

```
NAME "C:¥PC¥USER¥TEST1" AS "TEST10"
```

用例

ディスクの内容が以下の場合

```
>FILES
TEST.PRG SAMPLE.PRG SAMPLE.OBJ SAMPLE.SYM
>
```

上記の「TEST.PRG」を「TEST_2.PRG」に変更する場合

```
>NAME "TEST.PRG" AS "TEST_2.PRG"
>FILES
TEST_2.PRG SAMPLE.PRG SAMPLE.OBJ SAMPLE.SYM
>
```

上記の「SAMPLE」というファイル名をすべて（拡張子は無視して）「TEST」に変更する

```
>NAME "SAMPLE" AS "TEST"
>FILES
TEST_2.PRG TEST.PRG TEST.OBJ TEST.SYM
>
```

NEG( )

オンラインコマンド　ステートメント

概要

数値の2の補数を返します。

書式

NEG(〈数式〉)

説明

指定した〈数式〉の負の値（2の補数）を数値として返します。対象となる〈数式〉の種類は、整数・実数とその変数・関数です。

注意

2進表記の「&B0000 0000 0000 0011」（10進の「3」）の2の補数は、「&B1111 1111 1111 1111 1111 1111 1111 1100」（10進の「−3」）になります。

用例

プログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 A=IND(WX_0)
130 A=NEG(A)
140 OUTD WY_2,A
150 FEND
```

NEW

オンラインコマンド　オフラインコマンド

概要

パソコンのプログラムメモリを初期化します。

書式

NEW

説明

パソコンのプログラムメモリの内容を初期化します。プログラムの編集中に、NEWコマンドを実行すると、パソコンのプログラムメモリ上のプログラムと変数の値が初期化されます。

注意

初期化されるのはパソコンのプログラムメモリであって、BASICタイプCPUのメモリ内のプログラムおよび変数は初期化されません。

用例

```
>LIST ……現在エディット中のプログラムを表示する
100 FUNCTION SAMPLE
110 INTEGER A,B,C
 :
 :
170 END
180 FEND
>NEW……初期化する
>LIST ……プログラムを再表示する
>■ ……消えている
```

OCT$( )

オンラインコマンド　ステートメント

概要

数値を8進表記の文字列に変換します。

書式

OCT$(〈数式〉)

説明

指定した〈数式〉を8進表記の文字列に変換して返します。対象となる〈数式〉の種類は、整数・実数とその変数・関数です。

〈数式〉に実数が指定された場合は、小数点以下が四捨五入され、2バイト整数として扱われます。

関連命令

BIN$(),HEX$(),MEW$()

記述例

```
A$=OCT$(123)
```

用例

2進数「&B10000000」を8進表記の文字列に変換する場合

```
>PRINT OCT$(&B10000000)
 200
>■
```

10進数「200」を8進表記の文字列に変換する場合

```
>PRINT OCT$(200)
 310
>■
```

16進数「&H200」を8進表記の文字列に変換する場合

```
>PRINT OCT$(&H200)
 1000
>■
```

プログラミング例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 STRING B$
130 A=INL(WX_0) ……「WX_0」ポートの内容を
140 OUTL WY_2,A ……「WY_2」ポートに出力し
150 B$=OCT$(A)  ……8進表記変換して
160 PRINT B$ ……画面に表示する
170 FEND
```

OFF

オンラインコマンド　ステートメント

概要

I/Oを1ビット単位でOFFします。

書式

OFF␣〈I/O番号〉

説明

〈I/O番号〉で指定した外部出力またはメモリI/Oを1ビットだけOFFします(ビットを0にします)。
〈I/O番号〉は、以下のとおり指定します。

外部出力 I/O	Y_n	n=&M0～&M127F(0～2047)
メモリ I/O	R_n	n=&M0～&M97F (0～1567)
リンクメモリ I/O	L_n	n=&M0～&M127F(0～2047)

関連命令

ON

記述例

```
OFF Y_&M12F
```

用例

0番ポートの状態を32番ポートに出力(ON･OFF)するプログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 A=SW(X_0) ……0番ポートを調べる
130 IF A=0 THEN OFF Y_32 ……OFFなら32番ポートをOFFにする
140 IF A=1 THEN ON Y_32 ……ONなら32番ポートをONにする
150 GOTO 120
160 FEND
```

ON

オンラインコマンド　ステートメント

概要

I/Oを1ビット単位でONします。

書式

ON␣〈I/O番号〉

説明

〈I/O番号〉は指定した外部出力またはメモリI/Oを1ビットだけONします(ビットを1にします)。
〈I/O番号〉は、以下のとおり指定します。

外部出力 I/O	Y_n	n=&M0～&M127F(0～2047)
メモリ I/O	R_n	n=&M0～&M97F (0～1567)
リンクメモリ I/O	L_n	n=&M0～&M127F(0～2047)

関連命令

OFF

記述例

```
ON Y_&M100
```

用例

0番ポートの状態を32番ポートに出力(ON･OFF)するプログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 A=SW(X_0) ……0番ポートを調べる
130 IF A=1 THEN ON Y_32 ……ONなら32番ポートをONにする
140 IF A=0 THEN OFF Y_32 ……OFFなら32番ポートをOFFにする
150 GOTO 120
160 FEND
```

ONERR

ステートメント

概要

エラー割り込み処理を定義します。

書式

ONERR␣{〈ラベル名〉|〈行番号〉}

説明

エラーが起こったときに、指定したエラー処理ルーチンへジャンプすることができます。

エラー処理ルーチンの中では、必要な処理を行ったあと「ECLR」命令によってエラー状態を解除し、「RETERN」命令でエラー発生文の次の命令文に戻ります。

エラー処理ルーチンを変更するときは、再度「ONERR」命令を実行します。

〈行番号〉に「0」を指定すると「ONERR」命令を解除します。

注意

1.エラー処理ルーチン中に「GOSUB」命令は使用できません。

2.「ONERR」命令とエラー処理ルーチンは、同じプログラム(「FUNCTION～FEND」)内に記述します。

3.通常、エラー発生時には、エラーが発生したタスクが終了し、エラーの状態を表示します。エラー割り込みを使用すれば、プログラムの実行を停止せずに、エラー処理サブルーチンに実行を移すことができます。

4.1ファイル内で GOTO, GOSUB, ON GOTO, ON GOSUB, ONERR を合わせて600個まで記述できます。
また指定できるラベルの数は400個までです。

用例

「WAIT」命令によりエラー(タイムアウト)が発生したときに、画面に「"Time out error"」の文字列を表示する場合

```
100 FUNCTION SAMPLE
110 TMOUT 20 ……タイムアウトエラーを20秒に設定
120 ONERR ER_SUB ……エラー処理ルーチンの宣言
 :
190 WAIT SW(X_0)=1 ……「WAIT」中にエラーが発生すると
 :
300 ER_SUB: ……プログラム「ER_SUB」に分岐する
310 IF ERR(0)<>1004 THEN END ……タイムアウトエラー以外ならば終了する
320 PRINT "Time out error" ……タイムアウトエラーならばメッセージを表示する
330 ECLR ……エラーを解除して
340 RETURN ……元のプログラムに復帰する
400 FEND
```

ON INT( )

ステートメント

概要

高機能ユニットによる割り込み処理を定義します。

書式

ON_ INT(〈割り込み番号〉_〈タスク名〉

説明

割り込みユニット、高速カウンタユニット、パルス出力ユニットからの割り込み処理を定義します。
割り込みユニットに入力割り込みが発生したとき、他のタスクに記述した割り込み処理プログラムを実行します。
割り込み処理プログラムを実行中は、他のタスクは停止状態(WAIT状態)になります(コンソールタスクも含みます)。
〈割り込み番号〉は、以下のとおり指定します。

0～15 :割り込みユニット

16～23:高速カウンタユニット、パルス出力ユニット

〈タスク名〉には、割り込み処理ルーチンを記述したタスク名を指定します。

注意

1.END,FENDで割り込み処理を終了します。
2.割り込み処理サブルーチンの中では、I/OのWAIT命令は使用できません。
3.割り込み番号は、0～23まで使用できます。詳細は、割り込みユニットのハードウェア導入マニュアルをご覧ください。
4.割り込み処理中は、コンソール(CPUと接続しているパソコン)との通信ができなくなります。ただし、割り込み処理ルーチン内でPRINT命令は使うことができます。

関連命令

INT () ON,INT () OFF,INT () CLR,MYINT

記述例

```
ON INT(0) SUB_INT
```

用例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 ON INT(0) SUB_INT ……割り込み番号INT(0)に対応するプ
                        ログラムをSUB_INTに定義します
130 INT(0) ON ……割り込みINT(0)を許可します
140 FOR A=1 TO 7 STEP 1
150 OUTW WY_1,A
160 WAIT 1
170 OUTW WY_1,0
180 WAIT 1
190 NEXT
200 INT(0) OFF ……割り込みINT(0)を禁止します
210 FEND
220'
230 FUNCTION SUB_INT          割り込みサブルーチンSUB_INT
240 PRINT MYINT ……実行中の割り込み番号を表示します
250 OUTW WY_1,&B1111111111111111
260 INT(0) CLR ……割り込み要求を解除します
270 FEND
```

ON～GOSUB

ステートメント

概要

式の値に応じてサブルーチンを呼出します。

書式

ON␣〈数式〉␣GOSUB␣{〈ラベル名〉|〈行番号〉}[,{〈ラベル名〉|〈行番号〉}･･･]

説明

〈数式〉の値によって、指定したサブルーチンの呼び出しをします。

分岐先は行番号またはラベルで指定しますが、同一タスク内でなければなりません(タスクブロックを越える分岐はできません)。

飛び先の行番号またはラベルの記述の順序は、もっとも小さな〈数式〉の値に対応する飛び先の行番号またはラベルから記述します。

注意

1.〈数式〉の値が負になったときエラーが起こります。値が「0」または指定された分岐先の数より大きくなったときは、次のステートメントに実行が移りエラーにはなりません。

2.1ファイル内で GOTO, GOSUB, ON GOTO, ON GOSUB ONERR を合わせて600個まで記述できます。
また指定できるラベルの数は400個までです。

用例

```
100 FUNCTION SAMPLE
110 INTEGER I
120 FOR I=1 TO 3
130 ON I GOSUB ST1,ST2,ST3 ･･･････ 数式Iの値が1.2.3の時各々ラベル
140 NEXT I                          ST1,ST2,ST3のサブルーチンを
150 END                             呼び出します。
160 ST1:
170 ON Y_1
180 WAIT 1
190 RETURN
200 ST2:
210 ON Y_2
220 WAIT 1
230 RETURN
240 ST3:
250 ON Y_3
260 WAIT 1
270 RETURN
280 FEND
```

ON～GOTO

ステートメント

概要

式の値に応じて分岐します。

書式

ON␣〈数式〉␣GOTO␣{〈ラベル名〉|〈行番号〉}[,{〈ラベル名〉|〈行番号〉}･･･]

説明

〈数式〉の値によって、指定した分岐先に分岐します。

分岐先は行番号またはラベルで指定しますが、同一タスク内でなければなりません(タスクブロックを越える分岐は不可)。

飛び先の行番号またはラベルの記述の順序は、もっとも小さな〈数式〉の値に対応する飛び先の行番号またはラベルから記述します。

注意

1.〈数式〉の値が負になったときエラーが起こります。値が「0」または指定された分岐先の数より大きくなったときは、次のステートメントに実行が移りエラーにはなりません。

2.1ファイル内でGOTO,GOSUB,ON GOTO,ON GOSUB ONERRを合わせて600個まで記述できます。
また指定できるラベルの数は400個までです。

用例

```
100 FUNCTION SAMPLE
110 INTEGER I
120 FOR I=1 TO 3
130 ON I GOTO ST1,ST2,ST3 ･･･････ 数式Iの値が1.2.3.の時、各々ラベル
140 END                           ST1, ST2, ST3,へ分岐します
150 ST1: ･･･････････････････････ ラベルST1
160 ON Y_1
170 GOTO LOOP
180 ST2: ･･･････････････････････ ラベルST2
190 ON Y_2
200 GOTO LOOP
210 ST3: ･･･････････････････････ ラベルST3
220 ON Y_3
230 LOOP:
240 WAIT 1
250 NEXT I
260 FEND
```

ONSW( )

ステートメント

概要

入力割り込み処理を定義します。

書式

ONSW(〈I/O番号〉)=〈状態〉GOSUB {〈ラベル名〉 | 〈行番号〉}

説明

〈I/O番号〉で指定した入力ポートが指定した〈状態〉に変化すると、〈ラベル名〉または〈行番号〉から始まるサブルーチンを実行します。

サブルーチンの中で必要な処理をしたあと、「RETURN」命令により割り込みの発生直前に実行していた命令に戻ります。

「ONSW」命令と割り込み処理サブルーチンは同じプログラム(「FUNCTION～FEND」)内に記述します。また、割り込み処理ルーチン中に「GOSUB」命令は使えません。

割り込み処理ルーチンを変更するときは再度「ONSW」文を実行します。

〈行番号〉に「0」を指定すると「ONSW」命令を解除します。

〈I/O番号〉は、以下のとおり指定します。

外部入力 I/O	X_n	n=&M0～&M127F(0～2047)
外部出力 I/O	Y_n	n=&M0～&M127F(0～2047)
メモリ I/O	R_n	n=&M0～&M97F (0～1567)
リンクメモリ I/O	L_n	n=&M0～&M127F(0～2047)

用例

```
100 FUNCTION SAMPLE
110 INTEGER A,B
120 A=&HAA55;B=&HFFFF
130 ONSW(X_0)=1 GOSUB SUB   割り込み処理ルーチンの宣言
140 OUTW WY_2,A;WAIT 1
150 OUTW WY_2,0;WAIT 1
160 GOTO 140
170 SUB:・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・何かの処理中でも
180 PRINT SW(X_0) ・・・・・・・・・・・・・・・・・・・・割り込みが発生すれば
190 A=B・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・割り込み処理ルーチンを実行し
200 RETURN ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・戻る
210 FEND
```

注意

WAIT命令実行中は割り込みルーチンは作動しません。

OPEN

ステートメント

概要

指定されたディスクファイルを開きます。

書式

OPEN␣"〈ファイル名〉"␣[FOR〈モード〉]␣AS␣〈#〉〈ファイル番号〉

説明

〈ファイル名〉で指定されたディスクファイルを、指定された〈ファイル番号〉でオープンします。

このあと、オープンされたファイルへの入出力は、〈ファイル番号〉を指示することによりできます。

〈ファイル名〉には、ファイル名と拡張子を指定しますが、ディレクトリパス名は指定できませんので、カレントディレクトリ内でお使いください。

〈モード〉には、以下を指定します。

FOR INPUT　入力モード。
　　既存のファイルからの入力。
FOR OUTPUT　出力モード。
　　新しいファイルの作成(または上書き)。
FOR APPEND　追加モード。
　　既存のファイルの終りに追加出力。

省略時は入力モードになります。

〈ファイル番号〉には1～5を指定します。

関連命令

CLOSE

記述例

```
OPEN "TEST.ABC" FOR OUTPUT AS #1
```

用例

```
100 FUNCTION SAMPLE
110 INTEGER A,B,C
 ・
 ・
200 OPEN "INITDATA" FOR OUTPUT AS #1
210 A=1;B=2;C=3・・・・・・・・・・・・・・・・・・・・・・'変数の初期値を「INITDATA」に設定する
220 PRINT #1,A,B,C
230CLOSE #1
 ・
 ・
300 OPEN "INITDATA" FOR INPUT AS #1
310 INPUT #1,A,B,C ・・・・・・・・・・・・・・・・・・'変数の初期値を「INDATA」より読み出す
320 CLOSE #1
 ・
 ・
 ・
420 OPEN "TEST.DAT" FOR APPEND AS#1
430 PRINT #1,DATE$(),TIME$(),A,B,C・・・変数の値を「TEST.DAT」に記録する
440CLOSE#1
 ・
 ・
500FEND
```

OUTD

オンラインコマンド ステートメント

概要

ワード指定したI/O、メモリに、32ビットデータを出力します。

書式

OUTD␣〈ワード指定I/O番号〉,〈出力データ〉

説明

〈ワード指定I/O番号〉で指定した外部出力I/O、メモリI/O、リンクメモリI/O、データメモリ、リンクデータメモリに、32ビットの〈出力データ〉を出力します。

〈ワード番号〉は1ワード単位で記述できますので、〈出力データ〉の上位16ビットは、〈ワード指定I/O番号〉で指定した次の外部出力I/O、メモリI/O、リンクメモリI/O、データメモリ、リンクデータメモリに出力されます。

〈ワード指定I/O番号〉は、以下のとおり指定します。

外部出力 I/O	WY_n	n=0～126
メモリ I/O	WR_n	n=0～96
リンクメモリ I/O	WL_n	n=0～126
データメモリ	DT_n	n=0～2046
リンクデータメモリ	LD_n	n=0～254
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

注意

〈ワード指定I/O番号〉は、1ワード単位で指定します。したがって、「OUTD」命令でWY_10を指定すると、Y_&M10～Y_&M1Fの16個ビットに、次16ビット（Y_&M20～Y_&M2F）を合わせて32ビットデータとして出力します。

関連命令

OUTH,OUTL,OUTW

記述例

```
OUTD WY_1,&H12345678
```

用例

0番ポートの状態を調べ、ONの時は30～95番ポートにすべて「1」を出力し、OFFの時はすべて「0」を出力するプログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 A=SW(X_0) ……0番ポートを調べる
130 IF A=0 THEN OUTD WY_4,&HFFFFFFFF …OFFなら30～95番ポートをONにする
140 IF A=1 THEN OUTD WY_4,0 ……ONなら30～95番ポートをOFFにする
150 GOTO 120
160 FEND
```

注意

〈出力データ〉に「&HFFFFFFFF」を指定すると、2進表記で「1」が32個並びますので、32個のポートがすべてONになります。

OUTH

オンラインコマンド　ステートメント

概要

ワード指定したI/O、メモリの上位8ビットに、データを出力します。

書式

OUTH_〈ワード指定I/O番号〉,〈出力データ〉

説明

〈ワード指定I/O番号〉で指定した外部出力I/O、メモリI/O、リンクメモリI/O、データメモリ、リンクデータメモリの上位8ビットに〈出力データ〉の数値を出力します。
〈ワード指定I/O番号〉は、以下のとおり指定します。

外部出力 I/O	WY_n	n=0～127
メモリ I/O	WR_n	n=0～97
リンクメモリ I/O	WL_n	n=0～127
データメモリ	DT_n	n=0～2047
リンクデータメモリ	LD_n	n=0～255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

注意

〈ワード指定I/O番号〉は、1ワード単位で指定します。例えば、WY_1の上位8ビットは、Y_&M10～Y_&M1F (Y_16～Y_31) の16個のポートの上位8個、Y_&M18～Y_&M1F (Y_24～Y_31) を示します。

関連命令

IND(),INH(),INL(),INW(),OUTD,OUTL,OUTW

記述例

```
OUTH WY_1,&H00
```

用例

0番ポートの状態を調べ、ONの時は40～47番ポートにすべて「1」を出力し、OFFの時はすべて「0」を出力するプログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 A=SW(X_0) ……0番ポートを調べる
130 IF A=0 THEN OUTH WY_2,255 ……OFFなら40～47番ポートをONにする
140 IF A=1 THEN OUTH WY_2,0 ……ONなら40～47番ポートをOFFにする
150 GOTO 120
160 FEND
```

注意

〈出力データ〉に「255」を指定すると、2進表記で「11111111」(「1」が8個) なので8つの1ビットポートがすべてONになります。

OUTL

オンラインコマンド　ステートメント

概要

ワード指定したI/O、メモリの下位8ビットに、データを出力します。

書式

OUTL_〈ワード指定I/O番号〉,〈出力データ〉

説明

〈ワード指定I/O番号〉で指定した外部出力I/O、メモリI/O、リンクメモリI/O、データメモリ、リンクデータメモリの下位8ビットに〈出力データ〉の数値を出力します。
〈ワード指定I/O番号〉は、以下のとおり指定します。

外部出力 I/O	WY_n	n=0～127
メモリ I/O	WR_n	n=0～97
リンクメモリ I/O	WL_n	n=0～127
データメモリ	DT_n	n=0～2047
リンクデータメモリ	LD_n	n=0～255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

注意

〈ワード指定I/O番号〉は、1ワード単位で指定します。例えば、WY_1の下位8ビットは、Y_&M10～Y_&M1F (Y_16～Y_31) の16個のポートの下位8個、Y_&M10～Y_&M17 (Y_16～Y_23) を示します。

関連命令

IND(),INH(),INL()、INW(),OUTD,OUTH,OUTW

記述例

```
OUTL WY_0,&HFF
```

用例

0番ポートの状態を調べ、ONの時は32～39番ポートにすべて「1」を出力し、OFFの時はすべて「0」を出力するプログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 A=SW(X_0) ……0番ポートを調べる
130 IF A=0 THEN OUTL WY_2,255 ……OFFなら32～39番ポートをONにする
140 IF A=1 THEN OUTL WY_2,0 ……ONなら32～39番ポートをOFFにする
150 GOTO 120
160 FEND
```

注意

〈出力データ〉に「255」を指定すると、2進表記で「11111111」(「1」が8個) なので8つの1ビットポートがすべてONになります。

OUTW

オンラインコマンド　ステートメント

概要

ワード指定したI/O、メモリに、16ビットデータを出力します。

書式

OUTW_〈ワード指定I/O番号〉,〈出力データ〉

説明

〈ワード指定I/O番号〉で指定した外部出力I/O、メモリI/O、リンクメモリI/O、データメモリ、リンクデータメモリに、16ビットの〈出力データ〉の数値を出力します。
〈ワード指定I/O番号〉は、以下のとおり指定します。

外部出力 I/O	WY_n	n=0～127
メモリ I/O	WR_n	n=0～97
リンクメモリ I/O	WL_n	n=0～127
データメモリ	DT_n	n=0～2047
リンクデータメモリ	LD_n	n=0～255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

注意

〈ワード指定I/O番号〉は、1ワード単位で指定します。例えば、WY_1はY_&M10～Y_&M1F (Y_16～Y_31) の16個のポートそれぞれ示します。

関連命令

IND(),INH(),INL()、INW(),OUTD,OUTH,OUTL

記述例

```
OUTW WY_0,&H1234
```

用例

0番ポートの状態を調べ、ONの時は32～47番ポート にすべて「1」を出力し、OFFの時はすべて「0」を出力するプログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 A=SW(X_0) ……0番ポートを調べる
130 IF A=0 THEN OUTW WY_2,&HFFFF    OFFなら32～47番
                                    ポートをONにする
140 IF A=1 THEN OUTW WY_2, ……ONなら32～47番
150 GOTO 120                        ポートをOFFにする
160 FEND
```

注意

〈出力データ〉を「&HFFFF」に指定すると、2進表記で「1」が16個並びますので16個のポートがすべてONになります。

PAUSE

ステートメント

概要

テストモード時、実行中のタスクを一時停止します。

書式

PAUSE

説明

この命令を実行したタスクを一時停止します。
この命令は、PCの実行モードがテストモード (モード「0」以外) のときのみ有効です。それ以外のモードのときは無視されます。
CONT [!〈タスク番号〉] で一時停止しているタスクの実行を再開します。

注意

「PAUSE」命令はステートメントとして使います。直接実行はできません。

関連命令

CONT,STEP,TEST

用例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 LOOP:
130 PAUSE ……動作モードが「TEST」の場合、このルー
140 A=SW(X_0)              プのなかで実行が一時停止する
150 IF A=1 THEN OUTW WY_2,&H0FF0 ELSE OUTW WY_2,&HF00F
160 GOTO LOOP
170 FEND
```

上記のプログラム実行例

(写真未)

PODATA( )

オンラインコマンド　ステートメント

概要

パルス出力ユニットの経過値を読み出して返します。

書式

PODATA␣〈スロット番号〉,〈チャンネル番号〉

説明

パルス出力ユニットの経過値を読み出し返します。この値は、パルス出力ユニットの共有メモリのワードアドレス0、1または8、9の値です。
経過値は4バイト整数で返されます。
〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、0を指定します。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。その他詳細はパルス出力ユニットのマニュアルをご覧ください。

関連命令

POFCHG,POISET,POORG,POPSET,POREOS,PORESET,
POSTART,POSTART()

記述例

```
A=PODATA(2,0)
```

POFCHG

オンラインコマンド　ステートメント

概要

パルス出力ユニットのパルス周波数を切り替えます。

書式

POFCHG␣〈スロット番号〉,〈周波数切り替えリレー〉

説明

パルス出力ユニットの出力ポートY_&M13をON/OFFして、設定パルスの周波数を切り替えます。
〈スロット番号〉には、0～31を指定します。
〈周波数切り替えリレー〉は、以下のとおり指定します。
　0:低周波数
　1:高周波数

参照

出力ポートY_&M13の割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。その他詳細はパルス出力ユニットのマニュアルをご覧ください。

関連命令

PODATA(),POISET,POORG,POPSET,POREOS,
PORESET,POSTART,POSTART()

記述例

```
POFCHG 2,1
```

POISET

オンラインコマンド　ステートメント

概要

パルス出力ユニットの初期値を書き込みます。

書式

POISET_〈スロット番号〉,〈チャンネル番号〉,〈初期値〉

説明

パルス出力ユニットの初期値を書き込みます。
この設定値は、パルス出力ユニットの共有メモリのワードアドレス0、1に書き込まれます。
〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、0を指定します。
〈初期値〉には、24ビット整数（−16,777,216～＋16,777,215）を指定します。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメータ一覧表をご覧ください。その他の詳細はパルス出力ユニットのマニュアルをご覧ください。

関連命令

PODATA(),POFCHG,POORG,POPSET,POREOS,
PORESET,POSTART,POSTART()

記述例

```
POISET 2,0,-1000
```

用例

C=0（相対値制御）モード10000パルス出力します。
残りパルス数が1000パルスになった時点で減速を開始します。

```
100 FUNCTION SAMPLE
110 POISET 0,0,10000 ……初期値に10000を設定します
120 POPSET 0,0,1000
130 POFCHG 0,1
140 POSTART 0
150 WAIT SW(X_&M1)=0
160 POFCHG 0,0
170 FEND
```

C=0（相対値制御）モード1000パルス出力しながら現在値を画面に表示します。

```
100 FUNTION SAMPLE
110 INTEGER A
120 POISET 0,0,1000 ……初期値に1000を設定します
130 POPSET 0,0,500
140 POSTART 0
150 A=PODATA(0,0);PRINT A
160 IF A=0 THEN GOTO 150
170 FEND
```

POORG

オンラインコマンド　ステートメント

概要

パルス出力ユニットの原点復帰の準備をします。

書式

POORG_〈スロット番号〉

説明

パルス出力ユニットの出力ポートY_&M14をONして、原点復帰モードに切り替えます。
POSTART命令とセットで使ってください。
POORG命令につづいてPOSTART命令が実行されると原点復帰をスタートします。
〈スロット番号〉には、0～31を指定します。

参照

出力ポートY_&M14の割り付け内容については、巻末の高機能ユニットパラメータ一覧表をご覧ください。その他の詳細はパルス出力ユニットのマニュアルをご覧ください。

関連命令

PODATA(),POFCHG,POISET,POPSET,POREOS,
PORESET,POSTART,POSTART()

記述例

```
POORG 2
```

用例

原点復帰を行います。

```
100 FUNCTION SAMPLE
110 POORG 0 ……原点復帰の準備をします
120 POSTART 0
130 FEND
```

POPSET

オンラインコマンド　ステートメント

概要

パルス出力ユニットの目標値を書き込みます。

書式

POPSET␣〈スロット番号〉,〈チャネル番号〉,〈目標値〉

説明

パルス出力ユニットの目標値を書き込みます。この設定値は、ユニットの共有メモリのワードアドレス2、3に書き込まれます。
〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、0を指定します。
〈目標値〉には、24ビット整数（−16,777,216～＋16,777,215）を指定します。

参照

共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。その他の詳細はパルス出力ユニットのマニュアルをご覧ください。

関連命令

PODATA(),POFCHG,POISET,POORG,POREOS,
PORESET,POSTART,POSTART()

記述例

```
POPSET 2,0,9000
```

用例

C=0（相対値制御）モード5000パルス出力します。ラスト1000パルス目から減速を行います。

```
100 FUNCTION SAMPLE
110 POISET 0,0,5000
120 POPSET 0,0,1000 ……目標値を1000に設定します
130 POSTART 0
140 FEND
```

注意

C>P端子をP.CNT端子に接続してください。

POREOS

オンラインコマンド　ステートメント

概要

パルス出力ユニットのパルス出力方向を切り替えます。

書式

POREOS␣〈スロット番号〉,〈出力方向〉

説明

パルス出力ユニットの出力ポートY_&M12をON/OFFして、出力されるパルスの方向を決定します。
〈スロット番号〉には、0～31を設定します。
〈出力方向〉には、0または1を設定します。
・1パルス出力設定時の場合
　0：方向切換出力OUT1がOFFとなります。
　1：方向切換出力OUT1がONとなります。
・2パルス出力設定時の場合
　0：パルス出力する端子をOUT0に設定します。
　1：パルス出力する端子をOUT1に設定します。

参照

出力ポートY_&M12の割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。その他の詳細はパルス出力ユニットのマニュアルをご覧ください。

関連命令

PODATA(),POFCHG,POISET,POORG,POPSET,
PORESET,POSTART,POSTART()

記述例

```
POREOS 2,1
```

用例

C=0（相対値制御）モード2000パルスを逆転方向で出力します。

```
100 FUNCTION SAMPLE
110 POISET 0,0,2000
120 POPSET 0,0,1000
130 POREOS 0,1 ……パルス出力方向を切り替えます
140 POSTART 0
150 FEND
```

PORESET

オンラインコマンド　ステートメント

概要

パルス出力ユニットのパルス出力をストップし、内部カウンタをリセットします。

書式

PORESET␣〈スロット番号〉

説明

パルス出力ユニットの出力ポートY_&M10をONして、パルス出力をストップし、経過値、目標値を初期化します。
このとき、X_&M0～X_&M4の各フラグでONになっているものは、すべてOFFになります。
〈スロット番号〉には、0～31を指定します。

注意

パルス出力をしているときはPORESETはできません。

参照

入力ポートX_&M0～X_&M4および出力ポートY_&M10の割り付け内容については巻末の高機能ユニットパラメータ一覧表をご覧ください。その他詳細は出力ユニットのマニュアルをご覧ください。

関連命令

PODATA(),POFCHG,POISET,POORG,POPSET,
POREOS,POSTART,POSTART()

記述例

PORESET 2

POSTART

オンラインコマンド　ステートメント

概要

パルス出力ユニットのパルス出力をスタートします。

書式

POSTART␣〈スロット番号〉

説明

パルス出力ユニットの出力ポートY_&M11をONして、出力端子からパルス出力をスタートします。
〈スロット番号〉には、0～31を指定します。

参照

出力ポートY_&M11の割り付け内容については巻末の高機能ユニットパラメータ一覧表をご覧ください。その他詳細は出力ユニットのマニュアルをご覧ください。

関連命令

PODATA(),POFCHG,POISET,POORG,POPSET,
POREOS,PORESET,POSTART()

記述例

POSTART 2

POSTAT( )

オンラインコマンド　ステートメント

概要

パルス出力ユニットのユニットの動作状態を返します。

書式

POSTAT(〈スロット番号〉)

説明

パルス出力ユニットの動作状態を確認し結果を返します。
この値は、ユニットの入力ポートX_&M0～X_&M2、X_&M4の値です。
返す値は以下のとおりです。

ビット0:C=P一致フラグのとき1
ビット1:C>P比較フラグのとき1
ビット2:オーバー、アンダーフローフラグ
　　経過値が範囲外のとき1
ビット4:原点復帰中のとき1

〈スロット番号〉には、0～31を指定します。

参照

入力ポートX_&M0～X_&M4の割り付け内容については、巻末の高機能ユニットパラメーター覧表をご覧ください。その他詳細はパルス出力ユニットのマニュアルをご覧ください。

関連命令

PODATA(),POFCHG,POISET,POORG,POPSET,
POREOS,PORESET,POSTART

記述例

```
>PRINT BIN$(POSTAT (2))
10000 ……原点復帰中であることを示して
>                          います。
```

PRINT

オンラインコマンド　ステートメント

概要

パソコンの画面にデータを表示します。

書式

PRINT_ 〈式〉[,〈式〉]

説明

パソコンの画面に、〈式〉で指定したデータを表示します。
〈式〉には、定数、変数、式および関数を指定できます。
また、〈式〉を複数指定することもできます。
〈式〉に文字列を指定する場合は、「"(ダブルクォーテーション)」で前後を囲む必要があります。

注意

文字列の場合、文字列の長さは最大58バイト以内です。

用例

数値関数「INT (12.345) 」の結果を画面に表示する場合

```
>PRINT INT(12.345)
12
```

文字関数「RIGHT$ ("FP-BASIC",5) 」の結果を画面に表示する場合

```
>PRINT RIGHT$("FP-BASIC",5)
BASIC
```

数値変数「A」の内容を画面に表示する場合(すでに数値「1234」が代入されているものとする)

```
>PRINT A
1234
```

文字変数「A$」の内容を画面に表示する場合(すでに文字列「"FP-BASIC"」が代入されているものとする)

```
>PRINT A$
FP-BASIC
```

文字列「"FP-BASIC"」を画面に表示する場合

```
>PRINT "FP-BASIC"
FP-BASIC
>
```

プログラム例

```
100 FUNCTION SAMPLE
110 REAL A,B
120 DREAD A,B
130 PRINT A,"=>",ABS(A)
140 PRINT B,"=>",INT(B)
150 DATA−25,25,41
160 FEND
```

```
100 FUNCTION SAMPLE
110 LONG A,B,C
120 A=&B1111000000001111
130 PRINT "A= ",BIN$(A)
140 OUTW WY_1,A
150 WAIT 1
160 B=LSHIFT(A,1)
170 OUTW WY_1,B
180 PRINT "LSHIFT",BIN$(B)
190 WAIT 1
200 C=RSHIFT(B,1)
210 OUTW WY_1,C
210 PRINT "RSHIFT",BIN$(C)
220 WAIT 1
230 END
240 FEND
```

```
100 FUNCTION SAMPLE
110 LONG A,B,C
120 A=&B1111000000001111
130 PRINT" A= ",BIN$(A)
140 OUTW WY_1,A
150 WAIT 1
160 B=LROLL(A,1)
170 OUTW WY_1,B
180 PRINT "LROLL",BIN$(B)
190 WAIT 1
200 C=RROLL(B,1)
210 OUTW WY_1,C
220 PRINT "RROLL",BIN$(C)
230 WAIT 1
240 END
250 FEND
```

```
100 FUNCTION SAMPLE
110 INTEGER A,B,C,D,E,F,G,H
120 A=12
130 PRINT "A ",A
140 B=BCD(A)
150 PRINT "B=BCD(A) ",B
160 C=BIN(B)
170 PRINT "C=BIN(B) ",C
180 D=DECO(A)
190 PRINT "D=DECO(A)",D
200 E=ENCO(D)
210 PRINT "E=ENCO(D)",E
220 PRINT "A ·····",BIN$(A)
230 F=INV(A)
240 PRINT "F=INV(A)",BIN$(F)
250 G=NEG(A)
260 PRINT "G=NEG(A)",BIN$(G)
270 H=SEGT(A)
280 PRINT "A ·····",BIN$(A)
290 PRINT "H=SEGT(A) ·····",BIN$(H)
300 END
310 FEND
```

```
100 FUNCTION SAMPLE
110 INTEGER A,B,C,D,E,F
120 A=&B0000000100100011
130 B=&B0100010101100111
140 C=&B0000100110101011
150 D=&B0100110111101111
160 E=UNIT(A,B,C,D)
170 PRINT BIN$(E)
180 F=DIST(E,4)
190 END
200 FEND
```

PRINT%

オンラインコマンド　ステートメント

概要

高機能ユニットの共有メモリに式で指定したデータを出力します。

書式

PRINT%_〈スロット番号〉,〈先頭アドレス〉,〈式〉・・・

説明

〈スロット番号〉で指定された高機能ユニットの共有メモリの〈先頭アドレス〉に〈式〉で指定したデータを書き込みます。
〈式〉には、定数、変数、式、および関数を指定します。
〈式〉に数値を指定したときは、バイナリ形式（内部形式）で出力されます。
定数を指定したときは、指定した定数のデータ型で出力されます（10進数（−32,768〜+32,767）を用いると2バイトとして、10進数（−32,768〜+32,767）以外およびMEW表記だと4バイトとして扱われます）。
数値型の変数を指定した場合は、指定した変数の型にあわせて出力されます。
〈式〉に文字列を指定する場合は、前後を「"（ダブルクォーテーション）」で囲まなければなりません。
〈式〉を複数指定するときは、〈式〉と〈式〉の間を「,」（コンマ）で区切ります。
〈スロット番号〉には、0〜31を指定します。
〈先頭アドレス〉には、0〜2047を指定します（バイト単位）。

参照

高機能ユニットの共有メモリの割り付け内容については巻末の高機能ユニットパラメーター覧表をご覧ください。

注意

1. PRINT%命令は共有メモリのアドレスをバイト単位で指定します。

バイトアドレス	0	1	2	3	4	5	6	7
ワードアドレス	0		1		2		3	

2. 指定されたデータが数値データ、または数値変数のとき、出力されるデータは文字列に変換されずバイナリコードのままになります。
3. 出力するデータの最後にはターミネータが付加されません。したがって、ターミネータを付加したいときは最後に追加してください。

関連命令

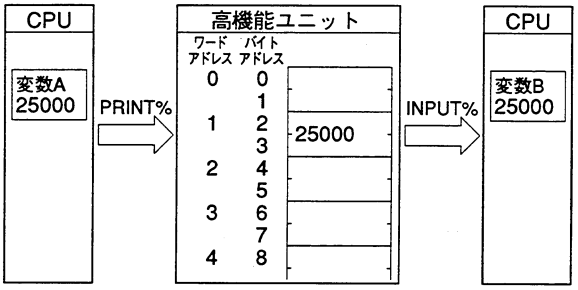
INPUT%,READ%,WRITE%

記述例

```
PRINT%4,0,"ABCDEFG"
```

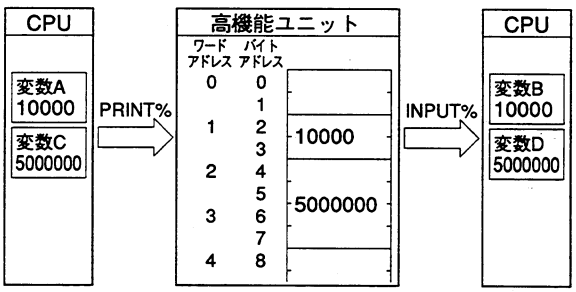
用例

変数の内容を高機能ユニットの共有メモリに対し書き込みを行い、そのデータを別の変数に読み出しを行います。




```
100 FUNCTION SAMPLE
110'
120'**PRINT%-INPUT%-1**
130'
140 INTEGER A,B
150 A=25000
160 PRINT% 2,2,A
170'
180 INPUT% 2,2,B
190 PRINT B
200'
210 FEND
```

2つの変数の内容を高機能ユニットの共有メモリに対し書き込みを行いそのデータを別の2つの変数に読み出しを行います。




```
100 FUNCTION SAMPLE
110'
120'** PRINT%−INPUT%−1 **
130 INTEGER A,C
140 LONG B,D
150 A=10000 ;B=5000000
160 PRINT% 2,2,A,B
170'
180 INPUT% 2,2,C,D
190 PRINT C
200 PRINT D
210'
220 FEND
```

PRINT

ステートメント

概要

ファイルにデータを書き出します。

書式

PRINT␣#〈ファイル番号〉,〈式〉[{ ; | , }〈式〉・・・]

説明

〈ファイル番号〉で指定したファイルに、〈式〉で指定した値を書き出します。
〈式〉には、数値型または文字型の定数または変数、式、及び関数を指定します。
また、〈式〉を複数指定することもできます。
数値データはすべて文字列に変換されます。

関連命令

INPUT

記述例

```
PRINT #1,A,A$,"TEST"
```

用例

```
100 FUNCTION SAMPLE
110 INTEGER A,B,C
 :
 :
200 OPEN "INITDATA" FOR OUTPUT AS #1
210 A=1;B=2;C=3
220 PRINT #1,A,B,C ……変数の初期値を「INITDATA」に
230 CLOSE #1                設定する
 :
 :
300 OPEN "INITDATA" FOR INPUT AS #1
310 INPUT #1,A,B,C ……変数の初期値を「INITDATA」よ
320 CLOSE #1                り読み出す
 :
 :
420 OPEN "TEST.DAT" FOR APPEND AS #1
430 PRINT #1,DATE$(),TIME$(),A,B,C ……変数の値を「TEST.DAT」に記録
440 CLOSE #1


                            する
 :
 :
500 FEND
```

PRINTR

オンラインコマンド　ステートメント

概要

リモートI/O子局の高機能ユニットの共有メモリへ式で指定したデータを出力します。

書式

PRINTR␣〈親局番号〉,〈子局番号〉,〈スロット番号〉,〈先頭アドレス〉,〈式〉･･･

説明

〈親局番号〉、〈子局番号〉、〈スロット番号〉で指定した高機能ユニットの共有メモリの〈先頭アドレス〉に、〈式〉で指定したデータを出力します。
〈式〉には、定数、変数、式、および関数を指定します。
〈式〉に数値を指定したときは、バイナリ形式（内部形式）で出力されます。
〈式〉に文字列を指定する場合は、前後を「"（ダブルクォーテーション）」で囲まなければなりません。
〈式〉を複数指定するときは、〈式〉と〈式〉の間を「,」（コンマ）で区切ります。
〈親局番号〉には、1～4を指定します。
〈子局番号〉には、1～32を指定します。
〈スロット番号〉には、0～31を指定します。
〈先頭アドレス〉には、0～2047を指定します（バイト単位）。

注意

1. PRINTR命令は共有メモリのアドレスをバイト単位で指定します。

バイトアドレス	0	1	2	3	4	5	6	7
ワードアドレス	0		1		2		3	

2. 指定されたデータが数値データ、または数値変数のとき、出力されるデータは文字列に変換されずバイナリコードのままになります。
3. 出力するデータの最後にはターミネータが付加されません。したがって、ターミネータを付加したいときは最後に追加してください。

関連命令

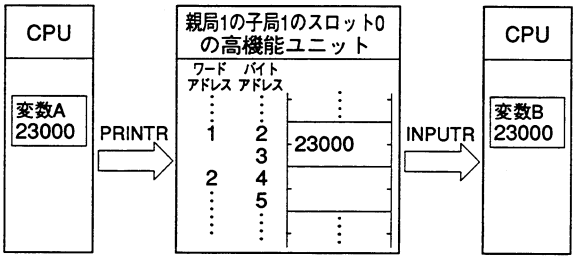
INPUTR,READR,WRITER

記述例

```
PRINTR 1,1,0,0,"ABC"
```

用例

変数の内容を親局1の子局1のスロット0の高機能ユニットの共有メモリに対し書き込み、別の変数に読込みます。




```
100 FUNCTION SAMPLE
110'
120'** PRINTR-INPUTR-1 **
130'
140 INTEGER A,B
150 A=23000
160 PRINTR 1,1,0,2,A
170'
180 INPUTR 1,1,0,2,B
190 PRINT B
200 FEND
```

PRINT USING

オンラインコマンド　ステートメント

概要

指定した書式（フォーマット）で画面にデータを表示します。

書式

PRINT_ USING_〈書式制御文字列〉{, | ; }〈式〉[, 〈式〉･･･]

説明

パソコンの画面に、〈書式制御文字列〉で指定した書式で、〈式〉で指定したデータを表示します。

〈式〉には、定数、変数、式および関数を指定できます。〈式〉に文字列を指定する場合は、「"（ダブルクォーテーション）」で前後を囲む必要があります。

〈書式制御文字列〉は、以下のとおり指定します。書式制御文字一覧

@　その位置からすべてを表示します。
&　2つの&の範囲に表示します。
!　最初に1文字のみを表示します。
#　右詰めで表示します。
,　コンマを付けて表示します。
¥　先頭に¥を付けて表示します。
＊　空白部分を＊で埋めて表示します。
.　小数点を付けて、指定の桁数で表示します。
＋　符号を指定の位置に付けて表示します。
－　符号を指定の位置に付けて表示します。

用例

@　その位置からすべてを表示します。

```
100 FUNCTION SAMPLE
110 INTEGER I1,I2,I3
120 STRING A$,B$,C$
130 PRINT "@"
140 PRINT USING "====@====","A"
150 PRINT USING "====@====","ABCDEFG"
160 PRINT USING "====@=@====","ABCDEFG","abcdefg"
```

結果

```
====A====
====ABCDEFG====
====ABCDEFG=abcdefg====
```

&　2つの&の範囲に表示します。

```
100 FUNCTION SAMPLE
110 INTEGER I1,I2,I3
120 STRING A$,B$,C$
130 PRINT "&"
140 PRINT USING "====&      &====","A"
150 PRINT USING "====&      &====","ABCDEFG"
160 PRINT USING "====&      &====","ABCDEFGHIJK"
```

結果

```
====A        ====
====ABCDEF====
====ABCDEF====
```

!　最初に1文字のみを表示します。

```
100 FUNCTION SAMPLE
110 INTEGER I1,I2,I3
120 STRING A$,B$,C$
130 PRINT "!"
140 PRINT USING "====!====","A"
150 PRINT USING "====!====","ABCDEFG"
160 PRINT USING "====!=!====","ABCDEFG","abcdefg"
```

結果

```
====A====
====A====
====A=a====
```

#　右詰めで表示します。

```
100 FUNCTION SAMPLE
110 INTEGER I1,I2,I3
120 STRING A$,B$,C$
130 PRINT "#"
140 PRINT USING "====####====",123
150 PRINT USING "====####====",123456
160 PRINT USING "====####=####====",123,456
```

結果

```
==== 123====
====%123456====
==== 123= 456====
```

, コンマを表示します。

```
100 FUNCTION SAMPLE
110 INTEGER I1,I2,I3
120 STRING A$,B$,C$
130 PRINT ","
140 PRINT USING "====#########,====",123
150 PRINT USING "====#########,====",123456
160 PRINT USING "====#########,====",12345678
```

結果

```
====       123====
====    123,456====
==== 12,345,678====
```

. 小数点を表示します。

```
100 FUNCTION SAMPLE
110 INTEGER I1,I2,I3
120 STRING A$,B$,C$
130 PRINT "."
140 PRINT USING "====####.###====",123
150 PRINT USING "====####.###====",123.456
160 PRINT USING "====####.###====",123.456789
```

結果

```
==== 123.000====
==== 123.456====
==== 123.456====
```

＋ 符号＋を指定の位置に付けて表示します。

```
100 FUNCTION SAMPLE
110 INTEGER I1,I2,I3
120 STRING A$,B$,C$
130 PRINT "+"
140 PRINT USING "====####+====",123
150 PRINT USING "====+####====",123
160 PRINT USING "====+####====",-123
```

結果

```
==== 123+====
==== +123====
==== -123====
```

－ 符号－を指定の位置に付けて表示します。

```
100 FUNCTION SAMPLE
110 INTEGER I1,I2,I3
120 STRING A$,B$,C$
130 PRINT "-"
140 PRINT USING "====####-====",-123
150 PRINT USING "====-####====",123
160 PRINT USING "====-####====",-123
```

結果

```
==== 123-====
==== +123====
==== -123====
```

PRIVATE

ステートメント

概要

ローカル変数として宣言します。

書式

PRIVATE␣{BYTE | INTEGER | LONG | REAL | STRING}␣〈変数名〉[,〈変数名〉･･･]

説明

「PRIVATE」をつけて変数宣言をすると、タスク間でローカルな変数として使用されます。

「PRIVATE」をつけずに変数宣言した場合は、タスク間でグローバルな変数として使用されます。

PRM

オンラインコマンド　ステートメント

概要

BASICタイプCPUのシステム構成に応じて、パラメータメモリを設定します。

書式

PRM␣〈機能番号〉,〈設定値〉

説明

BASICタイプCPUのシステム構成に応じて、プログラムエリアのサイズ、I/Oおよびメモリの保持・非保持、属性の設定、異常時の運転モードの設定、他のPCとのリンク状態の設定などを、パラメータメモリに書き込みます。

参照

パラメータメモリの具体的な内容については、巻末のパラメータメモリ一覧表をご覧ください。

関連命令

PRM(),PRMCLR

記述例

PRM 15,1

用例

メモリI/Oの保持エリア容量の設定（機能番号7番に設定する）を初期化状態の「60（ワード）」から「98（ワード）」に変更する場合

```
>PRINT PRM(7) ……メモリI/O保持エリアの内容を画面表示する
60 ……「60(ワード)」に設定されている
>PRM 7,98 ……「98(ワード)」に設定する
>PRINT PRM(7) ……メモリI/O保持エリアの内容を画面表示する
98 ……「98(ワード)」に変更されている
```

PRM( )

オンラインコマンド　ステートメント

概要

パラメータメモリの現在の設定値を返します。

書式

PRM(〈機能番号〉)

説明

〈機能番号〉で指定したパラメータメモリの現在の設定値を返します。

参照

パラメータメモリの内容については、巻末のパラメータメモリ一覧表をご覧ください。

関連命令

PRM,PRMCLR

記述例

PRINT PRM(15)

用例

メモリI/Oの保持エリア容量の設定（機能番号7番に設定されている）を画面に表示する場合。

```
>PRINT PRM(7) ……メモリI/O保持エリアの内容を画面表示する
60 ……「60(ワード)」に設定されている
>
```

PRMCLR

オンラインコマンド　ステートメント

概要

パラメータメモリの設定を初期化します。

書式

PRMCLR

説明

パラメータメモリの設定値をすべて初期化します。

関連命令

PRM,PRM()

記述例

PRMCLR

用例

メモリI/Oの保持エリア容量の設定（機能番号7番に設定する）を「98（ワード）」に設定した後初期化し、その内容を再度画面に表示し確認する。

```
>PRM 7,98 ……「98(ワード)」に設定する
>PRINT PRM(7) …… メモリI/O保持エリアの内容を画面表示する
98 ……「98(ワード)」に設定されている
>PRMCR ……初期化する
>PRINT PRM(7) …… メモリI/O保持エリアの内容を画面表示する
60 ……「60(ワード)」(初期値)に変わっている
>
```

PRMR

オンラインコマンド　ステートメント

概要

リモートI/O子局の占有スロット数を設定します。

書式

PRMR␣〈親局番号〉,〈子局番号〉,〈占有スロット数〉

説明

〈親局番号〉、〈子局番号〉で指定されたリモートI/O子局が占有するスロット数を登録マップに登録します。
占有スロット数は1つの子局につき32まで指定できますが、リモートI/O子局の総スロット数は128スロット以下でなければなりません。
〈親局番号〉には、1～4を指定します。
〈子局番号〉には、1～32を指定します。
〈占有スロット数〉には、1～32を指定します。

注意

登録マップについては、MEWNET-F（リモートI/Oシステム）導入マニュアルをご覧ください。

関連命令

DUMPR,PRMR(),RIOCLR,RIOSET,SLOTR,SLOTR()

記述

用例

```
>PRMR 1,1,3 ……親局番号1、子局番号1のリモートI/O子局に占有スロット数3を割りつけます
>SLOTR 1,1,0,120 ……親局番号1、子局番号1のリモートI/O子局のスロット0に16点入力ユニットを割り付けます
>RIOSET ……PRMR.SLOTRで設定された登録マップをリモートI/Oの現在値マップに転送します
>PRINT PRMR(1,1)
3
>PRINT SLOTR(1,1,0)
120
>
```

```
>PRINT SLOTR(1,1,0)
120
>SLOTR 1,1,0,,220 ……親局番号1、子局番号1のリモートI/O子局のスロット0にA/D、D/A変換ユニットを割り付けます
>RIOSET
>PRINT SLOTR(1,1,0)
220
>RIOCLR ……リモートI/Oのスロット割り付けを初期化します。
>PRINT SLOTR(1,1,0)
120
>
```

PRMR( )

オンラインコマンド　ステートメント

概要

リモートI/O子局の占有スロット数の設定値を返します。

書式

PRMR(〈親局番号〉,〈子局番号〉)

説明

〈親局番号〉、〈子局番号〉で指定されたリモートI/O子局が占有するスロット数の設定値を返します。
〈親局番号〉には、1～4を指定します。
〈子局番号〉には、1～32を指定します。

関連命令

DUMPR,PRMR,RIOCLR,RIOSET,SLOTR,SLOTR()

記述例

```
PRINT PRMR(1,1)
```

用例

```
>PRMR 1,1,3 ……親局番号1、子局番号1のリモートI/O子局に占有スロット数3を割りつけます
>SLOTR 1,1,0,120 ……親局番号1、子局番号1のリモートI/O子局のスロット0に16点入力ユニットを割り付けます
>RIOSET ……PRMR.SLOTRで設定された登録マップをリモートI/Oの現在値マップに転送します
>PRINT PRMR(1,1)
3
>PRINT SLOTR(1,1,0)
120
>
```

```
>PRINT SLOTR(1,1,0)
120
>SLOTR 1,1,0,,220 ……親局番号1、子局番号1のリモートI/O子局のスロット0にA/D、D/A変換ユニットを割り付けます
>RIOSET
>PRINT SLOTR(1,1,0)
220
>RIOCLR ……リモートI/Oのスロット割り付けを初期化します
>PRINT SLOTR(1,1,0)
120
>
```

PSADRS( )

オンラインコマンド　ステートメント

概要

位置決めユニットの現在位置を返します。

書式

PSADRS(〈スロット番号〉,〈軸No.〉)

説明

位置決めユニットの〈軸No.〉の位置決め実行中または、実行後の現在位置を読み出し、その値を返します。
返す値は実数型です。
〈スロット番号〉には、0～31を指定します。
〈軸No.〉は、以下のとおり指定します。

1:X軸
2:Y軸
3:Z軸

関連命令

PSADSET

記述例

```
PRINT PSADRS(4,1)
```

用例

位置決めFタイプ2軸ユニットをスロット4に入れ、I/O番号が&M40から始まっている場合の例です。

```
100 FUNCTION SAMPLE （教示プログラム例）
110 LONG ZJOGSP,ZMOV
120 ZJOGSP=10000
130 SLOTCLR
140 PSTYPE 4,2 ……Fタイプ2軸に設定
150 PSCTRL 4,1 ……運転モード
160 ZW:IF PSREADY(4)=0 THEN GOTO ZW …準備完了まで待つ
170 PSPCLR 4 ……パラメータを共有メモリに転送
180 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
190 PSPSET 4 ……パラメータをシステムメモリに転送
200 WAIT 0.2 ……位置決めユニットの処理待ち
210 PRINT %4,&H620,ZJOGSP ……X軸のJOG速度を設定（&H620:X軸JOG速度）
220 ON Y_&M67 ……X軸正転JOG始動(Y_&M67:X軸正転JOG)
230 WAIT 2
240 OFF Y_&M67 ……X軸正転JOG停止
250 ZW2:IF PSBUSY(4,1)=1 THEN GOTO ZW2 ……X軸JOG停止完了まで待つ
260 ZMOV=PSADRS(4,1) ……X軸の現在位置を読み出す
270 PSDSET 4,1,1,E:,A:ZMOV,10000,300,0,A:0,0 …現在位置をX軸位置データ1として設定
280 END
290 FEND
```

PSADSET

オンラインコマンド　ステートメント

概要

位置決めユニットの現在位置を変更します。

書式

PSADSET␣〈スロット番号〉,〈軸指定値〉,〈設定値〉

説明

位置決めユニットの各軸の現在位置を変更します。
〈スロット番号〉には、0～31を設定します。
〈軸指定値〉は、以下のとおり設定します。

1:X軸
2:Y軸
3:X軸とY軸
4:Z軸
5:X軸とZ軸
6:Y軸とZ軸
7:X軸とY軸とZ軸

〈設定値〉は、−8388607〈〈＋8388607の範囲で設定します。
復数軸を設定した場合は、各指定軸が〈設定値〉になります。

関連命令

PSADRS()

記述例

```
PSADSET 4,1,2000
```

用例

位置決めFタイプ2軸ユニットをスロット４に入れ、I/O番号が&M40から始まっている場合の例です。

```
100 FUNCTION SAMPLE （無限送りプログラム例）
110 SLOTCLR
120 PSTYPE 4,2 ……Fタイプ２軸に設定
130 PSCTRL 4,1 ……運転モード
140 ZW:IF PSREADY(4)=0 THEN GOTO ZW …準備完了まで待つ
150 PSPCLR 4 ……パラメータを共有メモリに転送
160 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
170 PSPSET 4 ……パラメータをシステムメモリに転送
180 WAIT 0.2 ……位置決めユニットの処理待ち
190 PSDSET 4,1,1,E:,I:10000,10000,300,0,A:0,0 ……位置データ１の設定
200 ZREP:
210 PSJOBNO 4,1,1 ……JOB1の始動するデータ番号を指定
220 PSSTART 4,1 ……JOB1を始動
230 WAIT SW(X_&M45)=1 ……位置決め完了まで待つ(X_&M45:X軸位置
240 WAIT SW(X_&M45)=0 決め完了)
250 PSADSET 4,1,0 …… X軸の現在位置を変更
260 GOTO ZREP
270 END
280 FEND
```

PSAID( )

オンラインコマンド　ステートメント

概要

位置決めユニットの補助出力の状態を読み出します。

書式

PSAID(〈スロット番号〉,〈JOBNo.〉)

説明

位置決めユニットの補助出力の状態を読み出します。
PSAID () が実行された時、補助出力ONの時は1、OFFの時は0を返します。
〈スロット番号〉には、0～31を設定します。
〈JOBNo.〉は、以下のとおり設定します。

1:JOB1
2:JOB2
3:JOB3

注意

1. JOBは軸の組み合わせにより、下表のように指定します。

	JOB1	JOB2	JOB3
独立	X	Y	Z
同時2軸	X,Y	Z	
同時3軸	X,Y,Z		

2. ユニットにない軸 (JOB) は指定できません。
3. 独立、同時2軸、同時3軸は位置決めユニットのパラメータNo.2で指定してください。(PSRRM命令の説明をご覧ください。)

関連命令

PSDSET

記述例

```
A=PSAID(4,1)
```

PSBUSY( )

オンラインコマンド　ステートメント

概要

位置決めユニットの動作状態を返します。

書式

PSBUSY(〈スロット番号〉,〈JOBNo.〉)

説明

位置決めユニットの動作状態を返します。
返す値は以下のとおりです。

0:停止中
1:動作中

〈スロット番号〉には、0～31を設定します。
〈JOBNo.〉は、以下のとおり設定します。

1:JOB1
2:JOB2
3:JOB3

注意

1. JOBは軸の組み合わせにより、下表のように指定します。

	JOB1	JOB2	JOB3
独立	X	Y	Z
同時2軸	X,Y	Z	
同時3軸	X,Y,Z		

2. ユニットにない軸(JOB)は指定できません。
3. 独立、同時2軸、同時3軸は位置決めユニットのパラメータNo.2で指定してください。(PSRRM命令の説明をご覧ください。)

記述例

```
PRINT PSBUSY(4,1)
```

用例

位置決めFタイプ2軸ユニットをスロット4に入れた場合の例です。

```
100 FUNCTION SAMPLE (多点送りプログラム例)
110 INTEGER I
120 SLOTCLR
130 PSTYPE 4,2 ……Fタイプ2軸に設定
140 PSCTRL 4,1 ……運転モード
150 ZW1:IF PSREADY(4)=0 THEN GOTO ZW1 ……準備完了まで待つ
160 PSPCLR 4 ……パラメータを共有メモリに転送
170 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
180 PSPSET 4 ……パラメータをシステムメモリに転送
190 WAIT 0.2 ……位置決めユニットの処理待ち
200 PSDSET 4,1,1,E:,I:10000,10000,300,0,A:0,0 ……位置データ1の設定
210 FOR I=1 TO 3
220 PSJOBNO 4,1,1 ……JOB1の始動するデータ番号を指定
230 PSSTART 4,1 ……JOB1を始動
240 WAIT 0.1 ……位置決めユニットの処理待ち
250 ZW2:IF PSBUSY(4,1)=1 THEN GOTO ZW2 ……JOB1の位置決め完了
260 PRINT "POINT NO ",I," END"                まで待つ
270 NEXT I
280 END
290 FEND
```

PSCTRL

オンラインコマンド　ステートメント

概要

位置決めユニットの稼働モードを設定します。

書式

PSCTRL_〈スロット番号〉,〈設定値〉

説明

位置決めユニットの出力ポートY_&M20（2軸以上）をON/OFFして、LOCALモードにするか、RUNモードにするかを設定します。

〈スロット番号〉には、0～31を設定します。

〈設定値〉は、以下のとおり設定します。

0:SLEEP（待機）モード

1:ACTIVE（運転）モード

注意

ティーチングユニットでLOCALモードに切り替えたときは、この命令で運転モードに設定しても、RUNモードには切り替わりません。

関連命令

PSREADY(),PSSTAT()

記述例

```
PSCTRL 4,1
```

用例

位置決めFタイプ2軸ユニットをスロット4に入れた場合の例です。

```
100 FUNCTION SAMPLE  (パラメータ設定プログラム例)
110 SLOTCLR
120 PSTYPE 4,2 ……Fタイプ2軸に設定
130 PSCTRL 4,1 ……運転モードに設定
140 ZW : IF PSREADY(4)=0 THEN GOTO ZW ……準備完了まで待つ
150 PSPCLR 4 ……パラメータを共有メモリに転送
160 PSPRM 4,1,2,0 ……軸モードを独立に設定単位をパ
170 PSPRM 4,1,3,0                    ルスに設定
180 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
190 PSPSET 4 ……パラメータをシステムメモリに
200 END                               転送
210 FEND
```

PSDSET

オンラインコマンド　ステートメント

概要

位置決めユニットの位置データを設定します。

書式

PSDSET␣〈スロット番号〉,〈軸指定値〉,〈データNo.〉,〈パターン〉,〈移動量〉,〈軸速度〉,〈加速度時間〉,〈ドウェルタイム〉,〈補助出力〉,〈補間速度〉

説明

位置決めユニットの位置データ設定をします。

〈スロット番号〉には、0～31を設定します。

〈軸指定値〉は、以下のとおり設定します。

1:X軸
2:Y軸
3:X軸とY軸
4:Z軸
5:X軸とZ軸
6:Y軸とZ軸
7:X軸とY軸とZ軸

〈データNo.〉には、Fタイプの場合は0～400の数値を指定します。Eタイプの場合は0～50の数値を指定します。

〈パターン〉は、以下のとおり指定します。

C:〈次のデータNo.〉続行点
P:〈次のデータNo.〉通過点
S:〈次のデータNo.〉円弧補助点
E:終了点
E:の後には数値を設定しません。
C:,P:の後には数値を設定します。
S:の後には〈データNo.〉+1の値を設定します。

〈移動量〉には、A:〈絶対値〉またはI:〈相対値〉を指定します。

〈軸速度〉には、0～速度制限の数値を指定します。

〈加速度時間〉には、ms単位で64～4,999の数値を指定します。

〈ドウェルタイム〉には、10ms単位で0～499の数値を指定します。

〈補助出力〉には、A:〈数値〉またはW:〈数値〉を指定します。数値の範囲は、0～255です。(0は補助出力未使用です。)

〈補間速度〉は、0～速度制限の数値を指定します。

	C続行点	P通過点	S円弧補助点	E終了点
次のNo.	●	●	●	–
移動量	●	●	●	●
軸速度	●	●	–	●
補間速度	●	●	–	●
加速度	●	●	–	●
ドウェルタイム	●	–	–	●
補助出力	●	●	–	●

注意

1. S:点の後の〈軸速度〉〈加減速時間〉〈ドウェルタイム〉〈補助出力〉〈補間速度〉は次の実行点と同じ値を設定してください。
2. S:点の後の次のNo.は〈データNo.〉+1の値を設定してください。

関連命令

PSAID(), PSSTART

記述例

```
PSDSET 2, 1, 1, C:2, A:2000, 1000, 64, 0, A:0, 1000
```

〈スロット番号〉
〈軸指定値〉
1はX軸を示します。
〈データNo.〉
〈パターン〉
Cは続行点
2は次のデータNo.を示します。
〈移動量〉
Aは絶対値を示します。
〈軸速度〉
〈加速度時間〉
〈ドウェルタイム〉
〈補助出力〉
AはAFTERモードを示します。
〈補間速度〉

PSECLR

オンラインコマンド　ステートメント

概要

位置決めユニットのエラー状態を解除します。

書式

PSECLR_〈スロット番号〉

説明

位置決めユニットのエラー状態を解除します。
〈スロット番号〉には、0～31を指定します。
この命令は位置決めユニットの出力ポートY_&M20 (2軸以上) をOFFします。

関連命令

PSERR(),PSSTAT()

記述例

```
PSECLR 4
```

用例

位置決めFタイプ2軸ユニットをスロット４に入れた場合の例です。

```
100 FUNCTION SAMPLE （エラークリアプログラム例）
110 INTEGER I
120 SLOTCLR
130 ONERR ZERR ……エラー発生時の処理ルーチンを指定
140 PSTYPE 4,2 ……Fタイプ2軸に設定
150 PSCTRL 4,1 ……運転モード
160 ZW:IF PSREADY(4)=0 THEN GOTO ZW ……準備完了まで待つ
170 PSPCLR 4 ……パラメータを共有メモリに転送
180 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
190 PSPSET 4 ……パラメータをシステムメモリに転送
200 WAIT 0.2 ……位置決めユニットの処理待ち
210 PSDSET 4,1,1,E:,I:10000,10000,300,0,A:0,0 ……位置データ１の設定
220 FOR I=1 TO 3
230 PSJOBNO 4,1,1 ……JOB1の始動するデータ番号を指定
240 PSSTART 4,1 ……JOB1を始動
250 WAIT 0.1 ……位置決めユニットの処理待ち
260 ZW1:IF PSBUSY(4,1)=1 THEN GOTO ZW1 ……JOB1の位置決め完了
270 PRINT "POINT NO ",I," END"　　まで待つ
280 NEXT I
290 END
300 ZERR:……エラー処理ルーチン
310 PRINT "ERR NO ",ERR(0) ……BasicPCのエラー番号を表示
320 IFB PSSTAT(4)=1 THEN ……位置決めユニットでエラー発生なら
330 PRINT "PU ERRCODE ",HEX$(PSERR(4,1)) …位置決めのエラー番号を表示
340WAIT.SW(X_8MO)=1 ……エラー原因を取り除く
350 PSECLR 4 ……位置決めユニットのエラー状態を解除
360PSCTRL 4,1
370 ELSEIF PSSTAT(4)>1 THEN ……位置決めユニットがLOCALモードなら
380 PRINT "PU LOCAL MODE"
390 ENDIF
400 ECLR
410 RETURN
420 FEND
```

PSERR( )

オンラインコマンド　ステートメント

概要

位置決めユニットのエラーコードを返します。

書式

PSERR(〈スロット番号〉,〈軸No.〉)

説明

位置決めユニットの〈軸No.〉のエラーコードを読み出し、その内容を2バイト整数で返します。

〈スロット番号〉には、0～31を設定します。

〈軸No.〉は、以下のとおり設定します。

1:X軸
2:Y軸
3:Z軸

注意

位置決めユニットのエラーコードは16進数値になっていますのでエラーコードを16進数表記の文字列に変換してください。

関連命令

PSECLR, PSSTAT()

記述例

```
PRINT HEX$(PSERR(4))
```

用例

位置決めFタイプ2軸ユニットをスロット4に入れた場合の例です。

```
100 FUNCTION SAMPLE （エラークリアプログラム例）
110 INTEGER I
120 SLOTCLR
130 ONERR ZERR ……エラー発生時の処理ルーチンを指定
140 PSTYPE 4,2 ……Fタイプ2軸に設定
150 PSCTRL 4,1 ……運転モード
160 ZW:IF PSREADY(4)=0 THEN GOTO ZW …準備完了まで待つ
170 PSPCLR 4 ……パラメータを共有メモリに転送
180 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
190 PSPSET 4 ……パラメータをシステムメモリに転送
200 WAIT 0.2 ……位置決めユニットの処理待ち
210 PSDSET 4,1,1,E:,I:10000,10000,300,0,A:0,0 ……位置データ1の設定
220 FOR I=1 TO 3
230 PSJOBNO 4,1,1 ……JOB1の始動するデータ番号を指定
240 PSSTART 4,1 ……JOB1を始動
250 WAIT 0.1 ……位置決めユニットの処理待ち
260 ZW1:IF PSBUSY(4,1)=1 THEN GOTO ZW1 ……JOB1の位置決め完了
270 PRINT "POINT NO ",I," END"　　まで待つ
280 NEXT I
290 END
300 ZERR:……エラー処理ルーチン
310 PRINT "ERR NO ",ERR(0) ……BasicPCのエラー番号を表示
320 IFB PSSTAT(4)=1 THEN ……位置決めユニットでエラー発生なら
330 PRINT "PU ERRCODE ",HEX$(PSERR(4,1)) …位置決めのエラー番号
                                        を表示
340WAIT SW(X_&MO)=1 ……エラー原因を取り除く
350 PSECLR 4 ……位置決めユニットのエラー状態を解除
360PSCTRL 4,1
370 ELSEIF PSSTAT(4)>1 THEN ……位置決めユニットが
380 PRINT "PU LOCAL MODE"……LOCALモードなら
390 ENDIF
400 ECLR
410 RETURN
420 FEND
```

PSJOBNO

オンラインコマンド ステートメント

概要

位置決めユニットのジョブで始動するデータNo.を設定します。

書式

PSJOBNO_〈スロット番号〉,〈JOBNo.〉,〈データNo.〉

説明

位置決めユニットの〈JOBNo.〉で始動する〈始動データNo.〉を設定します。

〈スロット番号〉には、0～31を設定します。

〈JOBNo.〉は軸の組み合わせで、以下のとおり設定します。

1:JOB1
2:JOB2
3:JOB3

始動する〈データNo.〉には、Fタイプの場合は、0～400の数値を指定します。Eタイプの場合は、0～50の数値を指定します。

注意

1. JOBは軸の組み合わせにより、下表のように指定します。

	JOB1	JOB2	JOB3
独立	X	Y	Z
同時2軸	X,Y	Z	
同時3軸	X,Y,Z		

2. ユニットにない軸(JOB)は指定できません。
3. 独立、同時2軸、同時3軸は位置決めユニットのパラメータNo.2で指定してください。(PSRRM命令の説明をご覧ください。)

関連命令

PSSTART

記述例

```
PSJOBNO 4,1,100
```

用例

位置決めFタイプ2軸ユニットをスロット4に入れ、I/O番号が&M40から始まっている場合の例です。

```
100 FUNCTION SAMPLE (無限送りプログラム例)
110 SLOTCLR
120 PSTYPE 4,2 ……Fタイプ2軸に設定
130 PSCTRL 4,1 ……運転モード
140 ZW:IF PSREADY(4)=0 THEN GOTO ZW ……準備完了まで待つ
150 PSPCLR 4 ……パラメータを共有メモリに転送
160 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
170 PSPSET 4 ……
                                パラメータをシステムメモリに転送
180 WAIT 0.2 ……位置決めユニットの処理待ち
190 PSDSET 4,1,1,E:,I:1000,10000,300,0,A:0,0 ……位置データ1の設定
200 ZREP:
210 PSJOBNO 4,1,1 ……JOB1の始動するデータ番号を指定
220 PSSTART 4,1 ……JOB1を始動
230 WAIT SW(X_&M45)=1 ……位置決め完了まで待つ(X_&M45:X軸
240 WAIT SW(X_&M45)=0          位置決め完了)
250 PSADSET 4,1,0 ……X軸の現在位置を変更
260 GOTO ZREP
270 END
280 FEND
```

PSLOAD

オンラインコマンド　ステートメント

概要

位置決めユニットに位置データをロードします。

書式

PSLOAD_〈スロット番号〉["〈ファイル名〉"]

説明

パソコンのディスク上の位置データを位置決めユニットにロードします(パソコンのディスク上のファイル→BASICタイプCPU→位置決めユニット)。
読み出されるファイル内容は、ASCII形式(テキストファイル)のファイルです。
〈スロット番号〉には、0～31を設定します。
〈ファイル名〉は、英数字および「_(アンダースコア)」8文字以内と、拡張子(PSD)で指定します。
〈ファイル名〉を省略するとウィンドウが開きます。

関連命令

PSSAVE

記述例

```
PSLOAD 4,"TEST.PSD"
```

PSORGH

オンラインコマンド　ステートメント

概要

位置決めユニットを機械原点復帰します

書式

PSORGH_〈スロット番号〉,〈軸指定値〉

説明

〈軸指定値〉を指定して、位置決めユニットを機械原点復帰します。
〈スロット番号〉には、0～31を設定します。
〈軸指定値〉は、以下のとおり設定します。

1:X軸
2:Y軸
3:X軸とY軸
4:Z軸
5:X軸とZ軸
6:Y軸とZ軸
7:X軸とY軸とZ軸

この命令は原点復帰が完了するまで待ちます。
ユニットにない軸は指定できません。

関連命令

PSPRM

記述例

```
PSORGH 4,1
```

用例

位置決めFタイプ2軸ユニットをスロット4に入れた場合の例です。

```
100 FUNCTION SAMPLE (原点復帰プログラム例)
110 SLOTCLR
120 PSTYPE 4,2 ……Fタイプ2軸に設定
130 PSCTRL 4,1 ……運転モード
140 ZW:IF PSREADY(4)=0 THEN GOTO ZW …準備完了まで待つ
150 PSPCLR 4 ……パラメータを共有メモリに転送
160 PSPRM 4,1,14,-100000 ……復帰アドレスを設定
170 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
180 PSPSET 4 ……パラメータをシステムメモリに転送
190 WAIT 0.2 ……位置決めユニットの処理待ち
200 PSORGH 4,1 ……機械原点復帰
210 PSORGS 4,1 ……ソフト原点復帰
220 END
230 FEND
```

PSORGS

オンラインコマンド　ステートメント

概要

位置決めユニットをソフト原点復帰します。

書式

PSORGS_〈スロット番号〉,〈軸指定値〉

説明

〈軸設定値〉を指定して、位置決めユニットを原点復帰します。
〈スロット番号〉には、0～31を設定します。
〈軸指定値〉は、以下のとおり設定します。

1:X軸
2:Y軸
3:X軸とY軸
4:Z軸
5:X軸とZ軸
6:Y軸とZ軸
7:X軸とY軸とZ軸

この命令は原点復帰が完了するまで待ちます。
ユニットにない軸は指定できません。

関連命令

PSPRM

記述例

```
PSORGS 4,1
```

用例

位置決めFタイプ2軸ユニットをスロット４に入れた場合の例です。

```
100 FUNCTION SAMPLE （原点復帰プログラム例）
110 SLOTCLR
120 PSTYPE 4,2 ……Ｆタイプ２軸に設定
130 PSCTRL 4,1 ……運転モード
140 ZW:IF PSREADY(4)=0 THEN GOTO ZW …準備完了まで待つ
150 PSPCLR 4 ……パラメータを共有メモリに転送
160 PSPRM 4,1,14,-100000 ……復帰アドレスを設定
170 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
180 PSPSET 4 ……パラメータをシステムメモリに転送
190 WAIT 0.2 ……位置決めユニットの処理待ち
200 PSORGH 4,1 ……機械原点復帰
210 PSORGS 4,1 ……ソフト原点復帰
220 END
230 FEND
```

PSPCLR

オンラインコマンド　ステートメント

概要

位置決めユニットの全パラメータをシステムメモリから共有メモリに転送します。

書式

PSPCLR_〈スロット番号〉

説明

位置決めユニットのシステムメモリから全パラメータを共有メモリに転送します。
〈スロット番号〉には、0～31を設定します。
PSPRM命令を使用する前にこの命令を実行してください。

関連命令

PSPRM,PSPRM(),PSPSET

記述例

```
PSPCLR 4
```

用例

位置決めFタイプ2軸ユニットをスロット４に入れた場合の例です。

```
100 FUNCTION SAMPLE （パラメータ設定プログラム例）
110 SLOTCLR
120 PSTYPE 4,2 ……Ｆタイプ２軸に設定
130 PSCTRL 4,1 ……運転モード
140 ZW : IF PSREADY(4)=0 THEN GOTO ZW ……準備完了まで待つ
150 PSPCLR 4 ……パラメータを共有メモリに転送
160 PSPRM 4,1,2,0 ……軸モードを独立に設定
170 PSPRM 4,1,3,0 ……単位をパルスに設定
180 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
190 PSPSET 4 ……パラメータをシステムメモリに転送
200 END
210 FEND
```

PSPRM

オンラインコマンド　ステートメント

概要

位置決めユニットの個々のパラメータを共有メモリへ設定します。

書式

PSPRM_〈スロット番号〉[,〈軸指定値〉,〈パラメータNo.〉,〈設定値〉]

説明

位置決めユニットのパラメータを共有メモリに設定します。
パラメータの設定は1回の命令でひとつしかできませんので、複数のパラメータを設定するときはその数だけ「PSPRM」命令を記述してください。
〈軸指定値〉以下が、省略されたときは、共有メモリのすべてのパラメータの設定値を表示します。
各パラメータの設定内容を変更するには、共有メモリのパラメータ設定値をシステムメモリに転送する必要があります。
パラメータを設定する時は、PSPCLR、PSPSETと組み合わせて実行してください。
〈スロット番号〉には、0～31を設定します。
〈軸指定値〉は、以下のとおり設定します。

1:X軸
2:Y軸
3:X軸とY軸
4:Z軸
5:X軸とZ軸
6:Y軸とZ軸
7:X軸とY軸とZ軸

〈パラメータNo.〉と〈設定値〉は、次頁の内容で指定します。
設定するパラメータはFタイプとEタイプでは異なります。
ユニットにない軸は設定できません。

関連命令

PSORGH,PSORGS,PSPCLR, PSPRM(),PSPSET

記述例

```
PSPRM 4,1,1,0
```

用例

位置決めFタイプ2軸ユニットをスロット4に入れた場合の例です。

```
100FUNKTION SAMPLE （パラメータ設定プログラム例）
110 SLOTCLR
120 PSTYPE 4,2 ……Fタイプ2軸に設定
130 PSCTRL 4,1 ……運転モード
140 ZW : IF PSREADY(4)=0 THEN GOTO ZW …準備完了まで待つ
150 PSPCLR 4 ……パラメータを共有メモリに転送
160 PSPRM 4,1,2,0 ……軸モードを独立に設定
170 PSPRM 4,1,3,0 ……単位をパルスに設定
180 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
190 PSPSET 4 ……パラメータをシステムメモリに転送
200 END
210 FEND
```

Fタイプ　パラメーター一覧

〈パラメータNo.〉	設定するパラメータの種類	〈設定値〉	デフォルト値
1	パルスアウトモード（整数）	0：パルス＋サイン 1：CW＋CCW	1
2	軸モード（整数）	0：独立 1：同時2軸 2：同時3軸	0
3	単位設定（整数）	0：パルス 1：mm 2：inch 3：degree　＊1	0
4	換算単位（実数）	1　（パルス） 0.0001～0.01　（mm） 0.00001～0.001（inch,degree）　＊1	1
5	速度制限値（実数）	$0 \leqq \frac{\text{速度制限値}}{\text{換算単位}} \leqq 400000$	400000 PLS/sec.
6	ソフトリミット＋（実数）	$0 \leqq \frac{\text{ソフトリミット＋}}{\text{換算単位}} \leqq 8388607$　＊2	8388607 PLS
7	ソフトリミット－（実数）	$-8388607 \leqq \frac{\text{ソフトリミット－}}{\text{換算単位}} \leqq 0$　＊2	－8388607 PLS
8	バイアス速度（実数）	0≦バイアス速度≦速度制限値	0 PLS/sec.
9	補間速度指定（整数）	0：長軸方向速度 1：軌跡速度	1
10	バックラッシュ補正（実数）	$0 \leqq \frac{\text{バックラッシュ補正}}{\text{換算単位}} \leqq 255$	0 PLS
11	誤差補正（実数）	0　（パルス） 0～±1.0000　（mm） 0～±1.00000（inch,degree）	0 PLS
12	位置決め完了時間（整数）	1～2000（msec.）	300msec.
13	復帰方向（整数）	0：アドレス＋方向 1：アドレス－方向	1
14	復帰アドレス（実数）	ソフトリミット－≦復帰アドレス≦ソフトリミット＋	0 PLS
15	復帰・JOG高速（実数）	復帰・JOG低速≦復帰・JOG高速≦速度制限値	50000 PLS/sec.
16	復帰・JOG低速（実数）	0≦復帰・JOG低速＜復帰・JOG高速	100 PLS/sec.
17	加減速時間（整数）	64～4999（msec.）	1000msec.
18	起動方法（整数）	0：通常即起動 1：通常復帰後起動 2：高速起動 3：テスト	0
19	原点復帰停止方法（整数）	0：近点ドグオン 1：近点ドグオフ 2：近点ドグオンオフ	0
20	I/F論理（整数）	&B____で指定してください。　&B＿＿＿＿＿＿ リミットオーバー入力 0：入力LED OFFでリミットオーバー 1：入力LED ONでリミットオーバー 原点入力 0：入力LED OFFで原点位置 1：入力LED ONで原点位置 原点近傍入力 0：入力LED ONで近点入力あり 1：入力LED OFFで近点入力あり ドライバ異常入力 0：入力LED OFFで異常 1：入力LED ONで異常 偏差カウンタ出力 0：クリア時 ON 1：クリア時 OFF 方向出力 0：CWでON、CCWでOFF 1：CWでOFF、CCWでON （例：&B101100のように指定します。）	000000

＊1）同時2軸、同時3軸モードでご使用の場合は、単位設定および換算単位を同じ設定にしてください。
＊2）ソフトリミット＋とソフトリミット－を両方とも0に設定した場合は、ソフトリミットを無視します。（位置決めユニットVer2.0以降）

Eタイプ　パラメーター覧

〈パラメータNo.〉	設定するパラメータの種類	〈設定値〉	デフォルト値
1	パルスアウトモード（整数）	0：パルス＋サイン 1：CW＋CCW	1
2	軸モード（整数）	0：独立（固定）　＊1	0
3	単位設定（整数）	0：パルス（固定）　＊1	0
4	換算単位（実数）	1（固定）　（パルス）　＊1	1
5	速度制限値（実数）	0≦速度制限値≦200000	200000 PLS/sec.
6	ソフトリミット＋（実数）	0≦ソフトリミット＋≦8388607　＊2	8388607 PLS
7	ソフトリミット－（実数）	－8388607≦ソフトリミット－≦0　＊2	-8388607 PLS
8	バイアス速度（実数）	0≦バイアス速度≦8000 バイアス速度≦軸速度（軸速度≠0の時）	0 PLS/sec.
9	補間速度指定（整数）	1（固定）軌跡速度　＊1	1
10	バックラッシュ補正（実数）	0（固定）　＊1	0 PLS
11	誤差補正（実数）	0（固定）　（パルス）　＊1	0 PLS
12	位置決め完了時間（整数）	1～2000（msec.）	300msec.
13	復帰方向（整数）	0：アドレス＋方向 1：アドレス－方向	1
14	復帰アドレス（実数）	ソフトリミット－≦復帰アドレス≦ソフトリミット＋	0 PLS
15	復帰・JOG高速（実数）	復帰・JOG低速≦復帰・JOG高速≦速度制限値	50000 PLS/sec.
16	復帰・JOG低速（実数）	0≦復帰・JOG低速＜500	100 PLS/sec.
17	加減速時間（整数）	64～4999（msec.）	1000msec.
18	起動方法（整数）	0：通常即起動 1：通常復帰後起動	0
19	原点復帰停止方法（整数）	0：近点ドグオン 1：近点ドグオフ 2：近点ドグオンオフ	0
20	I/F論理（整数）	&B____で指定してください。&B 外部入力 0：入力LED OFFで入力あり 1：入力LED ONで入力あり リミットオーバー入力 0：入力LED OFFでリミットオーバー 1：入力LED ONでリミットオーバー 原点入力 0：入力LED OFFで原点位置 1：入力LED ONで原点位置 原点近傍入力 0：入力LED ONで近点入力あり 1：入力LED OFFで近点入力あり ドライバ異常入力 0：入力LED OFFで異常 1：入力LED ONで異常 偏差カウンタ出力 0：クリア時 ON 1：クリア時 OFF 方向出力 0：位置＋でパルス指令1に出力,位置＋で方向出力オン 1：位置＋でパルス指令2に出力,位置＋で方向出力オフ （例：&B1101100のように指定します。）	100000
21	ユニットNo.	1～32　＊1 ＊3	1

＊1）この印の項目については、設定する必要はありません。
＊2）ソフトリミット＋とソフトリミット－を両方とも0に設定した場合は、ソフトリミットを無視します。
＊3）1に設定してください。

PSPRM( )

オンラインコマンド　ステートメント

概要

位置決めユニットのパラメータの設定値を返します。

書式

PSPRM(〈スロット番号〉,〈軸指定値〉,〈パラメータNo.〉)

説明

〈パラメータNo.〉で指定したパラメータの設定値を返します。この関数は位置決めユニットの共有メモリの値を返します。
〈スロット番号〉には、0～31を設定します。
〈軸指定値〉は、以下のとおり設定します。

1:X軸
2:Y軸
3:Z軸

〈パラメータNo.〉には、1～21を指定します。
ユニットにない軸は指定できません。

関連命令

PSPCLR,PSPRM,PSPSET

記述例

```
PRINT PSPRM(4,1,3)
```

PSPSET

オンラインコマンド　ステートメント

概要

位置決めユニットの全パラメータを共有メモリからシステムメモリに転送します。

書式

PSPSET␣〈スロット番号〉

説明

全パラメータを位置決めユニットの共有メモリから、システムメモリに転送します。
〈スロット番号〉には、0～31を設定します。
PSPRM命令でパラメータを設定した後、必ずこの命令を実行してください。

関連命令

PSPCLR,PSPRM,PSPRM()

記述例

```
PSPSET 4
```

用例

位置決めFタイプ2軸ユニットをスロット4に入れた場合の例です。

```
100 FUNCTION SAMPLE  (パラメータ設定プログラム例)
110 SLOTCLR
120 PSTYPE 4,2 ……Fタイプ2軸に設定
130 PSCTRL 4,1 ……運転モード
140 ZW : IF PSREADY(4)=0 THEN GOTO ZW ……準備完了まで待つ
150 PSPCLR 4 ……パラメータを共有メモリに転送
160 PSPRM 4,1,2,0 ……軸モードを独立に設定
170 PSPRM 4,1,3,0 ……単位をパルスに設定
180 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
190 PSPSET 4 ……パラメータをシステムメモリに
                転送
200 END
210 FEND
```

PSREADY( )

オンラインコマンド　ステートメント

概要

位置決めユニットの準備完了ステータスを返します。

書式

PSREADY(〈スロット番号〉)

説明

位置決めユニットの準備が完了したかを確認し、結果を返します。この値は、位置決めユニットの入力ポートX_&M0の値です。返す値は以下のとおりです。

0:Busy (準備中)
1:Ready (準備完了)

〈スロット番号〉には、0～31を設定します。

関連命令

PSCTRL

記述例

```
PRINT PSREADY(4)
```

用例

位置決めFタイプ2軸ユニットをスロット4に入れた場合の例です。

```
100 FUNCTION SAMPLE  (パラメータ設定プログラム例)
110 SLOTCLR
120 PSTYPE 4,2 ……Fタイプ2軸に設定
130 PSCTRL 4,1 ……運転モード
140 ZW : IF PSREADY(4)=0 THEN GOTO ZW ……準備完了まで待つ
150 PSPCLR 4 ……パラメータを共有メモリに転送
160 PSPRM 4,1,2,0 ……軸モードを独立に設定
170 PSPRM 4,1,3,0 ……単位をパルスに設定
180 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
190 PSPSET 4 ……パラメータをシステムメモリに
200 END                    転送
210 FEND
```

PSSAVE

オンラインコマンド

概要

位置決めユニットの位置データをセーブします。

書式

PSSAVE␣〈スロット番号〉["〈ファイル名〉"]

説明

位置決めユニットの位置データをパソコンのディスク上にセーブします(位置決めユニット→BASICタイプCPU→パソコンのディスク上のファイル)。

書き込まれるファイル内容は、ASCII形式(テキストファイル)のファイルです。

〈スロット番号〉には、0～31を設定します。

〈ファイル名〉は、英数字および「_(アンダースコア)」8文字以内と、拡張子(.PSD)で指定します。

〈ファイル名〉を省略するとウィンドウが開きます。

関連命令

PSLOAD

記述例

```
PSSAVE 4,"TEST.PSD"
```

PSSTART

オンラインコマンド　ステートメント

概要

位置決めユニットのジョブを始動します。

書式

PSSTART␣〈スロット番号〉,〈JOB指定値〉

説明

〈JOB指定値〉を指定して、位置決めユニットの各ジョブを始動します。

〈スロット番号〉には、0～31を設定します。

〈JOB指定値〉は、以下のとおり設定します。

1:JOB1
2:JOB2
3:JOB1とJOB2
4:JOB3
5:JOB1とJOB3
6:JOB2とJOB3
7:JOB1とJOB2とJOB3

注意

1. JOBは軸の組み合わせにより、下表のように指定します。

	JOB1	JOB2	JOB3
独立	X	Y	Z
同時2軸	X,Y	Z	
同時3軸	X,Y,Z		

2. ユニットにない軸(JOB)は指定できません。

3. 独立、同時2軸、同時3軸は位置決めユニットのパラメータNo.2で指定してください。(PSRRM命令の説明をご覧ください。)

関連命令

PSJOBNO,PSSTOP

記述例

PSSTART 4,1

用例

位置決めFタイプ2軸ユニットをスロット4に入れ、I/O番号が&M40から始まっている場合の例です。

```
100 FUNCTION SAMPLE  (無限送りプログラム例)
110 SLOTCLR
120 PSTYPE 4,2 ……Fタイプ2軸に設定
130 PSCTRL 4,1 ……運転モード
140 ZW:IF PSREADY(4)=0 THEN GOTO ZW  …準備完了まで待つ
150 PSPCLR 4 ……パラメータを共有メモリに転送
160 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
170 PSPSET 4 ……パラメータをシステムメモリに転送
180 WAIT 0.2 ……位置決めユニットの処理待ち
190 PSDSET 4,1,1,E:,I:1000,10000,300,0,A:0,0  ……位置データ1の設定
200 ZREP:
210 PSJOBNO 4,1,1 ……JOB1の始動するデータ番号を指定
220 PSSTART 4,1 ……JOB1を始動
230 WAIT SW(X_&M45)=1 ……位置決め完了まで待つ(X_&M45:X軸
240 WAIT SW(X_&M45)=0          位置決め完了)
250 PSADSET 4,1,0 ……X軸の現在位置を変更
260 GOTO ZREP
270 END
280 FEND
```

PSSTAT( )

オンラインコマンド　ステートメント

概要

位置決めユニットの動作状態を返します。

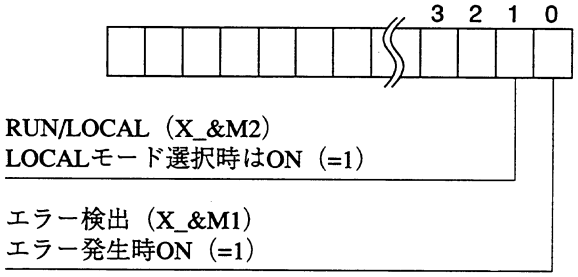
書式

PSSTAT(〈スロット番号〉)

説明

位置決めユニットの現在の動作状態を返します。
返す値は、以下のとおりです。

ビット0: エラー検出 (X_&M1)
　　エラー発生時ON (=1)
ビット1: RUN/LOCAL (X_&M2)
　　LOCALモード選択時はON (=1)



〈スロット番号〉には、0～31を設定します。

関連命令

PSCTRL,PSECLR,PSERR()

記述例

```
PRINT PSSTAT(4)
1 ……エラーが発生していることを示しています。
```

用例

位置決めFタイプ2軸ユニットをスロット4に入れた場合の例です。

```
100 FUNCTION SAMPLE （エラークリアプログラム例）
110 INTEGER I
120 SLOTCLR
130 ONERR ZERR ……エラー発生時の処理ルーチンを指定
140 PSTYPE 4,2 ……Fタイプ2軸に設定
150 PSCTRL 4,1 ……運転モード
160 ZW:IF PSREADY(4)=0 THEN GOTO ZW …準備完了まで待つ
170 PSPCLR 4 ……パラメータを共有メモリに転送
180 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
190 PSPSET 4 ……パラメータをシステムメモリに転送
200 WAIT 0.2 ……位置決めユニットの処理待ち
210 PSDSET 4,1,1,E:,I:10000,10000,300,0,A:0,0 ……位置データ1の設定
220 FOR I=1 TO 3
230 PSJOBNO 4,1,1 ……JOB1の始動するデータ番号を指定
240 PSSTART 4,1 ……JOB1を始動
250 WAIT 0.1 ……位置決めユニットの処理待ち
260 ZW1:IF PSBUSY(4,1)=1 THEN GOTO ZW1 ……JOB1の位置決め完了
270 PRINT "POINT NO ",I," END"　　まで待つ
280 NEXT I
290 END
300 ZERR: ……エラー処理ルーチン
310 PRINT "ERR NO ",ERR(0) ……BasicPCのエラー番号を表示
320 IFB PSSTAT(4)=1 THEN ……位置決めユニットでエラー発生なら
330 PRINT "PU ERRCODE ",HEX$(PSERR(4,1)) …位置決めのエラー番号
　　を表示
340 WAIT SW(X_&MO)=1 ……エラー原因を取り除く
350 PSCTRL 4,1
360 PSECLR 4 ……位置決めユニットのエラー状態を解除
370 ELSEIF PSSTAT(4)>1 THEN ……位置決めユニットがLOCALモードなら
380 PRINT "PU LOCAL MODE"
390 ENDIF
400 ECLR
410 RETURN
420 FEND
```

PSSTOP

オンラインコマンド　ステートメント

概要

位置決めユニットのジョブを停止します。

書式

PSSTOP_〈スロット番号〉,〈JOB指定値〉

説明

〈JOB指定値〉を指定して、位置決めユニットのジョブを停止します。

〈スロット番号〉には、0～31を設定します。

〈JOB指定値〉は、以下のとおり設定します。

1:JOB1
2:JOB2
3:JOB1とJOB2
4:JOB3
5:JOB1とJOB3
6:JOB2とJOB3
7:JOB1とJOB2とJOB3

注意

1. JOBは軸の組み合わせにより、下表のように指定します。

	JOB1	JOB2	JOB3
独立	X	Y	Z
同時2軸	X,Y	Z	
同時3軸	X,Y,Z		

2. ユニットにない軸 (JOB) は指定できません。
3. 独立、同時2軸、同時3軸は位置決めユニットのパラメータNo.2で指定してください。(PSRRM命令の説明をご覧ください。)
4. この命令はJOBが停止するまで待ちます。

関連命令

PSSTART

記述例

```
PSSTOP 4,1
```

用例

位置決めFタイプ2軸ユニットをスロット4に入れた場合の例です。

```
100FUNKTION SAMPLE  (手動停止プログラム)
110 SLOTCLR
120 XQT !2,ZSTOP ……手動停止プログラムを起動
130 PSTYPE 4,2 ……Fタイプ2軸に設定
140 PSCTRL 4,1 ……運転モード
150 ZW:IF PSREADY(4)=0 THEN GOTO ZW …準備完了まで待つ
160 PSPCLR 4 ……パラメータを共有メモリに転送
170 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
180 PSPSET 4 ……パラメータをシステムメモリに転送
190 WAIT 0.2 ……位置決めユニットの処理待ち
200 PSDSET 4,1,1,E:,I:500000,10000,300,0,A:0,0 …位置データ1の設定
210 PSJOBNO 4,1,1 ……JOB1の始動するデータ番号を指定
220 PSSTART 4,1 ……JOB1を始動
230 END
240 FEND
250 FUNCTION ZSTOP
260 WAIT SW(X_&M0)=1 ……手動停止入力がONするのを
                          待つ(X_&M0:手動停止入力)
270 PSSTOP 4,1 ……JOB1を停止
280 PRINT "MANUAL STOP TASK1"
290 END
300 FEND
```

PSTYPE

オンラインコマンド　ステートメント

概要

位置決めユニットのタイプを宣言します。

書式

PSTYPE␣〈スロット番号〉,〈タイプ値〉

説明

位置決めユニットのタイプを宣言します。
〈スロット番号〉には、0～31を設定します。
〈タイプ値〉には、1～5を指定します。

1:Fタイプ1軸ユニット
2:Fタイプ2軸ユニット
3:Fタイプ3軸ユニット
4:Eタイプ1軸ユニット
5:Eタイプ2軸ユニット

注意

位置決めユニットを使用する前には、必ずこの命令を使って位置決めユニットの種類を宣言してください。

記述例

```
PSTYPE 4,2
```

用例

位置決めFタイプ2軸ユニットをスロット4に入れた場合の例です。

```
100 FUNCTION SAMPLE　(パラメータ設定プログラム例)
110 SLOTCLR
120 PSTYPE 4,2 ……Fタイプ2軸に設定
130 PSCTRL 4,1 ……運転モード
140 ZW : IF PSREADY(4)=0 THEN GOTO ZW ……準備完了まで待つ
150 PSPCLR 4 ……パラメータを共有メモリに転送
160 PSPRM 4,1,2,0 ……軸モードを独立に設定
170 PSPRM 4,1,3,0 ……単位をパルスに設定
180 PSPRM 4,1,20,&B110100 ……インターフェイス論理を設定
190 PSPSET 4 ……パラメータをシステムメモリに転送
200 END
210 FEND
```

QUIT

オンラインコマンド　ステートメント

概要

実行中または一時停止中のプログラムを終了します。

書式

QUIT␣ {!〈タスク番号〉}

説明

現在実行中、あるいは一時停止中のタスクを終了します。
〈タスク番号〉を省略したときは、タスク1を終了します。
指定したタスクがすでにQUIT状態であってもエラーにはなりません。

記述例

QUIT !2　タスク2を終了する。

用例

タスク1を終了する

```
>QUIT
>
```

タスク2を終了する

```
>QUIT !2
>
```

QUIT ALL

オンラインコマンド

概要

すべてのタスクのプログラムを終了します。

書式

QUIT␣ALL

説明

すべてのタスクのプログラムを終了します。

記述例

QUIT ALL ……すべてのタスクを終了する。

RANDMIZE

オンラインコマンド　ステートメント

概要

新しい乱数系列を設定します。

書式

RANDMIZE␣[〈数式〉]
RANDMIZE(〈数式〉)

説明

新しい乱数の種(seed)を与えることによって乱数系列を変更します。
乱数のseedは数式によって−32,768〜32,767の間で与えます。

用例

```
100 FUNCTION SAMPLE
110 INTEGER A,B
120 FOR A=1 TO 5
130 RANDMIZE(A)
140 PRINT "COUNT",A
150 FOR B=1 TO 5
160 PRINT RND(B)
170 NEXT
180 NEXT
190 END
200 FEND
```

READ%

オンラインコマンド　ステートメント

概要

高機能ユニットの共有メモリから、I/Oまたはメモリにデータを読み出します。

書式

READ%_〈スロット番号〉,〈高機能ユニットのアドレス〉,〈ワード数〉,〈CPUユニットのアドレス〉

説明

〈スロット番号〉で指定された高機能ユニットの共有メモリから〈ワード数〉分のデータを読み出し、CPUユニットのI/Oまたはメモリに書き込みます。

〈スロット番号〉には、0～31を設定します。

〈高機能ユニットのアドレス〉には、高機能ユニットの共有メモリのワードアドレスを指定します。

〈ワード数〉には、読み出すデータのワード数を指定します。

〈CPUユニットのアドレス〉は、以下のとおり指定します。

外部入力 I/O	WX_n	n=0～127
外部出力 I/O	WY_n	n=0～127
メモリ I/O	WR_n	n=0～97
リンクメモリ I/O	WL_n	n=0～127
データメモリ	DT_n	n=0～2047
リンクデータメモリ	LD_n	n=0～255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

注意

READ%命令は、共有メモリのアドレスをワード単位で指定します。

バイトアドレス	0	1	2	3	4	5	6	7
ワードアドレス	0		1		2		3	

参照

高機能ユニットの共有メモリの割り付け内容については巻末の高機能ユニットパラメータ一覧表をご覧ください。

関連命令

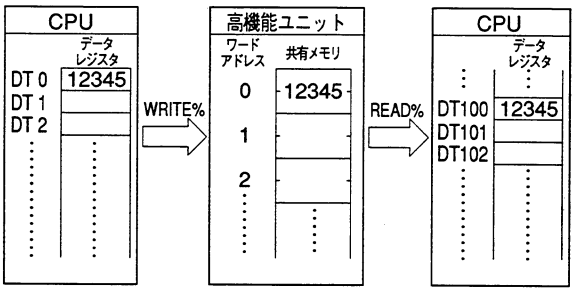
INPUT%,PRINT%,WRITE%

用例

READ% 0, 0, 2, DT_100を実行すると、スロット0の高機能ユニットの共有メモリのワードアドレス0,1の2ワードのデータをCPUユニットのDT_100およびDT_101へ書き込みます。

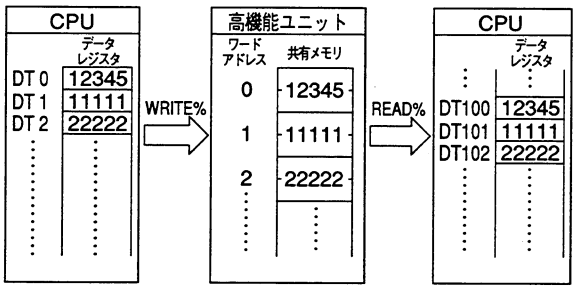
用例

データメモリの内容を高機能ユニットの共有メモリに対し書き込みを行い、別のデータメモリに読み出しを行います。




```
100 FUNCTION SAMPLE
110 '
120 '** WRITE-READ-1 **
130 '
140 OUTW DT_0,12345
150 WRITE% 2,DT_0,1,0
160 '
170 READ% 2,0,1,DT_100
180 PRINT INW(DT_100)
190 '
200 FEND
```

3つのデータレジスタの内容を高機能ユニットの共有メモリに対し書き込み・読み出しを行います。




```
100 FUNCTION SAMPLE
110 '
120 '** WRITE-READ-2 **
130 '
140 OUTW DT_0,12345
150 OUTW DT_1,11111
160 OUTW DT_2,22222
170 WRITE% 2,DT_0,3,0
180 '
190 READ% 2,0,3,DT_100
200 PRINT INW(DT_100)
210 PRINT INW(DT_101)
220 PRINT INW(DT_102)
230 '
240 FEND
```

READR

オンラインコマンド　ステートメント

概要

リモートI/O子局の高機能ユニットの共有メモリから、I/Oまたはメモリにデータを読み出します。

書式

READR␣〈親局番号〉,〈子局番号〉,〈スロット番号〉,〈高機能ユニットのアドレス〉,〈ワード数〉,〈CPUユニットのアドレス〉

説明

〈親局番号〉、〈子局番号〉、〈スロット番号〉で指定した高機能ユニットの共有メモリから、〈ワード数〉分のデータを読み出し、CPUユニットのI/Oまたはメモリに書き込みます。
〈親局番号〉には、1～4を指定します。
〈子局番号〉には、1～32を指定します。
〈スロット番号〉には、0～31を指定します。
〈高機能ユニットのアドレス〉には、高機能ユニットの共有メモリのワードアドレスを指定します。
〈ワード数〉には、読み出すデータのワード数を指定します。
〈CPUユニットのアドレス〉には、データを格納するI/O、メモリの先頭アドレスを指定します。

参照

高機能ユニットの共有メモリの割り付け内容については巻末の高機能ユニットパラメーター覧表をご覧ください。

関連命令

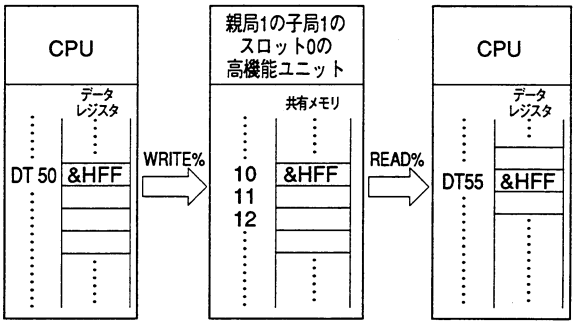
INPUTR,PRINTR,WRITER

記述例

```
READR 1,1,0,0,1,DT_2
```

用例

リモートI/O子局の共有メモリ書き込み・読み出しデータレジスタの内容を親局1の子局1のスロット0の高機能ユニットの共有メモリに対し書き込み、別のデータレジスタに読み込みます。




```
100 FUNCTION SAMPLE
110 '
120 '** WRITER-READR-1 **
130 '
140 INTEGER A
150 OUTW DT_50,&HFF
160 WRITER 1,1,0,DT_50,1,10
170 '
180 READR 1,1,0,10,1,DT_55
190 PRINT INW(DT_55)
200 FEND
```

注意

READR命令は共有メモリのアドレスをワード単位で指定します。

バイトアドレス	0	1	2	3	4	5	6	7
ワードアドレス	0		1		2		3	

REAL

ステートメント

概要

変数を4バイト実数として使用するよう宣言します。

書式

REAL␣〈変数名〉[,〈変数名〉･･･]
REAL␣〈変数名〉(〈添字の最大値〉[,〈添字の最大値〉･･･])

説明

4バイトの実数型の使用を宣言します。
変数の値の範囲(精度):
　0に最も近い値　−1.2E−38〜+1.2E−38
　0に最も遠い値　−1.2E+38〜+1.2E+38
〈変数名〉には、8文字までの英数字と「_(アンダースコア)」が使用できますが、先頭の1文字は必ず英字で書き始めなければならないという約束があります。また、Pで始まる変数名は使用できません。
注意
1.予約語は、変数名として使用することができません。
2.数値型の変数の場合末尾に記号は付けません。
3.値が代入される前の変数は、数値変数では「0」として扱われます。
4.変数宣言は、プログラム中の最初のタスクブロック(FUNCTION〜FEND)のはじめにまとめて記述します(タスクブロックの途中で変数宣言をすると正しくアップロードされない場合があります)。
5.配列変数の場合、添字の最小値は0です。

関連命令

BYTE,INTEGER,LONG,STRING

用例

```
100 FUNCTION SAMPLE
110 BYTE A
120 INTEGER B
130 LONG C
140 REAL D
150 STRING E$
```

RECV

オンラインコマンド　ステートメント

概要

ワードデータをMEWNET上の相手局から受信します。

書式

RECV␣〈自局のルートNo.〉,〈相手局のユニットNo〉,〈相手局のアドレス〉,〈ワード数〉,〈自局のアドレス〉

説明

指定したの〈自局のルートNo.〉のネットワーク上にある〈ユニットNo.〉の相手局からワードデータを受信します。
データは、〈相手局のアドレス〉で指定された相手局のI/Oまたはメモリから〈ワード数〉分だけ読み出され、〈自局のアドレス〉で指定された自局のI/Oまたはメモリへ書き込まれます。
〈自局のルートNo.〉には、1〜3を指定します。CPUに近い側のリンクユニットから1、2、3の順です。
〈相手局のユニットNo.〉には1〜63を指定します。相手局のリンクユニットのユニットNo.のスイッチと同じ番号を指定してください。
〈ワード数〉には、1〜16の範囲で受信ワード数を指定してくだい。
〈相手局のアドレス〉、〈自局のアドレス〉は、下表の範囲で指定してください。

外部入力 I/O	WX_n	n=0〜127
外部出力 I/O	WY_n	n=0〜127
メモリ I/O	WR_n	n=0〜97
リンクメモリ I/O	WL_n	n=0〜127
データメモリ	DT_n	n=0〜2047
リンクデータメモリ	LD_n	n=0〜255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

関連命令

RECVB,SEND,SENDB,STRATUM

記述例

```
RECV 1,2,DT_3,1,DT_0
```

用例

RECV 1,2,DT_100,2,DT_0を実行すると、自局のルートNo.1上のネットワーク上にあるユニットNo.2の相手局からDT_100〜DT_101の2ワードのデータを読み出し、自局のDT_0〜DT_1に書き込みます。

RECVB

オンラインコマンド　ステートメント

概要

ビットデータをMEWNET上の相手局から受信します。

書式

RECVB␣〈自局のルートNo.〉,〈相手局のユニットNo.〉,〈相手局のアドレス〉,〈相手側のビット番号〉,〈自局のアドレス〉,〈自局側のビット番号〉

説明

指定したの〈自局のルートNo.〉のネットワーク上にある〈ユニットNo.〉の相手局からビットデータを受信します。

ビットデータは、〈相手局のアドレス〉と〈相手局側のビットNo.〉で指定された相手局のメモリのビットの内容が読み出され、〈自局のアドレス〉と〈自局側のビットNo.〉で指定された自局のメモリへ書き込まれます。

〈自局のルートNo.〉には、1～3を指定します。CPUに近い側のリンクユニットから1、2、3の順です。

〈相手局のユニットNo.〉には1～63を指定します。相手局のリンクユニットユニットのスイッチと同じ番号を指定してください。

〈自局側のビットNo.〉と〈相手側のビットNo.〉には、1～16を指定します。下位ビットから1.2.3……の順に数えてください。

〈相手局のアドレス〉、〈自局のアドレス〉は、下表の範囲で指定してください。

外部入力 I/O	WX_n	n=0～127
外部出力 I/O	WY_n	n=0～127
メモリ I/O	WR_n	n=0～97
リンクメモリ I/O	WL_n	n=0～127
データメモリ	DT_n	n=0～2047
リンクデータメモリ	LD_n	n=0～255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

関連命令

RECV,SEND,SENDB,STRATUM

記述例

RECVB 1,2,WR_0,1,WR_1,1

用例

RECVB 1,2,DT_100,1,DT_0,1を実行すると、自局のルートNo.1のネットワーク上にあるユニットNo.2の相手局のDT_100のビット1の内容を読み出し、自局のDT_0のビット1に書き込みます。

RENUM

オンラインコマンド　オフラインコマンド

概要

編集中のプログラムの行番号をつけ変えます。

書式

RENUM_ [〈新行番号〉][,〈増分〉]

説明

編集中のプログラムの行番号を〈新行番号〉から始め、指定した〈増分〉で再割り付けします。
〈新行番号〉,〈増分〉は省略するとそれぞれ「10」が自動的に設定されます。「GOTO」、「GOSUB」などの分岐命令のパラメータに使用した行番号も新しく割り付けられた行番号に自動的に変更されます。

注意

行番号に指定できるのは、1～32767の範囲の整数です。

記述例

```
RENUM 5000,10
```

用例

以下のプログラムを入力した場合

```
10 FUNCTION SAMPLE
20 WAIT SW(R_0)=1
30 OFF R_0
40 OUTW WY_2,&HFF
50 GOTO 110
60 FEND
>
```

行番号を「100」で始め増分「100」に変更する場合

```
>RENUM 100,100
>LIST
100 FUNCTION SAMPLE
200 WAIT SW(R_0)=1
300 OFF R_0
400 OUTW WY_2,&HFF
500 GOTO 110
600 FEND
>
```

行番号を「100」で始め増分「10」に変更する

```
>RENUM 100,10
>LIST
100 FUNCTION SAMPLE
110 WAIT SW(R_0)=1
120 OFF R_0
130 OUTW WY_2,&HFF
140 GOTO 110
150 FEND
>
```

行番号を「10」で始め増分を「20」に変更する場合

```
>RENUM 10,20
>LIST
10 FUNCTION SAMPLE
30 WAIT SW(R_0)=1
50 OFF R_0
70 OUTW WY_2,&HFF
90 GOTO 110
110 FEND
>
```

RESET

オンラインコマンド

概要

BASICタイプCPUをリセットします。

書式

RESET

説明

BASICタイプCPUで実行中に処理をすべて終了し、プログラム、変数、パラメータメモリ、スロット割り付け以外のメモリを初期化します。
また、エラー、緊急停止および強制出力状態を解除します。
BASICタイプCPUのイニシャライズスイッチを「INITIALIZE」側に倒すのと同じ動作内容です。

関連命令

CLRVAL,CLRVAR,VERINIT

記述例

```
RESET
```

RESTORE

ステートメント

概要

DREAD文で読むDATA文を指定します。

書式

RESTORE␣[〈行番号〉]

説明

〈行番号〉が省略されると、次にくるDREAD文はプログラム中の最初のDATA文から読み始めます。
〈行番号〉を指定すると、指定された行のDATA文を読み始めます。

関連命令

DATA,DREAD

用例

```
100 FUNCTION SAMPLE
110 INTEGER A,B
120 LONG C
130 STRING D$
140 RESTORE 180 ……DREAD文で読み出すDATA文の行
                    番号を180番に指定します。
150 DREAD A,B,D,D$ ……DATA文で定義されたデータを変数
160 PRINT A,B,C,D$      A.B.C.D$にそれぞれ読み出します。
170 DATA 123,1111,45967890,"TEST"
180 DATA 321,2222,66666666,"test" …DREAD文で読み出す
190 FEND                           データを定義します。
```

RESUME

オンラインコマンド　ステートメント

概要

一時停止中のタスクの実行を再開します。

書式

RESUME␣ [!〈タスク番号〉]

説明

「HALT」コマンドで一時停止中のタスクの実行を再開します。

〈タスク番号〉を省略したときは、タスク1の実行を再開します。

注意

指定したタスクが、あらかじめ「HALT」コマンドで一時停止されていない（HALT状態ではない）ときはエラーになります。

関連命令

HALT

記述例

RESUME !2 ……一時停止しているタスク2を再開する。

用例

現在一時停止中のタスク1の実行を再開する

```
>RESUME
>
```

現在一時停止中のタスク2の実行を再開する

```
>RESUME !2
>
```

RETURN

ステートメント

概要

サブルーチンから復帰します。

書式

RETURN

説明

「GOSUB」、「ONERR」、「ONSW ( ) 」などの命令によって実行に移されたサブルーチンを終了し、実行を移した元のプログラムの命令の次の行に戻り、プログラムの実行を再開します。

関連命令

GOSUB,ON～GOSUB,ONERR,ONSW( )

用例

```
100 FUNCTION SAMPLE
110 INTEGER A,B
120 A=&M10;B=&M18
130 GOSUB LOW
140 GOSUB HIGH
150 END
160 LOW:
170 ON Y_A
180 WAIT 1
190 OFF Y_A
200 WAIT 1
210 RETURN
220 HIGH:
230 ON Y_B
240 WAIT 1
250 OFF Y_B
260 WAIT 1
270 RETURN
280 FEND
```

RFORCE

オンラインコマンド

概要

I/O、メモリの強制セットおよびリセットを解除します。

書式

RFORCE_ {ALL | 〈I/O番号〉}

説明

「FORCE」命令で強制的に「ON（セット）·OFF（リセット）」した外部入出力、メモリI/Oを強制モードから解除します。
〈I/O番号〉を省略すると、強制モードに設定されているすべてのI/Oを表示します。
〈I/O番号〉は、以下のとおり指定します。

外部入力 I/O	X_n	n=&M0～&M127F（0～2047）
外部出力 I/O	Y_n	n=&M0～&M127F（0～2047）
メモリ I/O	R_n	n=&M0～&M97F （0～1567）
リンクメモリ I/O	L_n	n=&M0～&M127F（0～2047）

〈I/O番号〉にALLを指定すると、強制モードに設定されているすべてのI/Oを強制モードから解除します。

関連命令

FORCE

記述例

```
RFORCE ALL
RFORCE Y_&M0
```

用例

外部出力ユニット「Y_0」の強制モードを解除する場合

```
>RFORCE Y_0
>
```

強制出力が設定されているI/Oを調べる場合（Y_0の接点）

```
>RFORCE
>Y_:0
>
```

強制出力が設定されていなかった場合（プロンプトのみ表示）

```
>RFORCE
>
```

RIGHT$( )

オンラインコマンド　ステートメント

概要

文字列の右端から指定した文字数分の文字列を返します。

書式

RIGHT$(〈文字列〉,〈文字数〉)

説明

〈文字列〉で指定した文字列の右端から〈文字数〉分の文字列を取り出し、その文字列を返します。
〈文字数〉が0の場合はエラーになります。
指定できる〈文字列〉の文字数の範囲は1～255です。

注意

〈文字数〉の文字数より大きい〈文字数〉を指定した場合、〈文字数〉は〈文字列〉の文字数になります。

関連命令

LEFT$(),LEN(),MID$()

記述例

```
A$=RIGHT$("ABCDEFG",2)
```

用例

文字列「"FP-BASIC"」の右から5文字を取り出す場合

```
>PRINT RIGHT$("FP-BASIC",5)
BASIC
>
```

〈文字データ〉の文字数より大きな〈文字数〉を指定した場合

```
>PRINT RIGHT$("FP-BASIC",80)
FP-BASIC ……80文字を指定しても「FP-BASIC」の
>                    8文字だけが表示される
```

プログラミング例

```
100 FUNCTION SAMPLE
110 STRING A$,B$
120 A$="FP-BASIC"  ……文字列「FP-BASIC」の
130 B$=RIGHT$(A$,4) ……右から4文字分を
140 PRINT B$ ……画面に表示する
150 FEND
```

RIOCLR

オンラインコマンド　ステートメント

概要

リモートI/Oシステムのスロット割り付けを初期化します。

書式

RIOCLR

説明

リモートI/Oシステムのリモート I/O子局が占有するスロット数、およびスロット割り付けの設定を初期化します。設定されている登録マップをクリアして、現在接続されているリモートI/O子局の占有スロット数とスロット割り付けを登録マップに登録します。

注意

登録マップについては、MEWNET-F (リモートI/Oシステム) 導入マニュアルをご覧ください。

関連命令

DUMPR,PRMR,PRMR(),RIOSET,SLOTR,SLOTR()

記述例

RIOCLR

用例

```
>PRMR 1,1,3 ·····親局番号1、子局番号1のリモートI/O子局に占有
                 スロット数3を割りつけます
>SLOTR 1,1,0,120 ·····親局番号1、子局番号1のリモートI/O子局のス
                 ロット0に16点入力ユニットを割り付けます
>RIOSET ·····PRMR.SLOTRで設定された登録マップをリ
>PRINT PRMR(1,1)  モートI/Oの現在値マップに転送します
3
>PRINT SLOTR(1,1,0)
120
>
```

```
>PRINT SLOTR(1,1,0)
120
>SLOTR 1,1,0,,220 ·····親局番号1、子局番号1のリモートI/O子
>RIOSET           局のスロット0にA/D、D/A変換ユニット
>PRINT SLOTR(1,1,0)  を割り付けます
220
>RIOCLR ·····リモートI/Oのスロット割り付けを初
>PRINT SLOTR(1,1,0)  期化します。
120
>
```

RIOSET

オンラインコマンド　ステートメント

概要

リモートI/Oシステムの子局の占有スロット数とスロット割り付けの設定値を転送します。

書式

RIOSET

説明

リモートI/Oシステムの親局と子局に、子局の占有スロット数とスロット割り付けについての現在の設定値を転送します。PRMR、SLOTRコマンドで設定されている登録マップを、リモートI/Oシステムに接続されている現在値マップに転送します。

注意

登録マップ、現在値マップについては、MEWNET-F (リモートI/Oシステム) 導入マニュアルをご覧ください。

関連命令

DUMPR,PRMR,PRMR(),RIOCLR,SLOTR,SLOTR()

記述例

RIOSET

用例

```
>PRMR 1,1,3 ·····親局番号1、子局番号1のリモートI/O子局に占有
                 スロット数3を割りつけます
>SLOTR 1,1,0,120 ·····親局番号1、子局番号1のリモートI/O子局のス
                 ロット0に16点入力ユニットを割り付けます
>RIOSET ·····PRMR.SLOTRで設定された登録マップをリ
>PRINT PRMR(1,1)  モートI/Oの現在値マップに転送します
3
>PRINT SLOTR(1,1,0)
120
>
```

```
>PRINT SLOTR(1,1,0)
120
>SLOTR 1,1,0,,220 ·····親局番号1、子局番号1のリモートI/O子
>RIOSET           局のスロット0にA/D、D/A変換ユニット
>PRINT SLOTR(1,1,0)  を割り付けます
220
>RIOCLR ·····リモートI/Oのスロット割り付けを初
>PRINT SLOTR(1,1,0)  期化します。
120
>
```

RND( )

オンラインコマンド　ステートメント

概要

乱数を返します。

書式

RND[(〈数式〉)]

説明

0以上1未満の乱数を返します。
発生する乱数は、XQTが実行されるごとに同じ乱数系列を取ります。
乱数系列を変更するには、「RANDMIZE」命令によって新しい乱数系列を設定します。発生する乱数は〈数式〉の値によって次のように異なります。
〈数式〉が0のときは、1つ前に発生した乱数の値をとります。
〈数式〉が正のときは、次の乱数を発生します。
〈数式〉が省略されたときは、正の値として次の乱数を発生します。

関連命令

RANDMIZE

RROLL( )

オンラインコマンド　ステートメント

概要

数値をビット単位で右ローテート（回転）します。

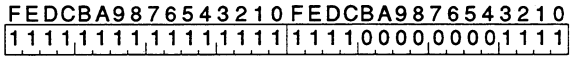
書式

RROLL(〈数式〉,〈ビット数〉)

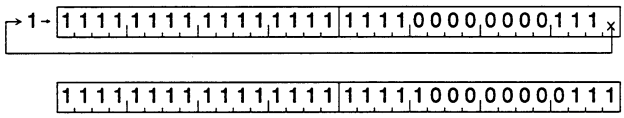
説明

〈数式〉を指定した〈ビット数〉だけ右にローテートし、その数値を返します。
対象となる〈数式〉の種類は、整数・実数およびその型の変数・関数です。
指定した〈数式〉にデータ型とローテート後のデータ型は同じになります。
〈数式〉に実数が指定された場合は、小数点以下が切り捨てられ、2バイト整数として処理されます。
32ビットの〈数式〉「&HFFFFF00F」を1ビット、右にシフトしたとき（結果は「&HFFFFF807」になります）。

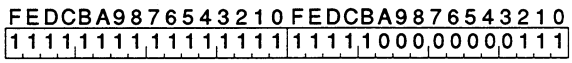
元のデータ



1ビット右へローテートします



結果



最下位ビットの情報を最上位ビットに持っていきます。

関連命令

LROLL(),LSHIFT(),RSHIFT()

用例

メモリI/Oの内容を読み込み、3ビット右ローテートして出力する場合

```
100 FUMCTION SAMPLE
110 INTEGER A,B
120 A=INW(WR_0) ……メモリI/Oの内容を読み込み
130 B=RROLL(A,3)  ……その値を3ビット右ローテートして
140 OUTW WY_2,B ……出力する
150 FEND
```

RSHIFT( )

オンラインコマンド　ステートメント

概要

数値をビット単位で右シフトします。

書式

RSHIFT(〈数式〉,〈ビット数〉)

説明

指定した〈数式〉を指定した〈ビット数〉だけ右にシフトし、その数値を返します。

対象となる〈数式〉の種類は、整数・実数およびその型の変数・関数です。

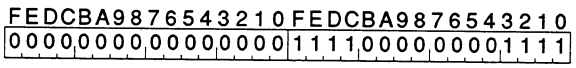
指定した〈数式〉のデータ型とシフト後のデータ型は同じになります。

〈数式〉に実数が指定された場合は、小数点以下が切り捨てられ、2バイト整数として処理されます。

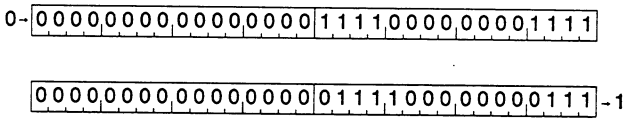
また、シフトして空いたビット（最下位ビット）には0が入ります。

32ビットの〈数式〉「&HF00F」を1ビット、右にシフトしたとき（結果は「&H7807」になります）。

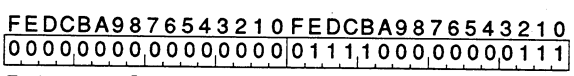
元のデータ



1ビット右へシフトします



結果



最上位1ビットに「0」が入り、最下位1ビットは捨てられます。

関連命令

LROLL(),LSHIFT(),RROLL()

用例

メモリI/Oの内容を読み込み、3ビット右シフトして出力する場合

```
100 FUMCTION SAMPLE
110 INTEGER A,B
120 A=INW(WR_0) ……メモリI/Oの内容を読み込み
130 B=RSHIFT(A,3) ……その値を3ビット右シフトして
140 OUTW WY_2,B ……出力する
150 FEND
```

SAVE

オンラインコマンド　オフラインコマンド

概要

パソコンのプログラムメモリからディスクにプログラムを書き込みます。

書式

SAVE␣"[〈ドライブ番号〉:][〈パス名〉][〈ファイル名〉[.PRG]]["]

説明

パソコンのプログラムメモリのソースプログラムをディスクに書き込みます。保存されるファイルの内容は、ASCII形式（テキストファイル）のソースプログラムです。
〈ドライブ番号〉を省略するとカレントドライブが、〈パス名〉を省略するとカレントディレクトリが、それぞれ指定されます。
〈ファイル名〉は、英数字および「_（アンダースコア）」8文字以内（数字で始まるものは使えません。）と、拡張子3文字で記述します。
ソースプログラムのファイル名の拡張子は、省略すると自動的に「.PRG」が指定されます。
〈ファイル名〉を省略するとウインドウが開きます。
ウインドウ上で参照できるファイル数は50個までです。

注意

1.書き込もうとしたディスクにすでに同名のファイルが存在したときは、「Over Write (Y/N) :」のメッセージが表示されます。
ファイルを上書きする場合は、Ⓨ を入力します。
2.〈ファイル名〉は15文字まで指定してもエラーにはなりませんが、先頭から8文字しか認識できません。したがって9文字目以降が違うファイル名は同一のファイル名として扱われます。
3.カレントディレクトリまたは指定したディレクトリに、書き込み可能なディスクがないときはエラーになります。

関連命令

FILES,LOAD

記述例

```
SAVE
SAVE "C:¥
SAVE "C:¥PC¥USER¥TEST"
```

用例

ディスクの内容が以下の場合

```
>FILES
TEST.PRG TEST_1.PRG TEST_2.PRG
>
```

プログラムを「TEST_3.PRG」として書き込む

```
>SAVE "TEST_3
>
```

すでに存在するファイル名を指定した場合

```
>SAVE "TEST_1"
Over Write (Y/N) :Y
>
```

SDERR( )

オンラインコマンド　ステートメント

概要

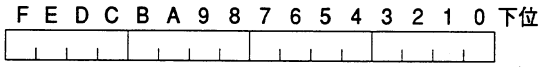
シリアルデータユニットの通信エラーステータスを返します。

書式

SDERR(〈スロット番号〉,〈チャンネル番号〉)

説明

シリアルデータユニットのシリアルデータ通信エラーステータスを返します。
ユニットの入力ポートX_&M4～X_&M8またはX_&M9～X_&MDの値を返します。
得られる値は、2バイト整数で以下のとおりです。



ビット0:受信フレーミングエラーのとき1
ビット1:受信パリティーエラーのとき1
ビット2:受信バッファフルエラーのとき1
ビット3:受信メッセージ長エラーのとき1
ビット4:送信メッセージ長エラーのとき1

〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、1または2を指定します。

参照

入力ポートX_8M4～X_8M0の割り付け内容については、巻末の高機能ユニットパラメーター一覧表をご覧ください。

関連命令

SDRD( ),SDRECV,SDRESET,SDSEND,SDTCHR,SDTERM

記述例

```
I=SDERR(0,1)
```

用例

シリアルデータユニットのCH1のエラーステータスを確認します。

```
>PRINT BIN$(SDERR(0,1))
100
```

上記の例では、受信バッファフルエラーとなっていることを示しています。

SDRDY( )

オンラインコマンド　ステートメント

概要

シリアルデータユニットの動作可能確認ステータスを返します。

書式

SDRDY(〈スロット番号〉)

説明

シリアルデータユニットのイニシャライズが完了し、動作可能状態になっているかを確認し結果を返します。
この値は、ユニットの入力ポートX_&MEの値です。
得られる値は、以下のとおりです。
〈スロット番号〉には、0～31を指定します。
　0:未完了（動作できません）
　1:完了（動作できます）

参照

入力ポートX_8MEの割り付け内容については、巻末の高機能ユニットパラメーター一覧表をご覧ください。

関連命令

SDER( ),SDRECV,SDRESET,SDSEND,SDTCHR,SDTERM

記述例

```
PRINT SDRDY(4)
```

用例

スロット4のシリアルデータユニットが動作可能状態にあるか確認します。

```
>PRINT SDRDY(4)
1
```

上記の例は動作可能状態にあることを示しています。

SDRECV

ステートメント

概要

シリアルデータユニットからデータを受信します。

書式

SDRECV_〈スロット番号〉,〈チャンネル番号〉,〈変数〉・・・

説明

シリアルデータユニットからデータを受信します。受信データの有無チェックから受信完了の確認までを行ないます。動作は以下のとおりです。(スロット0チャンネル1の場合)

```
WAIT SW (X_&M1) 'データを受信するまで待つ
INPUT %          '変数読み出し
ON Y_&M11
WAIT SW (X_&M1) =0
OFF Y_&M11
```

一度に受信できるデータ量は、1チャンネルあたり最大500バイト (250ワード) です。

受信バッファから読み込むとき、変数に数値変数を指定している場合は、数値ASCII文字列は自動的にバイナリーデータに変換されます (区切り記号には、「, (コンマ) 」が使用されます) 。

〈スロット番号〉には、0～31を指定します。

〈チャンネル番号〉には、1または2を指定します。

関連命令

SDERR(),SDRD(),SDRESET,SDSEND,SDTCHR,SDTERM

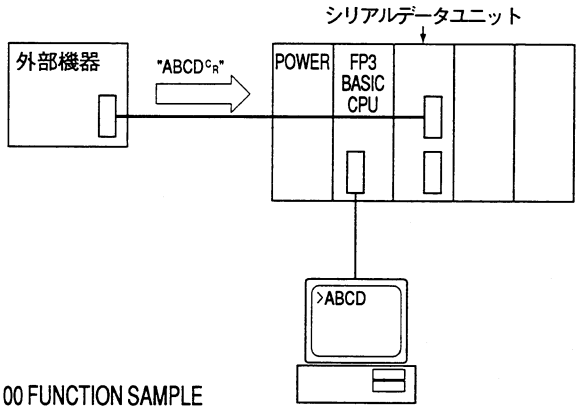
記述例

SDRECV 1,1,A$

用例

シリアルデータユニット

外部機器から受信したデータを画面に表示します。




```
100 FUNCTION SAMPLE
110 STRING A$
120 SDRECV 0,1,A$ ……スロット0のシリアルデータユニットのCH1で受信したデータをA$に格納します。
130 PRINT A$ ……A$の内容を画面に表示します。
140 FEND
```

SDRESET

ステートメント

概要

シリアルデータユニットをリセットします。

書式

SDRESET␣〈スロット番号〉

説明

シリアルデータユニットをソフトウェアリセットします。この命令を実行すると、シリアルデータユニットはイニシャライズされます。

動作は以下のとおりです。(スロット0の場合)

```
ONY_&M1D
WAITSW(X_&ME)=0
OFFY_&M1D
WAITSW(X_&ME)
```

〈スロット番号〉には、0～31を指定します。

関連命令

SDERR(),SDRD(),SDRECV,SDSEND,SDTCHR,SDTERM

記述例

```
SDRESET1
```

SDSEND

オンラインコマンド ステートメント

概要

シリアルデータユニットからデータを送信します。

書式

SDSEND_〈スロット番号〉,〈チャンネル番号〉,〈送信データ〉･･･

説明

シリアルデータユニットから、接続されている機器に対してデータを送信します。このとき、送信可能チェックから送信完了確認までおこないます。

動作は以下のとおりです。(スロット0、チャンネル1の場合)

```
WAIT SW (X_&M0) =0
PRINT %----'送信データ
ON Y_&M10
WAIT SW (X_&M0)
OFF Y_&M10
```

一度に送信できるデータ量は、1チャンネルあたり最大500バイト (250ワード) です。〈送信データ〉は、「, (コンマ) 」で区切ります。

送信バッファへ書き込むとき、数値データは自動的にASCII文字列に展開されます。

〈スロット番号〉には、0～31を指定します。

〈チャンネル番号〉には、1または2を指定します。

〈送信データ〉には、変数または定数を指定します。

注意

1. SDSENDを実行する場合、終端コードは自動的に付けて送信されます。(デフォルト:CR)
2. 終端コードをCR (0DH) 以外にする場合、シリアルデータユニットのディップスイッチの設定とSDTCHR命令で終端コードの設定を行ってください。(ディップスイッチの設定だけではCPUは認識しません。)
3. 終端コードSTX (02H) を付けて送信する場合は、送信データの先頭に始端コードを付けてください。(CHR$ (&H02))

関連命令

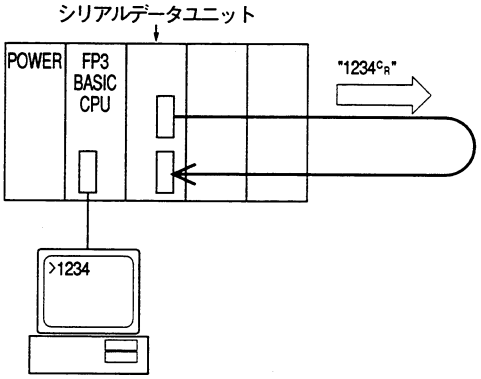
SDERR(),SDR(),SDRECV,SDRESET,SDTCHR,SDTERM

記述例

```
SDSEND 0,1,"ABCD0123"
```

用例

文字変換の内容をCH1から送信しそれをCH2で受信し画面に表示します。




```
100 FUNCTION SAMPLE
110 STRING A$ B$
120 SDTCHR 0,1,1 ･･･････････ スロット0のシリアルデータユニットのCH1の
130 A$="%1234"               終端コードを"CR"に設定します。
140 SDSEND 0,1,A$ ･･･････････ 文字列"%1234"をCH1から送ります
150 '                        (終端コード"CR"も付きます)
160 SDRECV 0,2,B$ ･･･････････ CH2で受信しそのデータをB$に格納します。
170 PRINT B$ ･･･････････････ B$の内容を画面に表示します。
180 FEND
```

SDTCHR

オンラインコマンド　ステートメント

概要

シリアルデータユニットで送受信するデータの終端コードを設定します。

書式

SDTCHR␣〈スロット番号〉,〈チャンネル番号〉,〈終端コードタイプ〉[,〈終端コード〉]

説明

シリアルデータユニットで送受信するデータの終端コードを設定します。

〈終端コードタイプ〉に「0」を設定したときは、〈終端コード〉を設定する必要があります。

〈スロット番号〉には、0～31を指定します。

〈チャンネル番号〉には、1または2を指定します。

〈終端コードタイプ〉は、以下のとおり指定します。

0:共有メモリ設定
1:CR (0Dh) ・・・・・デフォルト値
2:CR (0Dh) ,LF (0Ah)
3:ETX (O3h)

〈終端コード〉は、〈終端コードタイプ〉が「0」のときのみ、以下のとおり設定します。

数値なら下位8ビットのみ有効。

文字列ならば先頭1文字のみ有効。(文字数がないときは0 (nul))

設定値は、ユニットの共有メモリのワードアドレス1001および1002に書き込まれます。

注意

設定値は必ずシリアルデータユニットのディップSWの設定と同じにしてください。

参照

共有メモリの割り付け内容については巻末の高機能ユニットパラメーター覧表をご覧ください。

関連命令

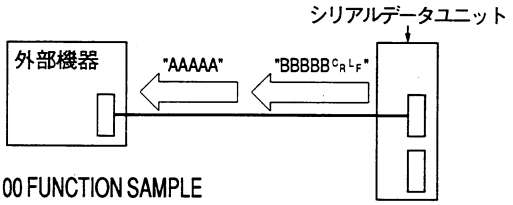
SDERR(),SDRD(),SDRECV,SDRESET,SDSEND,SDTERM

記述例

SDTCHR 4,1,1

用例

終端コードを付けずにデータを送信し、その後に終端コードを付けてデータを送信します。




```
100 FUNCTION SAMPLE
110 STRING A$ B$
120 SDTCHR 0,1,2 ……CH1の終端コードを"CR+LF"に設定します
130 '
140 A$="AAAAA"
150 SDTERM 0,1 ……終端コードを出さない。
160 SDSEND 0,1,A$ ……文字列"AAAAA"を送信します。
170 '
180 B$="BBBBB"
190 SDTERM 0,0 ……終端コードを出す。
200 SDSEND 0,1,B$ ……文字列"BBBBB"を送信します。
210 FEND                (終端コードもいっしょに送信します)
```

SDTERM

ステートメント

概要

シリアルデータユニットの終端コードの送信指定を設定します。

書式

SDTERM_〈スロット番号〉,〈設定コード〉

説明

シリアルデータユニットの、終端コードを送信するか、しないかを設定します。
この設定値は、ユニットの出力ポートY_&M1E、Y_&M1Fに書き込まれます。
〈スロット番号〉には、0～31を指定します。
〈設定コード〉は以下のとおり指定します。
0:CH1、CH2とも終端コードを送信する。
1:CH1は終端コードを送信しない。CH2は送信する。
2:CH1は終端コードを送信する。CH2は送信しない。
3:CH1、CH2とも終端コードを送信しない。

参照

出力ポートY_&M1E、Y_&M1Fの設定内容については、巻末の高機能ユニットパラメーター一覧表をご覧ください。

関連命令

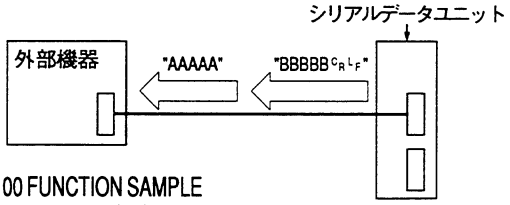
SDERR(),SDRD(),SDRECV,SDRESET,SDSEND,SDTCHR

記述例

```
SDTERM 4,0
```

用例

終端コードを付けずにデータを送信し、その後に終端コードを付けてデータを送信します。




```
100 FUNCTION SAMPLE
110 STRING A$ B$
120 SDTCHR 0,1,2 ……CH1の終端コードを"CR+LF"に設
130'                 定します。
140 A$="AAAAA"
150 SDTERM 0,1 ……終端コードを出さない。
160 SDSEND 0,1,A$ ……文字列"AAAAA"を送信します。
170'
180 B$="BBBBB"
190 SDTERM 0,0 ……終端コードを出す。
200 SDSEND 0,1,B$ ……文字列"BBBBB"を送信します。
200 FEND             (終端コードもいっしょに送信します)
```

SDTXACK ( )

オンラインコマンド　ステートメント

概要

シリアルデータユニットのデータ送信完了ステータスを返します。

書式

SDTXACK(〈スロット番号〉,〈チャンネル番号〉

説明

シリアルデータユニットが、データの送信を完了したかを確認し、結果を返します。
この値は、シリアルデータユニットの入力ポートX_&M00またはX_&M02の値です。
動作は以下のとおりです。

```
IFB SW(X_&M00)  THEN     'X_&M02(チャンネル2の場合)
    OFF Y_&M10           'Y_&M12 (チャンネル2の場合)
ELSE
    NOP                  'ノーオペレーション
ENDIF
```

得られる値は以下のとおりです。
0:送信中 (X_&M00=0/X_&M02=0)
1:送信完了X_&M00=1/X_&M02=1)
〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、1または2を指定します。

SDTXRDY( )

オンラインコマンド　ステートメント

概要

シリアルデータユニットのデータ送信可能ステータスを返します。

書式

SDTXRDY〈スロット番号〉,〈チャンネル番号〉

説明

シリアルデータユニットが、データの送信ができる状態にあるかどうかを確認し、結果を返します。
この値は、シリアルデータユニットの入力ポートX_&M00またはX_&M02および出力ポートY_&M10またはY_&M12の値です。
得られる値は以下のとおりです。

0:送信不可　X_&M00=1　OR　Y_&M10=1
　　X_&M02=1　OR　Y_&M12=1
0:送信可能　X_&M00=0　OR　Y_&M10=0
　　X_&M02=0　OR　Y_&M12=0

〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、1または2を指定します。

SDWRX( )

ステートメント

概要

シリアルデータユニットが受信待ち状態で待機します。

書式

SDWRX〈スロット番号〉,〈チャンネル番号〉

説明

シリアルデータユニットがデータを受信するまで待機します(受信データがあるまでWAITします)。シリアルデータユニットの入力ポートX_&M01またはX_&M03がONするまで待機します。
動作は以下のとおりです。

```
WAIT SW(X_&M01)    'X_&M03
```

〈スロット番号〉には、0～31を指定します。
〈チャンネル番号〉には、1または2を指定します。

SEGT( )

オンラインコマンド　ステートメント

概要

数値を7セグメント表示用データに変換します。

書式

SEGT␣〈数式〉

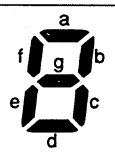
説明

指定した〈数式〉を7セグメント表示用データに変換します。
〈数式〉はBCDデータとして扱われ、4ビット単位で処理されます。
〈数式〉に実数を指定すると、小数点以下が四捨五入され、2バイト整数として扱われます。
指定した〈数式〉の変換前の「型」と変換後の数値は同じ「型」になります。

注意

変換対応表を以下に示します。

変換前		変換後	
16進表記	2進表記	16進表記	2進表記
			-gfedcba
&H0	&B0000	&H3F	&B00111111
&H1	&B0001	&H06	&B00000110
&H2	&B0010	&H5B	&B01011011
&H3	&B0011	&H4F	&B01100110
&H4	&B0100	&H66	&B01100110
&H5	&B0101	&H6D	&B01101101
&H6	&B0110	&H7D	&B01111101
&H7	&B0111	&H27	&B00100111
&H8	&B1000	&H7F	&B01111111
&H9	&B1001	&H6F	&B01101111
&HA	&B1010	&H77	&B01110111
&HB	&B1011	&H7C	&B01111100
&HC	&B1100	&H39	&B00111001
&HD	&B1101	&H5E	&B01011110
&HE	&B1110	&H79	&B01111001
&HF	&B1111	&H71	&B01110001



SELECT～CASE

ステートメント

概要

数式の値と一致するCASE文を実行します。

書式

SELECT␣〈数式〉～CASE␣〈数値〉～[CASE␣〈数値〉～][DEFAULT～]SELEND

説明

プログラム中の〈数式〉の値によって、実行する命令を選択します。
〈数値〉には、2バイト整数を指定してください。
変数は使用できません。
「CASE〈数値〉」は、「SELEND」を指定するまで、複数行指定できます。
「DEFAULT」の後には、「CASE〈数値〉」に一致しない場合の実行文を記述します。

注意

「SELECT」のネスティングは、最高10段まで可能です。
「SELECT」内の「CASE」と「DEFAULT」の数は合わせて13個まで可能です。

用例

```
100 FUNCTION SAMPLE
110 INTEGER I
120 FOR I=1 TO 5
130 SELECT I ·····数式Iの値によって実行する命
140 CASE 1          令を選択します。
150 ON Y_1          Iの値が1の時実行します。
160 WAIT 1
170 CASE 2
180 ON Y_2          Iの値が2の時実行します。
190 WAIT 1
200 DEFAULT
210 ON Y_7          Iの値が1と2でない時に実行します。
220 WAIT 1
230 SELEND
240 NEXT
250 FEND
```

SEND

オンラインコマンド ステートメント

概要

ワードデータをMEWNET上の相手局に送信します。

書式

SEND_〈自局のルートNo.〉,〈相手局のユニットNo.〉,〈自局のアドレス〉,〈ワード数〉,〈相手局のアドレス〉

説明

指定した〈自局のルートNo.〉のネットワーク上にある、〈相手局のユニットNo.〉で指定された相手局へワードデータを送信します。

データは、〈自局のアドレス〉で指定した自局のI/Oまたはメモリから〈ワード数〉分だけ読み出され、〈相手局のアドレス〉で指定した相手局のI/Oまたはメモリへ書き込まれます。

〈自局ルートNo.〉には、1～3を指定します（CPUユニットに近い方のリンクユニットから1、2、3と数えます）。

〈相手局のユニットNo.〉には、1～63を指定します。相手局のリンクユニットの局番設定スイッチの番号を設定してください。

〈ワード数〉には、1～16を指定します。

〈自局のアドレス〉〈相手局のアドレス〉は、下表の範囲で指定してください。

外部入力 I/O	WX_n	n=0～127
外部出力 I/O	WY_n	n=0～127
メモリ I/O	WR_n	n=0～97
リンクメモリ I/O	WL_n	n=0～127
データメモリ	DT_n	n=0～2047
リンクデータメモリ	LD_n	n=0～255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

関連命令

RECV,RECVB,SENDB,STRATUM

記述例

SEND 1,2,DT_0,1,DT_0

用例

SEND 1,2,DT_0,2,DT_100を実行すると、自局からルートNo.1上にあるユニットNo.2の相手局へ、 自局のDT_0～DT_1の2ワードのデータを相手局のDT_100～DT_101に書き込みます

SENDB

オンラインコマンド ステートメント

概要

ビットデータをMEWNET上の相手局に送信します。

書式

SENDB_〈自局のルートNo.〉,〈相手局ユニットNo.〉,〈自局のアドレス〉,〈自局のビット番号〉,〈相手局のアドレス〉,〈相手局のビット番号(2)〉

説明

指定した〈自局のルートNo.〉のネットワーク上にある、〈相手局のユニットNo.〉で指定された相手局へビットデータを送信します。

ビットデータは、〈自局のアドレス〉と〈自局のビットNo.〉で指定された自局のI/Oまたはメモリの内容から読み出され、〈相手局のアドレス〉と〈相手局のビットNo.〉で指定した相手局のI/Oまたはメモリへ書き込まれます。

〈自局ルートNo.〉には、1～3を指定します。CPUユニットに近い方のリンクユニットから1、2、3の順です。

〈相手局のユニットNo.〉には、1～63を指定します。相手局のリンクユニットの局番設定スイッチの番号を設定してください。

〈自局のビットNo.〉と〈相手局のビットNo.〉には、1～16を指定します。

下位ビットから1、2、3……のように数えてください。

〈自局のアドレス〉〈相手局のアドレス〉は、下表の範囲で指定してください。

外部入力 I/O	WX_n	n=0～127
外部出力 I/O	WY_n	n=0～127
メモリ I/O	WR_n	n=0～97
リンクメモリ I/O	WL_n	n=0～127
データメモリ	DT_n	n=0～2047
リンクデータメモリ	LD_n	n=0～255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

関連命令

RECV,RECVB,SEND,STRATUM

記述例

SENDB 1,2,WR_1,WR_0,1

用例

SENDB 1,2,DT_0,1,DT_100,1を実行するとNo.1自局のルートのネットワーク上にあるNo.2ユニットの相手局のDT_100のビット1へ自局のDT_0のビット1の内容を書き込みます。

SIN( )

オンラインコマンド　ステートメント

概要

正弦(サイン)を返します。

書式

SIN(〈数式〉)

説明

〈数式〉の値に対する正弦(サイン)の値を返します。
〈数式〉の単位はラジアンです。
角度が度で与えられている場合は、ラジアンに変換するためにπ／180を掛けます。
〈数式〉の型に関係なく単精度の値を返します。
(π=3.14159265358979323846･･･)

関連命令

ATN(),COS(),TAN()

用例

```
>PRINT SIN(3.1415926/180*30)
0.5
>
```

SLOT

オンラインコマンド　ステートメント

概要

ユニットの構成に応じてスロット割り付けをします。

書式

SLOT␣〈スロット番号〉,〈機能番号〉

説明

〈スロット番号〉で指定したユニットに、〈機能番号〉を割り付けます。
〈スロット番号〉は、0～23の数値で指定します。
〈機能番号〉は、3桁の整数で指定します。

参照

〈機能番号〉の詳細については巻末の『ユニットのI/O点数と機能番号』をご覧ください。

関連命令

SLOT(),SLOTCLR

記述例

```
SLOT 0,130
```

用例

```
>SLOT 1,103 ･･･････････････････････ スロットを32点出力ユニットに設定する
>PRINT SLOT(1) ･･･････････････････ 設定内容を確認する
103 ･･･････････････････････････････ 103(32点出力ユニット)に設定されている
>
```

SLOT( )

オンラインコマンド　ステートメント

概要

スロット割り付けの設定値を返します。

書式

SLOT(〈スロット番号〉)

説明

〈スロット番号〉で指定したユニットの機能番号を返します。
〈スロット番号〉は、0～23の数値で指定します。

関連命令

SLOT,SLOTCLR

記述例

PRINT SLOT(1)

用例

スロット1の設定内容を画面に表示する場合

```
>SLOT 1,103 ……スロットを32点出力ユニットに設定する
>PRINT SLOT(1) ……設定内容を確認する
103 ……103(32点出力ユニット)に設定されている
>
```

SLOTCLR

オンラインコマンド　ステートメント

概要

スロット割り付けを初期化します。(フリーロケーション)

書式

SLOTCLR

説明

設定されているスロット割り付けの設定値を初期化し、各スロットのI/Oのユニットのタイプを調べ、スロット番号およびI/O番号を自動的に割り付けます。

注意

スロット割り付けは、「VERINIT」命令でも初期化されます。

関連命令

SLOT,SLOT(),VERINIT

記述例

SLOTCLR

用例

```
>SLOT 1,103 ……スロットを32点出力ユニットに設定する
>PRINT SLOT(1) ……スロット1の設定内容を画面表示する
103 ……103(32点出力ユニット)に設定されている
                    I/Oマップを初期化する
>SLOTCLR ……スロット1の設定内容を画面表示する
PRINT SLOT(1) ……初期化され、現在の内容が表示される
102
>
```

SLOTR

オンラインコマンド　ステートメント

概要

リモートI/O子局のスロット割り付けをします。

書式

SLOTR␣〈親局番号〉,〈子局番号〉,〈スロット番号〉,〈機能番号〉

説明

〈親局番号〉、〈子局番号〉、〈スロット番号〉で指定されたユニットに、〈機能番号〉の割り付けを登録マップに登録します。

〈親局番号〉には、1～4を指定します。

〈子局番号〉には、1～32を指定します。

〈スロット番号〉には、0～31を指定します。

参照

〈機能番号〉については、巻末の「ユニットのI/O点数と機能番号」をご覧ください。

登録マップについては、MEWNET－F（リモートI/Oシステム）導入マニュアルをご覧ください。

関連命令

DUMPR,PRMR,PRMR(),RIOCLR,RIOSET,SLOTR()

記述例

SLOTR 1,1,2,120

用例

```
>PRMR 1,1,3 ·····親局番号1、子局番号1のリモートI/O子
  ·····          局に占有スロット数3を割りつけます
>SLOTR 1,1,0,120  親局番号1、子局番号1のリモートI/O子
                  局のスロット0に16点入力ユニッ
                  トを割り付けます
>RIOSET ·····PRMR.SLOTRで設定された登録マッ
>PRINT PRMR(1,1)  プをリモートI/Oの現在値マップ
3                 に転送します
>PRINT SLOTR(1,1,0)
120
>
```

```
>PRINT SLOTR(1,1,0)
120
>SLOTR 1,1,0,,220 ·····親局番号1、子局番号1のリモートI/O子
>RIOSET            局のスロット0にA/D、D/A変換ユ
>PRINT SLOTR(1,1,0) ニットを割り付けます
220
>RIOCLR ·····リモートI/Oのスロット割り付け
>PRINT SLOTR(1,1,0) を初期化します
120
>
```

SLOTR( )

オンラインコマンド　ステートメント

概要

リモートI/O子局のスロット割り付けの設定値を返します。

書式

SLOTR(〈親局番号〉,〈子局番号〉,〈スロット番号〉)

説明

〈親局番号〉、〈子局番号〉、〈スロット番号〉で指定されたスロットに対するスロット割り付けの設定値(機能番号)を返します。
〈親局番号〉には、1～4を指定します。
〈子局番号〉には、1～32を指定します。
〈スロット番号〉には、0～31を指定します。

関連命令

DUMPR,PRMR,PRMR(),RIOCLR,RIOSET,SLOTR

記述例

```
PRINT SLOTR(1,1,2)
```

用例

```
>PRMR 1,1,3 ……親局番号1、子局番号1のリモートI/O子
                局に占有スロット数3を割りつけます
>SLOTR 1,1,0,120 ……親局番号1、子局番号1のリモートI/O子
                局のスロット0に16点入力ユニットを
                割り付けます
>RIOSET ……PRMR.SLOTRで設定された登録マッ
>PRINT PRMR(1,1)  プをリモートI/Oの現在値マップ
3               に転送します
>PRINT SLOTR(1,1,0)
120
>


>PRINT SLOTR(1,1,0)
120
>SLOTR 1,1,0,,220 ……親局番号1、子局番号1のリモートI/O子
>RIOSET           局のスロット0にA/D、D/A変換ユ
>PRINT SLOTR(1,1,0)  ニットを割り付けます
220
>RIOCLR ……リモートI/Oのスロット割り付け
>PRINT SLOTR(1,1,0)  を初期化します。
120
>
```

SPACE$( )

オンラインコマンド　ステートメント

概要

指定した文字数分の空白(スペース)を返します。

書式

SPACE$(〈文字数〉)

説明

指定した〈文字数〉分の空白(スペース)を文字列として返します。
文字数が0の場合はエラーになります。

記述例

```
A$=SPACE$(10)
```

用例

「SPACE$」命令を使って、文字「F」と「P」の間に50個の空白(スペース)を入れる場合

```
>LIST
>PRINT "F"+SPACE$(50)+"P"
F                                                  P
>
```

50文字文の空白(スペース)を「F」と「P」の間に挿入して画面に表示するプログラミング例

```
100 FUNCTION SAMPLE
110 STRING A$,B$
120 A$=SPACE$(50)
130 B$="F"+A$+"P"
140 PRINT B$
150 FEND
```

SQR( )

オンラインコマンド　ステートメント

概要

数値の平方根(スクェアールート)を返します。

書式

SQR(〈数式〉)

説明

〈数式〉で指定された値の平方根を返します。
〈数式〉の型に関係なく単精度の値を返します。

記述例

```
PRINT SQR(2)
```

用例

```
100 FUNCTION SAMPLE
110 REAL A
120 READ A
130 PRINT SQR(A)
140 DATA 2, 1, 4
150 END
160 FEND
```

STEP

オンラインコマンド

概要

PAUSE命令により一時停止しているタスクの実行をステップモードで再開します。

書式

STEP␣ [!〈タスク番号〉]

説明

「PAUSE」命令による一時停止状態からタスクの実行を再開します。ただし、「CONT」命令と違い、1行実行するごとに一時停止状態になります(再び実行するにはSTEPまたはCONTを入力します)。
〈タスク番号〉を省略すると、一時停止しているすべてのタスクが実行を再開します。
「PAUSE」状態のタスクがないときは、この命令は無視されます。

タスク2の一時停止(PAUSE)の状態を解除し、次の1行を実行する場合

```
>STEP !2
```

すべての一時停止(PAUSE)状態のタスクの次の1行を実行する場合

```
>STEP
```

関連命令

CONT,PAUSE,TEST

記述例

```
STEP !1
STEP
```

STRATUM

オンラインコマンド　ステートメント

機能

データ転送命令が実行される相手ユニットの階層を指定します。

書式

STRATUM〈階層の深さ〉,〈1階層目のユニット番号〉,〈2階層目のルート番号〉,〈2階層目のユニット番号〉,〈3階層目のルート番号〉,〈3階層目のユニット番号〉,〈4階層目のルート番号〉,

階層の深さ	0～3
1階層目のユニット番号 2階層目のユニット番号 3階層目のユニット番号	1～64
2階層目のルート番号 3階層目のルート番号 4階層目のルート番号	1～6

説明

データ転送命令（SEND、RECV、SENDB、RECVB）が実行される相手ユニットがリンク階層のどの位置にあるかを指定します。

相手ユニットが自機と同じ階層にあるときは、〈階層の深さ〉で0を指定し、以下のパラメータは不要です。

相手ユニットが2階層目にあるときは、〈階層の深さ〉で1を指定し、2階層目のユニット番号、3階層目のルート番号までを指定します。

相手ユニットが3階層目にあるときは、〈階層の深さ〉で2を指定し、2階層目のユニット番号、3階層目のルート番号までを指定します。

相手ユニットが4階層目にあるときは、〈階層の深さ〉で3を指定し、3階層目のユニット番号、4階層目のルート番号までを指定します。

〈ルート番号〉は、CPUから数えて何台目のリンクユニットかで指定します。

この命令は全てのタスクについて有効となります。

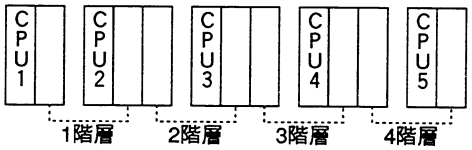
階層の指定は、次にこの命令を実行するまで有効です。

関連命令

SEND,RECV,SENDB,RECVB

使用例

下図のように構成されたシステムで、CPU1のDT_100から500ワードのデータをCPU5のFL_100へ送信するには、次のようにします。




```
STRATUM 3,2,2,4,2,6,2 ……階層を指定
SEND 1,8,DT_100,500,FL_100 ……送信命令
```

STR$( )

オンラインコマンド　ステートメント

概要

数値を文字列に変換します。

書式

STR$(〈数式〉)

説明

〈数式〉を数値型から文字型に型変換して返します。

関連命令

VAL()

記述例

```
A$=STR$(123)
```

用例

数値「123」を文字「"123"」に変換する場合

```
>PRINT STR$(123)
123
>
```

数値「123」を文字数（桁数）を調べる場合

```
>PRINT LEN(123)
3
>
```

文字型に変換してから文字数（桁数）を調べる場合

```
>PRINT LEN(STR$(123))
3
>
```

数値「123」を文字型（「"123"」）に変換し文字数を調べて画面に表示するプログラミング例

```
100 FUNCTION SAMPLE
110 INTEGER A,C
120 STRING B$
130 A=123
140 B$=STR$(A)
150 C=LEN(B$)
160 PRINT C
170 FEND
```

STRING

ステートメント

概要

変数を文字列変数として使用するよう宣言します。

書式

STRING_〈変数名〉[,〈変数名〉･･･]
STRING_〈変数名〉(〈添字の最大値〉[,〈添字の最大値〉･･･])

説明

文字型の変数の使用を宣言します。
〈変数名〉には、8文字までの英数字と「_（アンダースコア）」が使用できますが、先頭の1文字は必ず英字で書き始めるという約束があります。
また、文字型の変数には、変数名の末尾に「$」を付けなければなりません。この場合、末尾の「$」を含めて、変数名は8文字以内でなければなりません。
文字型変数のサイズは、通常1～80文字です。ただし、「$」の後に、「＊」と0～255までの数値を記述することにより、変数のサイズを任意に決めることができます。この場合、末尾の「$」を含めて、変数名は8文字以内でなければなりません。また、Pで始まる変数名は使用できません。

注意

1.予約語は、変数名として使用することができません。
2.文字型の変数（例:A$,DATA1$）では変数の末尾に「$」記号を付けます（数値型の変数の場合末尾に記号は付けません）。
3.値が代入される前の変数は、文字変数では「空の文字列（""ヌルストリング）」として扱われます。
4.変数宣言は、プログラム中の最初のタスクブロック（FUNCTION～FEND）のはじめにまとめて記述します。（タスクブロックの途中で変数宣言をすると正しくアップロードされない場合があります）。
5.配列変数の場合、添字の最小値は0です。

関連命令

BYTE,INTEGER,LONG,REAL

用例

```
100 FUNCTION SAMPLE
110 BYTE A
120 INTEGER B
130 LONG C
140 REAL D
150 STRING E$
 ・
 ・
 ・
300 FEND
```

SVCR

オンラインコマンド　ステートメント

概要

共有メモリの状態をディスクにセーブします。

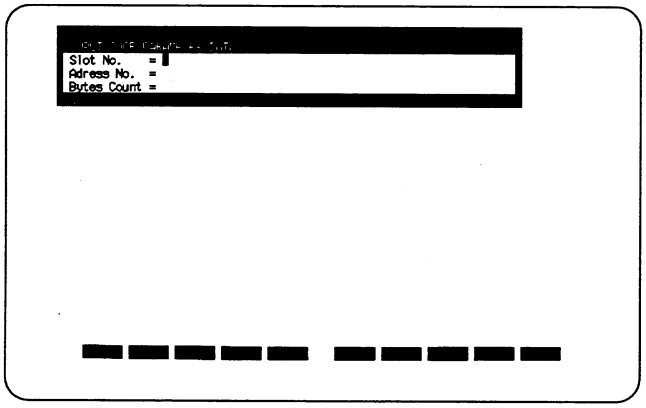
書式

SVCR_〈スロット番号〉,〈アドレス〉,〈バイト数〉,["〈ファイル名〉"]

説明

〈スロット番号〉で指定した高機能ユニットの共有メモリの〈アドレス〉から〈バイト数〉分を、〈ファイル名〉で指定したファイルにセーブします。

〈アドレス〉、〈バイト数〉、〈ファイル名〉が省略されたときは、次のようなウィンドウでの選択となります。



〈ファイル名〉の拡張子には「.CMN」を指定しますが、拡張子を省略しても自動的に「.CMN」が付けられます。

関連命令

LDCR,LDIO,SVIO

記述例

SVCR 0,0,10,"TEST2"

SVIO

オンラインコマンド　ステートメント

概要

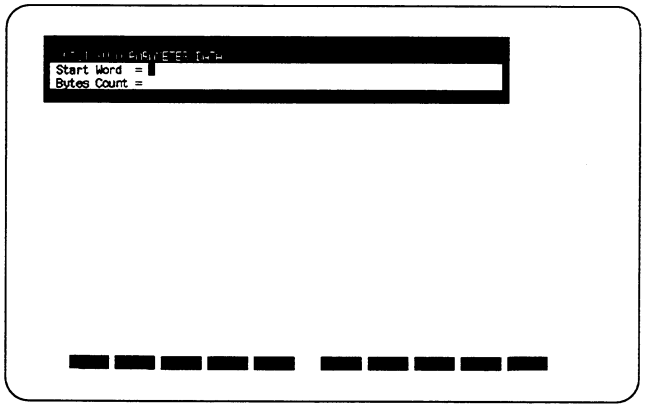
I/Oの状態をディスクにセーブします。

書式

SVIO␣[〈先頭I/O番号のワードアドレス〉,〈バイト数〉,"〈ファイル名〉"]

説明

〈先頭I/O番号のワードアドレス〉から〈バイト数〉分を、〈ファイル名〉で指定したファイルにセーブします。
なお、〈先頭I/O番号のワードアドレス〉には、I/Oの種別は付けません。
〈ファイル名〉の拡張子により、セーブするI/Oの種別を自動的に判断します。
[〈先頭I/O番号のワードアドレス〉,〈バイト数〉,"〈ファイル名〉"]が省略されたときは、次のようなウィンドウでの選択になります。



〈ファイル名〉には、必ず以下の拡張子を指定します。

出力I/O	.Y
メモリI/O	.R
データメモリ	.DT
リンクメモリ	.L

関連命令

LDCR,LDIO,SVCR

記述例

```
SVIO 0,8,"TEST1.Y"
```

用例

```
>SVIO 1,2,"TEST.Y"  ……「TEST.Y」というファイル名で
                        出力I/Oをセーブする
Over Write (Y/N) : Y ……すでに同名ファイルが存在す
                        るときに表示されます
>FILES
FPB.EXE  SPEED.COM  RSDRV.SYS  PRINT.SYS
CONFIG.SYS AUTOEXDC.BAT TEST.PRG  TEST.OBJ
TEST.SYM TEST.Y
>
```

```
>FILES
FPB.EXE SPEED.COM RSDRV.SYS PRINT.SYS
CONFIG.SYS AUTOEXDC.BAT TEST.PRG TEST.OBJ
TEST.SYM TEST.Y
>LDIO "TEST.Y"  ……出力I/Oファイル「TEST.Y」を
>                  ロードする
```

SVVAL

オンラインコマンド

概要

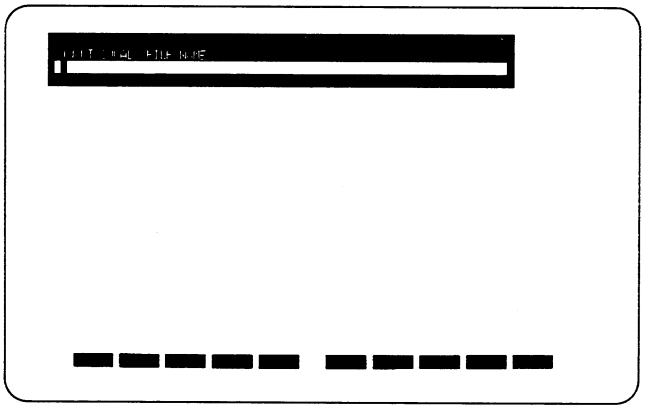
グローバル変数の値をディスクにセーブします。

書式

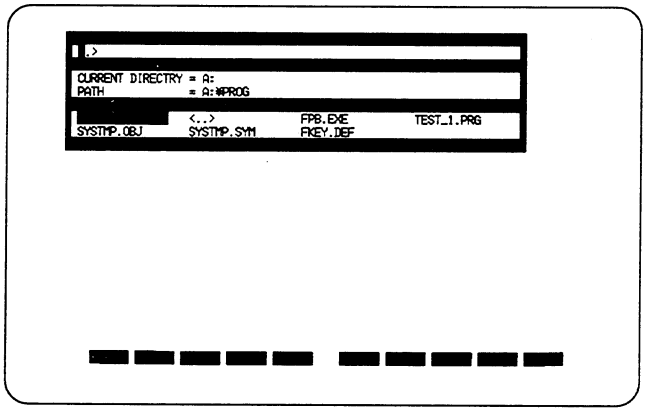
SVVAL␣["〈ファイル名〉"]

説明

グローバル変数データをすべてファイルにセーブします。〈ファイル名〉を省略した場合は次のようなウィンドウが表示されます。、



入力したファイル名にディレクトリ名またはワイルドカード(*、?)を使用した場合はさらに次のウィンドウが表示されます。また、[↵]キーのみを押した場合も、"*.*"が入力されたものとして同様のウィンドウが表示されます。



[BS]、[TAB]キーによりファイルを選択し、[↵]キーで上部(選択ファイル名表示エリア)に表示されます。また、ファイル名を直接キーボードから入力することも可能です。選択ファイル名表示エリアにファイル名を表示させ、[↵]キーを押すとそのファイル名をセーブのファイル名として処理します。[ESC]キーにより処理を中断し、ウィンドウを閉じます。

関連命令

LDVAL

記述例

```
SVVAL "TEST"
```

SW( )

オンラインコマンド　ステートメント

概要

I/Oの状態を1ビット単位で返します。

書式

SW(〈I/O番号〉)

説明

〈I/O番号〉で指定した外部入力I/O、外部出力I/O、メモリI/O、リンクメモリI/OのON/OFF状態を返します。ONのときは「1」を、OFFの時は「0」を返します。
〈I/O番号〉は、以下のとおり指定します。

外部入力 I/O	X_n	n=&M0～&M127F(0～2047)
外部出力 I/O	Y_n	n=&M0～&M127F(0～2047)
メモリ I/O	R_n	n=&M0～&M97F (0～1567)
リンクメモリ I/O	L_n	n=&M0～&M127F(0～2047)

記述例

```
A=SW(X_&M0)
```

用例

0番ポートの状態を調べ、ONの時は32番ポートに「1」を出力し、OFFの時は「0」を出力するプログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 A=SW(X_0) ……0番ポートの状態を調べる
130 IF A=1 THEN ON Y_32 ……ONなら32番ポートをONにする
140 IF A=0 THEN OFF Y_32 …OFFなら32番ポートをOFFにる
150 GOTO 120
160 FEND
```

SWAP

ステートメント

概要

2つの変数の値を入れ替えます。

書式

SWAP␣〈変数〉,〈変数〉

説明

2つの変数の値を交換できます。ただし、2つの変数の型が一致していないとエラーになります。

記述例

```
SWAP A,B
```

用例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 INTEGER B
130 A=1
140 B=20
150 WHILE A<11
160 PRINTA,B
170 INC A
180 DEC B
190 WEND
200 SWAP A,B ……変数Aと変数Bの値を入れ替えます
210 PRINT "SWAP",A,B
220 FEND
```

TAN( )

オンラインコマンド　ステートメント

概要

数値の正接(タンジェント)を返します。

書式

TAN(〈数式〉)

説明

〈数式〉の値に対する正接(タンジェント)の値を返します。
〈数式〉の単位はラジアンです。
角度が度で与えられている場合は、ラジアンに変換するためにπ／180を掛けます。
〈数式〉の型に関係なく単精度の値を返します。
(π=3.14159265358979323846…)

関連命令

ATN(),COS(),SIN()

用例

```
>PRINT TAN(3.1415926/180*45)
1.
>
```

TEST

オンラインコマンド

概要

テストモードの設定と内容の読み出しをします。

書式

TEST␣[〈モード番号〉]

説明

テストRUN時のモードを、〈モード番号〉で設定します。
〈モード番号〉を省略すると、現在のモードの設定内容を画面に表示します。
BASICタイプCPU本体の「INITIALIZE/TEST」スイッチを「TEST」側に倒すことで、テストRUN状態になります。
〈モード番号〉には、以下の数値を設定します。

0:シミュレーションモード
(ポートへの実出力をせずにプログラムを実行します)
1:ポーズモード
(プログラム実行時に「PAUSE」命令を有効にします)
2:シミュレーション
(上記の2つを合わせたモードです)

関連命令

CONT,PAUSE,STEP

記述例

```
TEST 1
```

用例

現在設定されているテストランモードを調べます

```
>TEST
0 ……シミュレーションモードに設定されている
>
```

テストランモードを「ポーズモード」に設定します

```
>TEST 1
>
```

TIME

オンラインコマンド　ステートメント

概要

タイマ経過値を設定します。

書式

TIME␣〈!タスク番号〉,〈設定値〉

説明

指定した〈タスク番号〉に設定されているタイマの経過値を〈設定値〉にします。
タイマの経過値を任意に設定することにより、WAIT命令等の動作確認などのプログラムの調整を容易にします。
なお、タイマは各タスクに1つずつ使用できます。

関連命令

TIME()

記述例

```
TIME !1,200
```

用例

タスク番号1のタイマの経過値を200秒に設定する場合

```
>TIME !1,200  ……タイマ経過値を200秒に設定する
>PRINT TIME(!1)  ……タイマ経過値を確認する
200……200秒に設定されている
>
```

TIME( )

オンラインコマンド　ステートメント

概要

タイマ経過値を返します(秒単位)

書式

TIME(〈!タスク番号〉)

説明

指定した〈タスク番号〉に設定されているタイマの経過値を秒単位で返します。

関連命令

TIME

記述例

```
PRINT TIME(!1)
```

用例

```
>PRINT TIME(!1)
200……200秒に設定されている
>
```

TIME$

オンラインコマンド　ステートメント

概要

時刻を設定します。

書式

TIME$␣"〈時〉:〈分〉:〈秒〉"

説明

〈時〉は24時間表記で指定します。また、その時間が1桁の場合、先頭に0を付けて指定します。

　8時=08

〈分〉が1桁の場合、先頭に0を付けて指定します。

　9分=09

〈秒〉が1桁の場合、先頭に0を付けて指定します。

　6秒=06

午前8時9分6秒の場合はTIME$ "08:09:06"となります。

関連命令

DATE$

記述例

```
TIME$"18:37:00"
```

TIME$( )

オンラインコマンド　ステートメント

概要

時刻を返します。

書式

TIME$()

説明

現在時刻を返します。

返される時刻は"時:分:秒"です（午前8時9分6秒の場合は、"08:09:06"が返ります）。

関連命令

DATE$()

記述例

```
PRINT TIME$()
```

TIMER

ステートメント

概要

システム用タイマの経過値を初期化します。

書式

TIMER

説明

システム用タイマの経過値を初期化します。
BASICタイプCPUには、通常のタイマ(WAIT命令)以外に、各タスクごとにシステム用のリアルタイムタイマが用意されています(単位:秒,タイマ値:0～655.35)。
TIMER命令は、このタイマの初期化命令です。

関連命令

TIMER()

記述例

TIMER

用例

```
100 FUNCTION MAIN
110 INTEGER A
120 TIMER ……システムタイマを初期化する(0に戻す)
130 FOR A=1 TO 10000
140 ON Y_&M10
150 NEXT
160 PRINT TIMER() ……命令実行の所要時間を表示する
170 END
180 FEND
```

TIMER( )

ステートメント

概要

システム用タイマの経過値を返します。(10ms単位)

書式

TIMER()

説明

システム用タイマの経過値を返します。
BASICタイプCPUには、通常のタイマ(WAIT命令)以外に、各タスクごとにシステム用のリアルタイムタイマが用意されています(単位:秒,タイマ値:0～655.35)。
TIMER()関数は、このシステム用タイマの経過値を得るための関数です。
10msec単位で経過値が得られ、655.35秒の次は0に戻ります。
TIMER命令をセットでタスクごとの時間管理等に使用します。

関連命令

TIMER

記述例

```
>PRINT TIMER()
23.50
>
```

用例

```
'命令実行時間の測定
100 FUNCTION SAMPLE
110 INTEGER I
120 REAL A,B
130 TIMER ……システムタイマを初期化
140 FOR I=0 TO 999
150 '
160 NEXT I
170 A=TIMER() ……FOR～NEXTﾙｰﾌﾟ1000回の処
180 TIMER 理時間を求める
190 FOR I=0 TO 999
200 ON Y_&M10;OFF Y_&M10 ……FOR～NEXTﾙｰﾌﾟの中で調べ
210 NEXT I たい命令を実行する
220 B=TIMER()
230 PRINT B-A,"msec" ……命令の実行時間を表示する
240 FEND
```

TMOUT

オンラインコマンド　ステートメント

概要

I/O待ちWAIT文のタイムアウト時間を定義します。

書式

TMOUT␣〈数式〉

説明

I/O待ちWAIT文のタイムアウトエラーを〈数式〉で指定します。
〈数式〉は、秒単位で指定します（指定範囲：0～32,765）。

関連命令

WAIT

記述例

```
TMOUT 20
```

用例

「WAIT」命令によりエラー（タイムアウト）が発生したときに、画面に「"TIME OUT ERROR"」の文字列を表示する場合

```
100 FUNCTION SAMPLE
110 TMOUT 20 ……待ち時間を「20」秒に設定する
120 ONERR ER_SUB ……エラー処理ルーチンの宣言
 :
 :
190 WAIT SW(X_0)=1 ……「WAIT」命令でX_0がOFFのまま「20」秒待つ
 :
 :
300 ER_SUB: ……「20」秒以上待った場合はこの行に来る
310 IF ERR(0)〈〉1004 THEN END ……タイムアウトエラーならば終了する
320 PRINT "TIME OUT ERROR" ……タイムアウトエラーならばメッセージを表示する
330 ECLR ……エラーを解除して
340 RETURN ……元のプログラムに復帰する
 :
 :
400 FEND
```

TOFF

オンラインコマンド

概要

TONコマンドで設定したトレースモードを解除します。

書式

TOFF␣[!〈タスク番号〉]

説明

「TON」コマンドで設定したトレースモードを解除します。
〈タスク番号〉を省略すると、すべてのタスクのトレースモードを解除します。

関連命令

TON

記述例

```
TOFF !2
```

用例

```
>TON !1 タスクをトレースする
>XQT
>[1:00010] ……タスク番号と実行中の行番号を表示
>[1:00020]
>[1:00030]
>[1:00040]
>[1:00050]
 :
 :
>TOFF ……トレースモードを解除する
>
```

TON

オンラインコマンド

概要

実行中の行番号を表示します。(トレースモード)

書式

TON␣ [!〈タスク番号〉]

説明

〈タスク番号〉で指定したタスクが現在実行中の行番号を、画面に表示するモード(トレースモード)を設定します。
〈タスク番号〉を省略すると、タスク1に対してトレースモードが有効になります。
トレース中に、さらに他のタスクをトレースモードに設定することができます。

注意

QUITを実行するとトレースも終了します。

関連命令

TOFF

記述例

TON !1

用例

```
>TON !1 タスクをトレースする
>XQT
>[1:00010]  ····· タスク番号と実行中の行番号を表示
>[1:00020]
>[1:00030]
>[1:00040]
>[1:00050]
 :
 :
>TOFF  ····· トレースモードを解除する
>
```

TSTAT

オンラインコマンド

概要

タスクごとのステータスを表示します。

書式

TSTAT

説明

各タスクのステータス(動作状態)を表示します。
表示されるステータスは、以下のとおりです。

quit	初期状態
wait	入力待ち
run	実行中
halt	一時停止
pause	一時停止

関連命令

MONTS

記述例

TSTAT

用例

```
>TSTAT
task1 quit    00010
task2 wait    00500
task3 run     01000
task4 halt    01500
 :
 :
task14 quit   02000
task15 quit   02500
task16 quit   03000
>
```

UNIT( )

オンラインコマンド　ステートメント

概要

複数の数値の下位4ビットを結合します。

書式

UNIT(〈数式〉,[〈数式〉],[〈数式〉],[〈数式〉])

説明

指定した〈数式〉の各々の下位4ビットを結合し、2バイトの整数として返します。
対象となる〈数式〉の種類は、整数・実数とその変数・関数です。
〈数式〉に実数が指定された場合は、小数点以下が四捨五入され、2バイト整数として処理されます。
〈数式〉が省略された場合は、"0"が指定されたものとして処理します。
〈例〉A=&H1001,B=&H1002,C=&H2003,D=&H3004の場合、E=UNIT (A,B,C,D) を実行すると、Eは&H4321となります。
同様にF=UNIT (A,B) を実行すると、Fは&H21となります。

用例

外部入力I/O WX_0～WX_4の下位4ビットを合成し外部出力I/O WY_2に出力するプログラム例

```
100 FUNCTION SAMPLE
110 INTEGER A,B,C,D,E
120 A=INW(WX_0)
130 B=INW(WX_1)
140 C=INW(WX_3)
150 D=INW(WX_4)
160 E=UNIT(A,B,C,D)
170 OUTW WY_2,E
180 FEND
```

UPLD

オンラインコマンド

概要

オブジェクトプログラムをCPUからパソコンのプログラムメモリに転送します。

書式

UPLD␣["[〈ドライブ番号〉:][〈パス名〉]〈ファイル名〉"]

説明

BASICタイプCPUのメモリ内のオブジェクトファイルとシンボリックファイルをパソコンのプログラムメモリ上に読み出し、逆コンパイル（ソースファイルに復元）した後、〈ファイル名〉で指定したソースファイルをディスクに書き込みます。

得られるファイルは、拡張子［＊.PRM］を付けたソースファイルです。

シンボリックファイル（拡張子が「.SYM」）がBASICタイプCPUのプログラムメモリ上にないときは警告メッセージを表示したあと、「変数名」をFP-BASICが自動的に割り付けて、拡張子「＊.BIN」を付けたシンボリックファイルを復元します。復元されたシンボリックファイルでは、「ラベル名」は消去されますが、「ラベル行」は残ります。

〈ドライブ番号〉、〈パス名〉、〈ファイル名〉を省略すると、復元されるソースファイルの〈ファイル名〉が「SMSTMP.PRM」に、復元されるシンボリックファイル名が「SYSTMP.BIN」に設定されます。

注意

逆コンパイルの際に数値が、10進数に変換されることがあります。また、変数宣言がメインプログラムの最初にないと正確に逆コンパイルできません。

関連命令

DWNLD

記述例

```
>UPLD "C:¥TEST"
```

用例

FP3H-BASICタイプ本体からファイルをアップロードし「TEST.PRG」ファイルを生成する場合

```
>UPLD "TEST"
>
```

```
>NEW ……プログラムを消去する
>LIST ……表示されても何もでない（プログラムは消えている）
```

FP3H-BASICタイプ本体からファイルをアップロードし「SYSTMP.PRG」ファイルを生成する場合

```
>UPLD ……アップロード
>LIST ……表示させてみる
100 FUNCTION SAMPLE
110 INTEGER A
120
・
・
・
```

VAL( )

オンラインコマンド　ステートメント

概要

文字列を数値に変換します。

書式

VAL(〈文字列〉)

説明

〈文字列〉を文字型から数値型に変換して返します。

注意

〈文字列〉で指定した文字列中の空白（スペース）は無視します。

関連命令

STR$()

記述例

I=VAL("123")

用例

文字「"123"」を数値「123」に変換する場合

```
>PRINT VAL("123")
123
>
```

文字「"012"」の左から2文字目と数値「4」を（算術演算）する場合

```
>PRINT LEFT$("012",2)+4
!!ERROR 24 ……結果は「5」を期待しているがパラメータタイプ
>                  エラーが発生する
```

↓

数値型にしてから加算する

```
>PRINT VAL(LEFT$("012",2))+4
5
>
```

プログラミング例

```
100 FUNCTION SAMPLE
110 INTEGER A
120 STRING B$,C$
130 B$="012"  ……文字列「012」の
140 C$=LEFT$(B$,2) ……左から2文字目に
150 A=VAL(C$)+4 ……数値の「4」を加算した結果を
160 PRINT A ……画面に表示する
170 FEND
```

VER

オンラインコマンド

概要

BASICタイプCPUのバージョン名を表示します。

書式

VER

説明

パソコンに接続されているBASICタイプCPUのバージョンを画面に表示します。

注意

ROM運転時には、ROM化したプログラムのファイル名を、「File Name」の後に表示します。

記述例

VER

用例

```
>VER
FP3 BASIC TYPE Version 2.2(H) ……命令を入力するとバージョンが
 Program size(K) : 0/128          表示される
Variable  size(K) : 0/41
>
```

VERINIT

オンラインコマンド

概要

BASICタイプCPUのすべての状態を初期化します。

書式

VERINIT

説明

接続されているBASICタイプCPUのメモリを初期化します。パラメータメモリ、スロット割り付け、プログラム、変数、およびパラメータメモリで設定されたI/O、メモリの保持・非保持の属性も初期化されますので注意が必要です。

関連命令

CLRVAL,CLRVAR

記述例

VERINT

用例

```
>PRINT PRM(7) ……メモリI/O保持エリアの内容を画面表示する
60 ……「60(ワード)」に設定されている
>PRM 7,98 ……「98(ワード)」に設定する
>PRINT PRM(7) ……メモリI/O保持エリアの内容を画面表示する
98 ……「98(ワード)」に変更されている
>VERINIT ……すべての状態を初期化します
>PRINT PRM(7) ……メモリI/O保持エリアの内容を画面表示する
60 ……初期設定にもどっている
>
```

WAIT

オンラインコマンド　ステートメント

概要

プログラムを一時的に停止状態にします。

書式

(1)WAIT␣〈数式〉
(2)WAIT␣SW␣(〈I/O番号〉)=〈数式〉
(3)WAIT␣SW␣(〈I/O番号〉)=〈数式〉␣{AND | OR}
␣SW(〈I/O番号〉)=〈数式〉

説明

(1) では、〈数式〉の値だけプログラムを一時停止します。〈数式〉は、秒単位で指定します (指定範囲:0.01～32765)。
(2) では、〈I/O番号〉で指定したI/Oの状態が〈数式〉と一致するまで、プログラムを一時停止します。(3) の場合では、論理式を使用して、複数のI/Oの状態が定められた条件になるまで、プログラムを一時停止することもできます。
〈I/O番号〉は、以下のとおり指定します。

外部入力 I/O	X_n	n=&M0～&M127F(0～2047)
外部出力 I/O	Y_n	n=&M0～&M127F(0～2047)
メモリ I/O	R_n	n=&M0～&M97F (0～1567)
リンクメモリ I/O	L_n	n=&M0～&M127F(0～2047)

注意

1.「書式 (2)」は、WAIT〈条件式〉に一般化することができます。この場合、〈条件式〉の値が真になるまでプログラムを一時停止します。
2.WAIT文のタイムアウトエラーは、TMOUTで定義できます。
3.WAIT I/Oの同時にWAIT可能なI/O数はFP3H BASICは64個まで、FP1 BASICは24個までです。

関連命令

TMOUT

記述例

```
WAIT 3
WAIT SW(X_&M0)
WAIT SW(X_&M0)=1
```

用例

ポート「WY_2」に1秒間だけ「&HFF」を出力する場合

```
:
:
140 OUTW WY_2,&HFF ……WY_2にFFを出力する
150 WAIT 1 ……次の消灯命令を実行する迄の1秒間だけ
160 OUTW WY_2,0 ……出力する
170 GOTO 140 ……消灯の命令"0"
:
:
```

入力ポート「X_0」と「X_1」が共に「ON」になるまで待つ場合

```
:
:
140 WAIT SW(X_0)=1 AND SW(X_1)=1
(「140 WAIT SW(X_0) AND SW(X_1)」と記述することもできます。)
:
:
```

WHILE～WEND

ステートメント

概要

条件式が満たされている間、命令文を繰り返し実行します。
(WHILE条件ループ)

書式

WHILE␣〈条件式〉～WEND

説明

〈条件式〉が真の時、WHILE文とWEND文の間にある文の実行を繰り返します。
〈条件式〉が偽になると、WEND文に続く文に実行が移ります。

注意

「WHILE～WEND」のネスティングは最高10段まで可能です。

関連命令

DO～LOOP

用例

条件式 I<10の時、この間のプログラムをくり返し実行 します。

```
100 FUNCTION SAMPLE
110 INTEGER I
120 I=1
130 WHILE I<10
140 OUTW WY_0,I
150 WAIT 1
170 INC I
180 WEND
190 END
200 FEND
```

WRITE%

オンラインコマンド　ステートメント

概要

I/Oまたはメモリから、高機能ユニットの共有メモリにデータを書き込みます。

書式

WRITE%␣〈スロット番号〉,〈CPUユニットのアドレス〉,〈ワード数〉,〈高機能ユニットのアドレス〉

説明

CPUユニットのI/O、メモリから、〈ワード数〉分のデータを読み出し、〈スロット番号〉で指定された高機能ユニットの共有メモリへ書き込みます。
〈相手局のスロット番号〉には、0～31を指定します。
〈CPUユニットのアドレス〉は、下表の範囲でI/Oまたはメモリの先頭アドレスを指定します。

外部入力 I/O	WX_n	n=0～127
外部出力 I/O	WY_n	n=0～127
メモリ I/O	WR_n	n=0～97
リンクメモリ I/O	WL_n	n=0～127
データメモリ	DT_n	n=0～2047
リンクデータメモリ	LD_n	n=0～255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

〈ワード数〉には、書き込むデータのワード数を指定します。
〈高機能ユニットのアドレス〉には、高機能ユニットの共有メモリのワードアドレスを指定します。

注意

WRITE命令は、共有メモリのアドレスをワード単位で指定します。

バイトアドレス	0	1	2	3	4	5	6	7
ワードアドレス	0		1		2		3	

参照

高機能ユニットの共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表を御覧ください。

関連命令

INPUT%,PRINT%,READ%

記述例

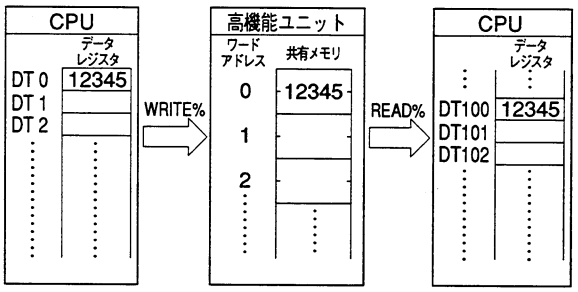
```
WRITE% 4,DT_0,10,0
```

用例

WRITE%　0, DT_100, 2, 0を実行すると、CPUユニットのDT_100の内容をスロット0の高機能ユニットの共有メモリのワードアドレス0, 1へ書き込みます。

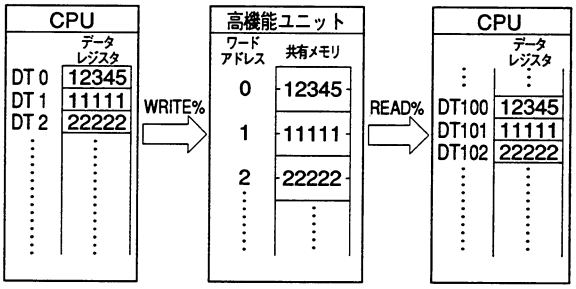
用例

データレジスタの内容を高機能ユニットの共有メモリに対し書き込みを行い、別のデータレジスタに読み出しを行います。




```
100 FUNCTION SAMPLE
110'
120'** WRITE-READ-1 **
130'
140 OUTW DT_0,12345
150 WRITE% 2,DT_0,1,0
160'
170 READ% 2,0,1,DT_100
180 PRINT INW(DT_100)
190'
200 FEND
```

3つのデータレジスタの内容を高機能ユニットの共有メモリに対し書き込み・読み出しを行います。




```
100 FUNCTION SAMPLE
110'
120'** WRITE-READ-2 **
130'
140 OUTW DT_0,12345
150 OUTW DT_1,11111
160 OUTW DT_2,22222
170 WRITE% 2,DT_0,3,0,
180'
190 READ% 2,0,3,DT_100
200 PRINT INW(DT_100)
210 PRINT INW (DT_101)
220 PRINT INW (DT_102)
230'
240 FEND
```

WRITER

オンラインコマンド　ステートメント

概要

I/Oまたはメモリから、リモートI/O子局の高機能ユニットの共有メモリにデータを書き込みます。

書式

WRITER␣〈親局番号〉,〈子局番号〉,〈スロット番号〉,〈CPUユニットアドレス〉,〈ワード数〉,〈高機能ユニットのアドレス〉

説明

CPUユニットのI/Oまたはメモリから、〈ワード数〉分のデータを読み出し、〈親局番号〉,〈子局番号〉,〈スロット番号〉で指定した高機能ユニットの共有メモリに書き込みます。
〈親局番号〉には、1～4を指定します。
〈子局番号〉には、1～32を指定します。
〈スロット番号〉には、0～31を指定します。
〈CPUユニットのアドレス〉には、下表の範囲でI/O、メモリの先頭アドレスを指定します。

外部入力 I/O	WX_n	n=0～127
外部出力 I/O	WY_n	n=0～127
メモリ I/O	WR_n	n=0～97
リンクメモリ I/O	WL_n	n=0～127
データメモリ	DT_n	n=0～2047
リンクデータメモリ	LD_n	n=0～255
ファイルメモリ	FL_n	n=プログラムエリアのサイズによる

〈ワード数〉には、書き込むデータのワード数を指定します。
〈高機能ユニットのアドレス〉には、高機能ユニットの共有メモリのワードアドレスを指定します。

注意

WRITER命令は共有メモリのアドレスをワード単位で指定します。

バイトアドレス	0	1	2	3	4	5	6	7
ワードアドレス	0		1		2		3	

参照

高機能ユニットの共有メモリの割り付け内容については、巻末の高機能ユニットパラメーター覧表を御覧ください。

関連命令

INPUTR,PRINTR,READR

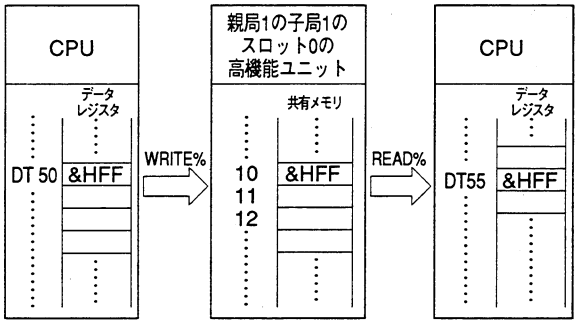
記述例

```
WRITER 1,1,0,DT_0,1,0
```

用例

リモートI/O子局の共有メモリ書き込み・読み出し

データレジスタの内容を親局1の子局1のスロット0の高機能ユニットの共有メモリに対し書き込み別のデータレジスタに読み込みます。




```
100 FUNCTION SAMPLE
110'
120'** WRITE-READ-1 **
130'
140 INTEGER A
150 OUTW DT_50 , &HFF
160 WRITER 1, 1, 0, DT_50, 1, 10,
170'
180 READR 1, 1, 0, 10, 1, DT_55
190 PRINT INW(DT_55)
200 FEND
```

XQT

オンラインコマンド　ステートメント

概要

BASICタイプCPUのプログラムを実行します。

書式

XQT[␣〈タスク番号〉][,〈プログラム名〉][,〈実行開始行番号〉]-[〈実行終了行番号〉]

説明

プログラムを〈タスク番号〉で指定したタスクで実行します。
〈タスク番号〉には1～16の整数を指定し、省略するとタスク1で実行します。
〈プログラム名〉を省略すると、メインプログラム（最初に定義されたプログラム）を実行します。
〈実行開始行番号〉を省略したときは、プログラムの先頭から実行を開始し、〈実行終了行番号〉を省略したときは、プログラムの終了まで実行を続けます。すでに実行中のタスクに対してこの命令を実行するとエラーになります。また、プログラム中でステートメントとして使用すると〈プログラム名〉以降の指定は無効となります。

注意

〈実行終了番号〉の命令は実行しません。

記述例

XQT ……メインプログラムをタスク1で実行する。
XQT !1, MAIN, 1500-2000 ……メインプログラムをタスク1で1500行から1999行まで実行する。
（※行番号2000の命令は実行しません。）

用例

プログラムをタスク1で実行する
>XQT

プログラム「JOB2」をタスク2で実行する
>XQT !2, JOB2

行番号340からのプログラムをタスク1で実行し700行で中止する
>XQT 340-700 ……（※行番号700の命令は実行しません。）

行番号340からのプログラムをタスク2で実行し700行で中止する
>XQT !2, JOB2, 340-700 ……（※行番号700の命令は実行しません。）

3章 資料集

3-1
FP-BASIC 性能仕様

(1)主な機能

項目		項目
総合機能		PCコンパイラ型マルチタスク構造型BASIC 同時に実行できる最大タスク数　FP3、FP3H：最大16タスク、FP1：最大6タスク タスク間グローバル変数/ローカル変数装備
PC動作環境設定機能		CPU初期化 CPUパラメータメモリ設定 スロット割り付け
編集機能	エディタ	スクロール可能フルスクリーンエディタ 編集可能サイズ：最大255バイト/行　最大64Kバイト/ファイル ファイルセーブ/ロード
	コンパイラ	PC実行プログラム作成/転送（分割コンパイル可能） ROMライタファイル作成 実行ファイルサイズ最大128Kバイト（OBJファイルとSYMファイルの合計）＊
PC運転/デバッグ機能		運転/停止　トレースモード　テストモード＊ 運転モード/ステータス表示 I/O、メモリ、変数のモニタ I/O、メモリの状態のダンプ タイマ経過値表示/設定 高機能ユニット割り込み設定状態の表示＊ I/O、メモリ、その他の状態の一括ダンプ表示と編集＊ I/O、メモリ、変数の状態のファイルセーブ・ロード＊ 強制入出力＊
PCターミナル機能		パソコンのディスクドライブアクセス可能＊

注)＊印の機能については使用するPCにより制限される場合があります。

(2)動作環境

項目	項目
適応PC	①FP3H-BASICタイプCPU（AFP3261M） FP-BASICのすべてのコマンド、命令、関数が使用できます ②Ver.2.0以降のFP3-BASICタイプCPU（AFP3251） FP-BASICのすべてのコマンド、命令、関数が使用できます ③Ver.2.0未満のFP3-BASICタイプCPU（AFP3251） 使用できるコマンド、命令、関数が限定されます ④FP1 BASICタイプCPU 使用できるコマンド、命令、関数が限定されます
適応パソコン	EPSON　PC286シリーズ/386/シリーズ/486シリーズ　640×400ドット表示モード NEC　PC9801シリーズ　メモリ640KB実装　640×400ドット表示モード （初代PC9801、PC9801E/F/U2、PC9801XA、PC9801LTは使用できません）
OS	NEC日本語MS-DOS Ver.3.10以降 （CONFIG.SYSにPRINT.SYSの組み込みが必要（RSDRV.SYS）の組み込みは不要）
メモリ容量	Ver.2.3以降は471Kバイト以上、Ver.2.3未満は、520Kバイト以上の空きメモリの確保が必要 （NECDIC.DRV以外のかな漢字変換システムを使用する場合は、EMSメモリが必要です）

3-2
スロット番号とI/O割り付け

(1)自動割り付け
(SLOTCLRコマンドによる割り付け)

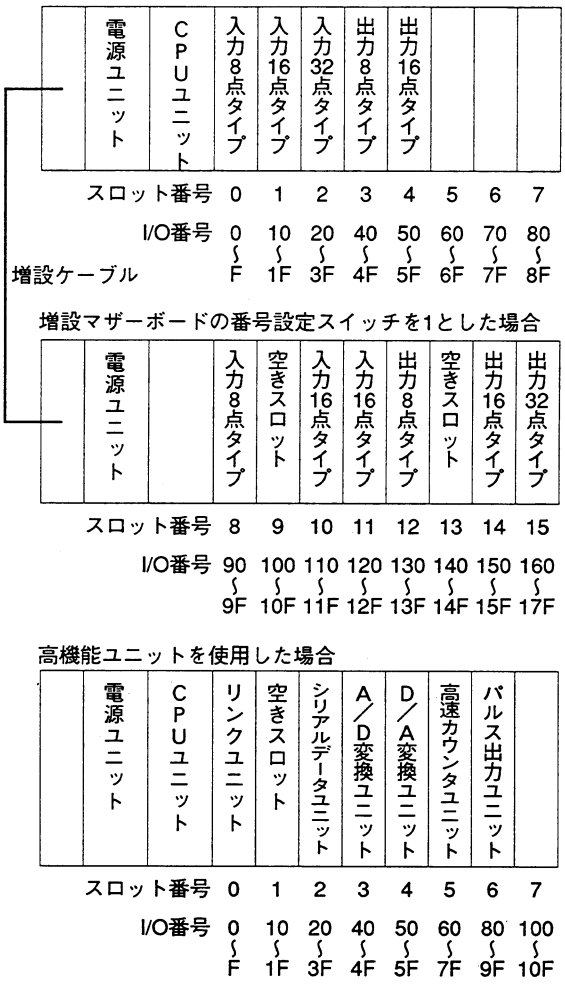
各スロットのスロット番号およびI/O番号は、SLOTCLRコマンドによりマザーボードに装着したユニットの位置で自動的に設定されます。

FP-BASICからSLOTCLRコマンドを実行すると、各スロットのI/Oユニットのタイプを調べ、スロット番号およびI/O番号を自動的に割り付けます。

●SLOTCLRコマンドによるスロット割り付けの内容

1. 8点ユニットには16点分のI/O番号が割り付けられます。
2. 空きスロットには16点分のI/O番号が出力として割り付けられます。
3. リンクユニットなどの実際にI/Oをもたないユニットには、16点分のI/O番号が出力として割り付けられます。ただし、実際の出力はできません。

各ユニットに割り付けられる占有点数についてはP.207の「ユニット占有点数／機能番号一覧」をご覧ください。



(2)任意割り付け
(SLOTコマンドによる割り付け)

FP-BASICのSLOTコマンドを使用して、スロット番号およびI/O番号を自由に割り付けることができます(任意割り付け)。また、事前にSLOTCLRコマンドで自動割り付けした場合は、SLOTコマンドで再割り付けしたスロット番号およびI/O番号以降の割り付け情報はシフトされます。

●SLOTコマンドによるスロット割り付けの内容

1. 各スロットに装着されるユニットの種類を、I/Oユニット、高機能ユニットの中から選択します。
2. 各スロットのI/O点数は、0点、8・16点、32点、64点の中から選択します。リンクユニットなどのI/Oを持たないユニットには、0点を選択することでI/O番号の空きをなくすことができます。
3. I/O点数は、正しく設定してください。I/O点数を実際に装着されているユニットの点数よりも少なく設定すると、そのユニットで使用できるI/O点数が減少します。

注意

・ユニット交換時には、SLOTCLRコマンドまたはSLOTコマンドを使用して必ずスロット割り付けをしてください。スロット割り付けを行わないと正常な動作が得られないことがあります。

●設定の手順

SLOT 〈スロット番号〉,〈機能番号〉
①　②

①スロット番号は、10進数で0～23の範囲で指定してください。

②機能番号は、3桁の10進数で指定します。
指定内容はそれぞれ100桁の位はユニットの種類、10桁の位は入力点数、1桁の位は出力点数を指定します。

桁	属性	ユニットの種類	表記
100の桁	I/Oの種類	I/Oユニット	1
		高機能ユニット	2
		空ユニット	3
		リンクユニット	4
10の桁	入力点数	0点	0
		8,16点	2
		32点	3
		64点	4
		128点	5
1の桁	出力点数	0点	0
		8,16点	2
		32点	3
		64点	4
		128点	5

〈例1〉スロット番号2番を16点出力ユニットに設定する場合。
SLOT 2,120 と記述します。

〈例2〉スロット番号3番を32点出力ユニットに設定する場合。
SLOT3,103 と記述します。

注意

この定義文はプログラムの先頭付近に記述してください。

(3)ユニット-占有点数/機能番号一覧

ユニットの種類		ご注文品番	入力点数	出力点数	自動割り付け時の占有点数(SLOTCLRコマンド実行時)	任意割り付け時にSLOTコマンドで指定する機能番号
空きスロット		———	0点	(16点) *1	16点 *1	302(300) *2
入力ユニット	8点タイプ	AFP33041	16点	0点	16点	120
		AFP44051				
	16点タイプ	AFP33023	16点	0点	16点	120
		AFP33043				
		AFP33053				
	32点タイプ	AFP33024	32点	0点	32点	130
		AFP33014				
	64点タイプ	AFP33027	64点	0点	64点	140
		AFP33017				
出力ユニット	16点タイプ	AFP33103	0点	16点	16点	102
		AFP33203				
		AFP33483				
		AFP33583				
		AFP33703				
	32点タイプ	AFP33484	0点	32点	32点	103
		AFP33584				
	64点タイプ	AFP33487	0点	64点	64点	104
		AFP33587				
A/D変換ユニット	4ch非絶縁タイプ	AFP3400	16点	0点	16点	220
	8ch非絶縁タイプ	AFP3402	0点	(16点) *1	16点 *1	202(200) *2
		AFP3403				
		AFP3405				
	8ch絶縁タイプ	AFP3406				
		AFP3407				
		AFP3408				
D/A変換ユニット	2ch非絶縁タイプ	AFP3410	16点	0点	16点	220
		AFP3411				
	2ch絶縁タイプ	AFP3412	0点	(16点) *1	16点 *1	202(200) *2
		AFP3413				
		AFP3416				
		AFP3417				
	4ch絶縁タイプ	AFP3414				
		AFP3415				
		AFP3418				
		AFP3419				
熱電対入力ユニット		AFP3420	16点	0点	16点	220
測温抵抗体入力ユニット		AFP3421	16点	0点	16点	220
シリアルデータユニット		AFP3460	16点	16点	32点	222
データプロセスユニット		AFP3461	16点	16点	32点	222
FD1コントロールユニット		AFP3471	16点	16点	32点	222
バーコードリーダI/Fユニット		AFP34601	0点	(16点) *1	16点*1	202(200) *2
高速カウンタユニット		AFP3621	16点	16点	32点	222
		AFP3622				
パルス出力ユニット		AFP3480	16点	16点	32点	222
位置決めユニットEタイプ	1軸タイプ	AFP3431E	16点	16点	32点	222
	2軸タイプ	AFP3432E	32点	32点	64点	233
位置決めユニットFタイプ(トランジスタ出力型)	1軸タイプ	AFP3431	16点	16点	32点	222
	2軸タイプ	AFP3432	32点	32点	64点	233
位置決めユニットFタイプ(ラインドライバ出力型)	1軸タイプ	AFP3434	16点	16点	32点	222
	2軸タイプ	AFP3435	32点	32点	64点	233
	3軸タイプ	AFP3436				
割り込みユニット		AFP3452	16点	0点	16点	220
MEWNET-TRトランスミッタマスタユニット		AFP3750	0,32,64,128点	0,32,64,128点	*3	4□□ *3
MEWNET-FリモートI/Oマスタユニット		AFP3470	0点	(16点) *1	16点*1	402(400) *2
		AFP3472				
MEWNET-Wリンクユニット		AFP3720	0点	(16点) *1	16点*1	402(400) *2
MEWNET-Pリンクユニット		AFP3710	0点	(16点) *1	16点*1	402(400) *2
MEWNET-Hリンクユニット		AFP3700	0点	(16点) *1	16点*1	402(400) *2
C-NETリンクユニット		AFP3463	0点	(16点) *1	16点*1	402(400) *2
コンピュータコミュニケーションユニット		AFP3462	0点	(16点) *1	16点*1	402(400) *2
ESBリンクユニット		AFP3770	0点	(16点) *1	16点*1	402(400) *2
		AFP3771				
拡張データメモリユニット		AFP32091	16点	0点	16点	220
		AFP32092				

注) *1 SLOTCLRコマンド実行時,リンクユニットなど実際にI/Oをもたないユニットには,16点分のI/O番号が出力として割り付けられます。ただし,実際の出力はできません。
*2 SLOTコマンドを使って,上記の( )内の機能番号を指定すると占有点数を0にすることができます。
*3 MEWNET-TRトランスミッタマスタユニットは,ディップスイッチの設定によりI/O点数が変わります。使用点数に合わせて,機能番号を設定してください。設定方法については,前ページの"設定の手順"の項をご覧ください。

3-3
I/O・メモリ一覧

■I/O・メモリ構成一覧

BASICタイプCPUのI/O、メモリの構成一覧です。なお、1ワードは16点（16ビット）です。

名称	点数/サイズ	アドレス指定	機能	
外部入力 （X_） （WX_）	2048点（128ワード）	0～2047 &M0～&M127F	入力ユニットに対応するI/Oです。	*3
外部出力 （Y_） （WY_）	2048点（128ワード）	0～2047 &M0～&M127F	出力ユニットに対応するI/Oです。 外部出力ユニットで使用しない場合は、 メモリI/Oとして使用できます。	*1 *2 *3
メモリI/O （R_） （WR_）	1568点（98ワード）	0～1567 &M0～&M97F	CPU内部だけ使用できるI/Oです。	*1 *3
特殊メモリI/O （R_） （WR_）	176点（11ワード）	14400～14575 &M9000～&M910F	あらかじめ用途が決まっているメモリI/Oです。読み出し専用です。	*3
リンクメモリI/O （L_） （WL_）	2048点（128ワード）	0～2047 &M0～&M127F	PCリンク使用時のデータ受け渡し用のメモリI/Oです。 リンク用に使用しない部分は、メモリI/Oとして使用できます。	*1 *2 *3
データメモリ （DT_）	2048ワード	0～2047	CPU内部で使用できるメモリです。	*1
特殊データメモリ （DT_）	256ワード	9000～9255	あらかじめ用途が決まっているデータメモリです。 読み出し専用です。	
リンクデータメモリ （LD_）	256ワード	0～255	PCリンク使用時のデータ受け渡し用のメモリI/Oです。	*1 *2
ファイルメモリ	*4	*4	ユーザプログラムエリアの未使用領域を利用する拡張メモリです。 データメモリとして使用します。	

注）*1:パラメータメモリにより、保持/非保持の設定ができます。
*2:使用しないY_、L_はRとして、LD_はD_としてそれぞれ使用できます。
*3:X_、Y_、R_、L_のビット単位でのアドレス指定で、&M表記（ミュー表記）を使用する場合、最下位の桁は16進表記、それ以外の桁は10進表記です。したがってワード単位のアドレス指定では、&M表記の最下位桁を除いた数値を指定します。
*4:ユーザープログラム領域に設定値により異なります。

3-4
パラメータメモリー覧

■プログラム容量の設定

No.	説明	デフォルト値	表示設定範囲の説明
0	ユーザプログラム容量の設定	(FP3) 64 (FP3H)128	(FP3) 16～64Kバイト *1 (FP3H) 16～128Kバイト *2

■タスク実行時間の設定

No.	説明	デフォルト値	表示設定範囲の説明
1	ユーザタスクのタイムスライスのベース時間設定	2	0～2 0…0.5msec. 1…1msec. 2…2msec.
2	MEWNET-Hタスクへの割り当て時間の設定	1	1～32 (設定値)×2msec.
39	ユーザータスクへの割り当て時間の設定	1	1～8 (設定値)×ベース時間

■I/O、メモリの保持・非保持の属性の設定

No.	説明	デフォルト値	表示設定範囲の説明
7	メモリI/Oの保持エリアの設定	60	0～98（ワード単位で設定可） *2
8	データメモリの保持エリアの設定	0	0～2048（1点単位で設定可） *2
9	ファイルメモリの保持エリアの設定	0	(FP3) 0～24576ワード *2 (FP3H) 0～57344ワード *2
10	PCリンク0用リンクメモリI/O保持エリアの設定	0	0～64（ワード単位で設定可） *2
11	PCリンク1用リンクメモリI/O保持エリアの設定	64	64～128（ワード単位で設定可） *2
12	PCリンク0用リンクデータメモリ保持エリアの設定	0	0～128（1点単位で設定可） *2
13	PCリンク1用リンクデータメモリ保持エリアの設定	128	128～256（1点単位で設定可） *2
15	出力の保持・非保持の設定	1	0：非保持 1：保持 *2
16	プログラムモードでの出力の保持・非保持の設定	1	0：非保持 1：保持 *2
19	メモリの保持・非保持の設定方向設定値以降を保持	1	0：設定値未満を保持 *2 1：設定値以上を保持 *2

■異常時の運転モード

No.	説明	デフォルト値	表示設定範囲の説明
21	出力ヒューズ断時	0	0：停止 1：運転
22	特殊ユニット異常時	0	0：停止 1：運転
23	I/O照合の異常時	0	0：停止 1：運転
26	エラーの発生時の処置	0	0：停止 1：運転
27	リモートI/O交信異常時の処置	0	0：停止 1：運転
28	リモート子局のユニット異常時の処置	0	0：停止 1：運転
32	MEWNET SEND/RECV命令待ち時間	800	4～32760（10msec～81.9sec） (2.5msec×設定値)
35	リモートI/O子局接続待ちモード設定	1	0：解除（接続を待たずに運転開始） 1：設定（接続を待って運転開始）

■PCリンク0のリンクエリアの割り付け

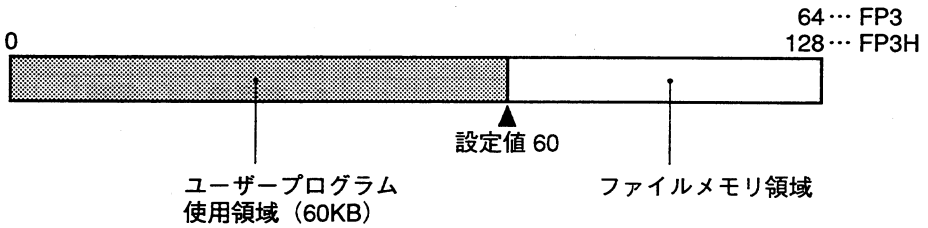
No.	説明	デフォルト値	表示設定範囲の説明	
40	リンクメモリI/Oリンク範囲の設定	0	0～64ワード	*3
41	リンクデータメモリリンク範囲の設定	0	0～128ワード	*3
42	リンクメモリ送信開始ワードNo.の設定	0	0～63	
43	リンクメモリI/O送信ワード数の設定	0	0～64	
44	リンクデータメモリ送信開始ワードNo.の設定	0	0～127	
45	リンクデータメモリ送信ワード数の設定	0	0～127	

■PCリンク1のリンクエリアの割り付け

No.	説明	デフォルト値	表示設定範囲の説明	
50	リンクメモリI/Oリンク範囲の設定	0	0～64ワード	*3
51	リンクデータメモリリンク範囲の設定	0	0～128ワード	*3
52	リンクメモリ送信開始ワードNo.の設定	64	64～127	
53	リンクメモリI/O送信ワード数の設定	0	0～64	
54	リンクデータメモリ送信開始ワードNo.の設定	128	128～255	
55	リンクデータメモリ送信ワード数の設定	0	0～127	

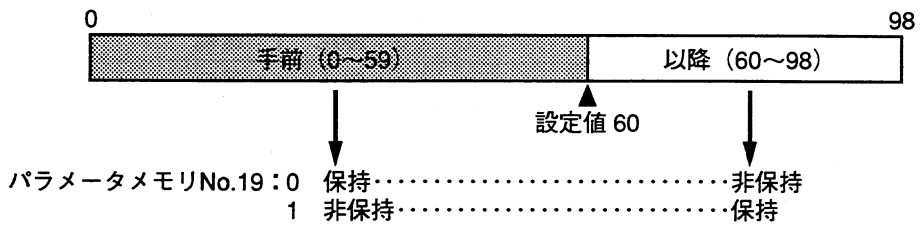
*1:設定した使用領域以外は、ファイルメモリ(FL)として使用されます。

例



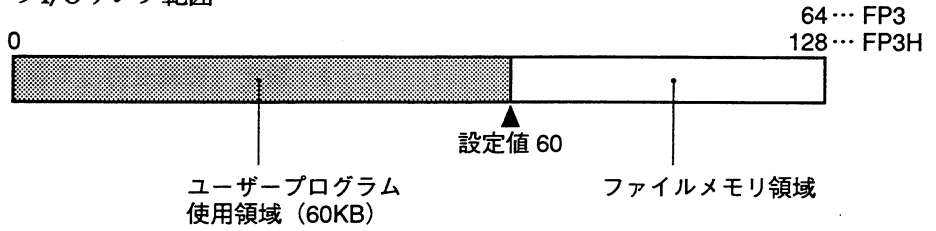
*2:指定値の手前までを保持にするか、指定値以降を保持にするかを選択できます。
パラメーターメモリNo.19の値で設定します。

例7.メモリI/Oの保持エリア



*3:実際に設定されるリンクエリア範囲は、設定範囲の小さい方から設定値までがリンクエリアとなります。

例40.リンクI/Oリンク範囲



3-5
特殊メモリI/O一覧

BASICタイプCPUの内部で、あらかじめ用途が決まっているメモリI/Oです。読み出し専用なので、ON、OFF、OUT命令の対象にはなりません。SW、IN命令でのみ使用できます。

ワードNo.	番号	名称	内容	備考
900	R_14400 R_&M9000	自己診断エラー	正常時：0 検出時：1	自己診断の結果はDT_9000に格納します。 同コードはERRに反映します。
	R_14401 R_&M9001	瞬停検知	正常時：0 検出時：1	DT_9001に検出した瞬停回数を格納します。
	R_14402 R_&M9002	ヒューズ断検知	正常時：0 検出時：1	DT_9002,9003に検出したスロット番号を格納します。
	R_14403 R_&M9003	特殊ユニット異常	正常時：0 検出時：1	DT_9006,9007に検出したスロット番号を格納します。
	R_14404 R_&M9004	I/O照合異常	正常時：0 検出時：1	DT_9010,9011に検出したスロット番号を格納します。
	R_14405 R_&M9005	電池異常	正常時：0 検出時：1	
	R_14406 R_&M9006	電池異常保持	正常時：0 検出時：1	1度電池異常を検出すると復帰後も保持します。 電源OFFかイニシャライズSW操作で0にすることができます。
	R_14407 R_&M9007	演算異常	正常時：0 検出時：1	発生した行番号をDT_9017に格納します。
	R_14408 R_&M9008	リモートI/Oリードライトエラー	正常時：0 検出時：1	コントロールデータが指定外のとき。 マスタユニットがないとき。 アドレス修飾エラーが発生しているとき。 書き込みデータが指定範囲を越えているとき。

ワードNo.	番号	名称	内容
901	R_14416 R_&M9010	常時ON	
	R_14417 R_&M9011	常時OFF	
	R_14425 R_&M9018	未使用	
	R_14426 R_&M9019	未使用	
	R_14427 R_&M901A	0.1秒クロックパルス	
	R_14428 R_&M901B	0.2秒クロックパルス	
	R_14429 R_&M901C	1秒クロックパルス	
	R_14430 R_&M901D	2秒クロックパルス	
	R_14431 R_&M901E	1分クロックパルス	
902	R_14432 R_&M9020	RUNモード	PROG.モード時：0 RUNモード時：1
	R_14433 R_&M9021	テストモード	通常モード時：0 テストモード時：1
	R_14434 R_&M9022	ポーズ	通常実行中：0 PAUSE命令またはSTEP命令で一時停止中：1
	R_14435 R_&M9023	ポーズモード	テストRUN時のポーズ命令無効：0 テストRUN時のポーズ命令有効：1
	R_14436 R_&M9024	シミュレーションモード	テストRUN時にも通常出力をする：0 テストRUN時に実出力をしない：1
	R_14439 R_&M9027	リモートモード	RUN←→PROG.の遠隔切り換えが可能なモード：1
	R_14440 R_&M9028	強制モード	強制出力をしていないとき（解除時）：0 I/O、メモリI/Oの強制入出力をしているとき：1
	R_14443 R_&M902B	割り込み異常フラグ	割り込み異常発生時：1
903	R_14448 R_&M9030	MEWNET送受信命令実行可能	実行不可（実行中）：0 実行可：1
	R_14449 R_&M9031	MEWNET送受信命令実行完了	正常終了：0 異常終了：1　異常コードはDT_9039にセットされる。
	R_14453 R_&M9035	メモリアクセス命令実行可能	実行不可（実行中）：0 実行可：1
	R_14454 R_&M9036	メモリアクセス命令実行完了	正常終了：0 異常終了：1
905	R_14480 R_&M9050	MEWNET伝送異常（リンク1）	正常時：0 伝送異常または設定異常：1
	R_14481 R_&M9051	MEWNET伝送異常（リンク2）	正常時：0 伝送異常または設定異常：1
	R_14482 R_&M9052	MEWNET伝送異常（リンク3）	正常時：0 伝送異常または設定異常：1

ワードNo.	番号	名称	内容
906	R_14496 R_&M9060	PCリンク0用伝送保証メモリI/O	ユニットNo. 1の伝送正常時 :ON 伝送異常時または停止時:OFF
	R_14497 R_&M9061		ユニットNo. 2の伝送正常時 :ON 伝送異常時または停止時:OFF
	R_14498 R_&M9062		ユニットNo. 3の伝送正常時 :ON 伝送異常時または停止時:OFF
	R_14499 R_&M9063		ユニットNo. 4の伝送正常時 :ON 伝送異常時または停止時:OFF
	R_14500 R_&M9064		ユニットNo. 5の伝送正常時 :ON 伝送異常時または停止時:OFF
	R_14501 R_&M9065		ユニットNo. 6の伝送正常時 :ON 伝送異常時または停止時:OFF
	R_14502 R_&M9066		ユニットNo. 7の伝送正常時 :ON 伝送異常時または停止時:OFF
	R_14503 R_&M9067		ユニットNo. 8の伝送正常時 :ON 伝送異常時または停止時:OFF
	R_14504 R_&M9068		ユニットNo. 9の伝送正常時 :ON 伝送異常時または停止時:OFF
	R_14505 R_&M9069		ユニットNo.10の伝送正常時 :ON 伝送異常時または停止時:OFF
	R_14506 R_&M906A		ユニットNo.11の伝送正常時 :ON 伝送異常時または停止時:OFF
	R_14507 R_&M906B		ユニットNo.12の伝送正常時 :ON 伝送異常時または停止時:OFF
	R_14508 R_&M906C		ユニットNo.13の伝送正常時 :ON 伝送異常時または停止時:OFF
	R_14509 R_&M906D		ユニットNo.14の伝送正常時 :ON 伝送異常時または停止時:OFF
	R_14510 R_&M906E		ユニットNo.15の伝送正常時 :ON 伝送異常時または停止時:OFF
	R_14511 R_&M906F		ユニットNo.16の伝送正常時 :ON 伝送異常時または停止時:OFF
907	R_14512 R_&M9070	PCリンク0用動作モードメモリI/O	ユニットNo. 1のRUN時 :ON PROG.時:OFF
	R_14513 R_&M9071		ユニットNo. 2のRUN時 :ON PROG.時:OFF
	R_14514 R_&M9072		ユニットNo. 3のRUN時 :ON PROG.時:OFF
	R_14515 R_&M9073		ユニットNo. 4のRUN時 :ON PROG.時:OFF
	R_14516 R_&M9074		ユニットNo. 5のRUN時 :ON PROG.時:OFF
	R_14517 R_&M9075		ユニットNo. 6のRUN時 :ON PROG.時:OFF
	R_14518 R_&M9076		ユニットNo. 7のRUN時 :ON PROG.時:OFF
	R_14519 R_&M9077		ユニットNo. 8のRUN時 :ON PROG.時:OFF
	R_14520 R_&M9078		ユニットNo. 9のRUN時 :ON PROG.時:OFF
	R_14521 R_&M9079		ユニットNo.10のRUN時 :ON PROG.時:OFF
	R_14522 R_&M907A		ユニットNo.11のRUN時 :ON PROG.時:OFF
	R_14523 R_&M907B		ユニットNo.12のRUN時 :ON PROG.時:OFF
	R_14524 R_&M907C		ユニットNo.13のRUN時 :ON PROG.時:OFF
	R_14525 R_&M907D		ユニットNo.14のRUN時 :ON PROG.時:OFF
	R_14526 R_&M907E		ユニットNo.15のRUN時 :ON PROG.時:OFF
	R_14527 R_&M907F		ユニットNo.16のRUN時 :ON PROG.時:OFF

ワードNo.	番号	名称	内容
908	R_14528 R_&M9080	PCリンク1用伝送保証メモリI/O	ユニットNo. 1 の伝送正常時 : ON 伝送異常時または停止時 : OFF
	R_14529 R_&M9081		ユニットNo. 2 の伝送正常時 : ON 伝送異常時または停止時 : OFF
	R_14530 R_&M9082		ユニットNo. 3 の伝送正常時 : ON 伝送異常時または停止時 : OFF
	R_14531 R_&M9083		ユニットNo. 4 の伝送正常時 : ON 伝送異常時または停止時 : OFF
	R_14532 R_&M9084		ユニットNo. 5 の伝送正常時 : ON 伝送異常時または停止時 : OFF
	R_14533 R_&M9085		ユニットNo. 6 の伝送正常時 : ON 伝送異常時または停止時 : OFF
	R_14534 R_&M9086		ユニットNo. 7 の伝送正常時 : ON 伝送異常時または停止時 : OFF
	R_14535 R_&M9087		ユニットNo. 8 の伝送正常時 : ON 伝送異常時または停止時 : OFF
	R_14536 R_&M9088		ユニットNo. 9 の伝送正常時 : ON 伝送異常時または停止時 : OFF
	R_14537 R_&M9089		ユニットNo.10の伝送正常時 : ON 伝送異常時または停止時 : OFF
	R_14538 R_&M908A		ユニットNo.11の伝送正常時 : ON 伝送異常時または停止時 : OFF
	R_14539 R_&M908B		ユニットNo.12の伝送正常時 : ON 伝送異常時または停止時 : OFF
	R_14540 R_&M908C		ユニットNo.13の伝送正常時 : ON 伝送異常時または停止時 : OFF
	R_14541 R_&M908D		ユニットNo.14の伝送正常時 : ON 伝送異常時または停止時 : OFF
	R_14542 R_&M908E		ユニットNo.15の伝送正常時 : ON 伝送異常時または停止時 : OFF
	R_14543 R_&M908F		ユニットNo.16の伝送正常時 : ON 伝送異常時または停止時 : OFF
909	R_14544 R_&M9090	PCリンク1用動作モードメモリI/O	ユニットNo. 1 のRUN時 : ON PROG.時 : OFF
	R_14545 R_&M9091		ユニットNo. 2 のRUN時 : ON PROG.時 : OFF
	R_14546 R_&M9092		ユニットNo. 3 のRUN時 : ON PROG.時 : OFF
	R_14547 R_&M9093		ユニットNo. 4 のRUN時 : ON PROG.時 : OFF
	R_14548 R_&M9094		ユニットNo. 5 のRUN時 : ON PROG.時 : OFF
	R_14549 R_&M9095		ユニットNo. 6 のRUN時 : ON PROG.時 : OFF
	R_14550 R_&M9096		ユニットNo. 7 のRUN時 : ON PROG.時 : OFF
	R_14551 R_&M9097		ユニットNo. 8 のRUN時 : ON PROG.時 : OFF
	R_14552 R_&M9098		ユニットNo. 9 のRUN時 : ON PROG.時 : OFF
	R_14553 R_&M9099		ユニットNo.10のRUN時 : ON PROG.時 : OFF
	R_14554 R_&M909A		ユニットNo.11のRUN時 : ON PROG.時 : OFF
	R_14555 R_&M909B		ユニットNo.12のRUN時 : ON PROG.時 : OFF
	R_14556 R_&M909C		ユニットNo.13のRUN時 : ON PROG.時 : OFF
	R_14557 R_&M909D		ユニットNo.14のRUN時 : ON PROG.時 : OFF
	R_14558 R_&M909E		ユニットNo.15のRUN時 : ON PROG.時 : OFF
	R_14559 R_&M909F		ユニットNo.16のRUN時 : ON PROG.時 : OFF

3-6 特殊データメモリ一覧

BASICタイプCPUの内部で、あらかじめ用途が決まっているメモリI/Oです。このデータメモリは、読み出し専用なので書き込みはできません。

番号	名称	内容		備考
DT_9000	自己診断コード			自己診断の結果が格納されます。
DT_9001	瞬停回数			瞬停の回数が格納されます。
DT_9002	ヒューズ断ユニット	正常：0 異常：1		bit0～bit7～bit15 →スロット0→スロット7→スロット15
DT_9003	ヒューズ断ユニット	正常：0 異常：1		bit0～bit7～bit15 →スロット16→スロット23→スロット31
DT_9004	異常高機能ユニット	正常：0 異常：1		bit0～bit7～bit15 →スロット0→スロット7→スロット15
DT_9005	異常高機能ユニット	正常：0 異常：1		bit0～bit7～bit15 →スロット16→スロット23→スロット31
DT_9010	照合異常ユニット	正常：0 異常：1		bit0～bit7～bit15 →スロット0→スロット7→スロット15
DT_9011	照合異常ユニット	正常：0 異常：1		bit0～bit7～bit15 →スロット16→スロット23→スロット31
DT_9017	演算エラー行番号			最初の演算エラーが発生した行番号が格納されます。
DT_9020	ユーザープログラム容量			PRM（0）に設定されたユーザプログラムの容量が格納されます。 単位：ワード
DT_9021	ファイルメモリ最大値			ファイルメモリ（FL）の最大番号が格納されます。
DT_9022	スキャンタイム（現在値）			スキャンタイム（マルチタスクの1周期）の現在の値が格納されます。 単位：100 μS　例）200→20ms以内、225→22.5ms以内。
DT_9023	スキャンタイム（最小値）			スキャンタイム（マルチタスクの1周期）の最小値が格納されます。
DT_9024	スキャンタイム（最大値）			スキャンタイム（マルチタスクの1周期）の最大値が格納されます。
DT_9029	ポーズ番号			テストRUN時のポーズしている行番号が格納されます。
DT_9036	リモートI/Oメモリアクセス命令完了異常コード	正常時：0		
		異常時	58H	タイムアウト：相手先不在で送信不可状態
			58H	アクセスエリア無エラー：指定したメモリアクセスエリアがスレーブユニット上に存在しない
			58H	送信アンサー待ちタイムアウトエラー
			58H	送信バッファアンサー空き待ちタイムアウトエラー
			58H	レスポンス待ちタイムアウトエラー
DT_9039	SEND/RECV完了コード	正常終了：0		MEWNETデータ転送命令（SEND/RECV）

番号	名称	内容	
DT_9131	異常子局の確認	上位8ビット：マスタユニット選択	0～3（CPUに近い順）
		下位8ビット：表示選択	00H：異常子局表示 01H：I/O異常子局
DT_9132	異常子局スロット	bit 0・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・bit 15 スロット0・・・・・・・・・・・・・・・・・・・・・・・・・・・・・スロット15	
		DT_9131：異常子局表示（現在値）	0：正常 1：異常
		DT_9131：I/O異常子局表示（I/O属性変化）	0：変化無 1：変化有
DT_9133	異常子局スロット	bit 16・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・bit 31 スロット0・・・・・・・・・・・・・・・・・・・・・・・・・・・・・スロット15	
		DT_9131：異常子局表示（現在値）	0：正常 1：異常
		DT_9131：I/O異常子局表示（I/O属性変化）	0：変化無 1：変化有
DT_9134	異常子局スロット	bit 0・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・bit 15 スロット0・・・・・・・・・・・・・・・・・・・・・・・・・・・・・スロット15	
		DT_9131：異常子局表示（累積値）	0：正常 1：異常
		DT_9131：I/O異常子局表示（瞬停止）	0：無 1：有
DT_9135	異常子局スロット	bit 16・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・bit 31 スロット0・・・・・・・・・・・・・・・・・・・・・・・・・・・・・スロット15	
		DT_9131：異常子局表示（累積値）	0：正常 1：異常
		DT_9131：I/O異常子局表示（瞬停止）	0：無 1：有
DT_9136	リモートI/O異常コード	ビット0～7：マスタユニット1	0：正常 1：異常
		ビット8～15：マスタユニット2	
DT_9137	リモートI/O異常コード	ビット0～7：マスタユニット3	0：正常 1：異常
		ビット8～15：マスタユニット4	

番号	名称	内容
DT_9140	PCリンクステータス	PCリンク0の受信回数RINGカウンタ
DT_9141		PCリンク0の受信間隔現在値（＊2.5ms）
DT_9142		PCリンク0の受信間隔最小値（＊2.5ms）
DT_9143		PCリンク0の受信間隔最大値（＊2.5ms）
DT_9144		PCリンク0の送信回数RINGカウンタ
DT_9145		PCリンク0の送信間隔現在値（＊2.5ms）
DT_9146		PCリンク0の送信間隔最小値（＊2.5ms）
DT_9147		PCリンク0の送信間隔最大値（＊2.5ms）
DT_9148		PCリンク1の受信回数RINGカウンタ
DT_9149		PCリンク1の受信間隔現在値（＊2.5ms）
DT_9150		PCリンク1の受信間隔最小値（＊2.5ms）
DT_9151		PCリンク1の受信間隔最大値（＊2.5ms）
DT_9152		PCリンク1の送信回数RINGカウンタ
DT_9153		PCリンク1の送信間隔現在値（＊2.5ms）
DT_9154		PCリンク1の送信間隔最小値（＊2.5ms）
DT_9155		PCリンク1の送信間隔最大値（＊2.5ms）
DT_9160	リンクユニットNo.	LINK1のユニットNo.が格納されます。
DT_9161	異常フラグ	LINK1の異常フラグが格納されます。
DT_9162	リンクユニットNo.	LINK2のユニットNo.が格納されます。
DT_9163	異常フラグ	LINK2の異常フラグが格納されます。
DT_9164	リンクユニットNo.	LINK3のユニットNo.が格納されます。
DT_9165	異常フラグ	LINK3の異常フラグが格納されます。
DT_9170	LINK1ステータス	PC-LINKアドレス重複先
DT_9171		テストモード結果
DT_9172		トークン紛失回数
DT_9173		（2重トークン回数）
DT_9174		無信号状態回数
DT_9175		同期異常回数
DT_9176		送信NACK
DT_9177		送信NACK
DT_9178		送信WACK
DT_9179		送信WACK
DT_9180		送信アンサー
DT_9181		送信アンサー
DT_9182		未定義コマンド
DT_9183		パリティエラー回数
DT_9184		Endcode受信エラー
DT_9185		フォーマットエラー
DT_9186		NOTサポート
DT_9187		自己診断結果
DT_9188		ループ切り替え回数
DT_9189		リンク不可状態発生回数
DT_9190		主ルート入力断線回数
DT_9191		副ルート入力断線回数
DT_9192		ループ再構成処理中
DT_9193		ループ運転モード
DT_9194		ループ入力状態

番号	名称	内容
DT_9200	LINK2ステータス	PC-LINKアドレス重複先
DT_9201		テストモード結果
DT_9202		トークン紛失回数
DT_9203		（2重トークン回数）
DT_9204		無信号状態回数
DT_9205		同期異常回数
DT_9206		送信NACK
DT_9207		送信NACK
DT_9208		送信WACK
DT_9209		送信WACK
DT_9210		送信アンサー
DT_9211		送信アンサー
DT_9212		未定義コマンド
DT_9213		パリティエラー回数
DT_9214		Endcode受信エラー
DT_9215		フォーマットエラー
DT_9216		NOTサポート
DT_9217		自己診断結果
DT_9218		ループ切り替え回数
DT_9219		リンク不可状態発生回数
DT_9220		主ルート入力断線回数
DT_9221		副ルート入力断線回数
DT_9222		ループ再構成処理中
DT_9223		ループ運転モード
DT_9224		ループ入力状態
DT_9230	LINK3ステータス	PC-LINKアドレス重複先
DT_9231		テストモード結果
DT_9232		トークン紛失回数
DT_9233		（2重トークン回数）
DT_9234		無信号状態回数
DT_9235		同期異常回数
DT_9236		送信NACK
DT_9237		送信NACK
DT_9238		送信WACK
DT_9239		送信WACK
DT_9240		送信アンサー
DT_9241		送信アンサー
DT_9242		未定義コマンド
DT_9243		パリティエラー回数
DT_9244		Endcode受信エラー
DT_9245		フォーマットエラー
DT_9246		NOTサポート
DT_9247		自己診断結果
DT_9248		ループ切り替え回数
DT_9249		リンク不可状態発生回数
DT_9250		主ルート入力断線回数
DT_9251		副ルート入力断線回数
DT_9252		ループ再構成処理中
DT_9253		ループ運転モード
DT_9254		ループ入力状態

3-7
FP-BASICエラーコード一覧

FP-BASICインタープリタからのエラーメッセージです。

エラーコード	内　容
1	NEXT文に対応するFOR文がありません。
2	1.文法エラーが発生しました。 2.コマンドで使用した変数が定義されていません。 3.配列として使用した変数が配列宣言されていません。
3	GOSUB文がないのにRETURN文を実行しようとしました。
4	1.プログラム中のGOTO,GOSUB,ONERR,ON GOTO,ON GOSUB文が合わせて600個を越えました。 2.プログラム中のラベルが400個を越えました。
6	数字や変数のオーバーフローが発生しました。
7	1.GOSUB文のネスト・オーバーが発生しました。 （GOSUB文のネスティングは最大8段まで可能） 2.エディタのバッファオーバーフローが発生しました。 〔64kbyte以上のプログラムファイルはLOADおよび編集はできません。ただし、市販エディタで64kbyte以上のプログラムファイルを作成すると、COMPILE、DWNLDは可能です。〕
8	1.GOTO、GOSUBで呼ばれた行番号がありません。 2.削除したい行番号がありません。
9	1.宣言していない配列変数を使おうとしました。 2.配列の大きさが定義サイズを越えました。（配列領域オーバー）
10	1.変数が二重定義されています。 2.予約語が変数として使用されています。
11	0による割り算が実行されています。
12	コマンドで使用すべき命令をステートメントとして使用しました。（またはその逆）
13	左カッコと右カッコの数が合いません。
14	パラメータの数が合いません。（配列で配列要素が合いません。）
15	DIMで定義されていない関数、変数または配列を使おうとしました。
16	FUNCTION～FENDの間以外にプログラムがあります。
17	FUNCTION～FENDの中でFUNCTION宣言をしました。
18	変数宣言文が行の先頭にありません。
19	ループの外でEXITしています。
20	ループが正しく対応していません。
21	変数の型変換エラーです。
22	パラメータの設定テーブルが設定範囲を越えました。
23	1行の文字数が多すぎます。（255文字まで）
24	中間コードエラーです。 1.未定義コマンドを実行しようとしました。 〔IF文、FOR分でERROR 24が発生した場合、FP3、FP1使用時は起動時のコマンドオプション/Hを付けて、FP3H使用時は/Hを付けずに再起動してください。〕 2.関数のパラメータタイプが不適当です。
25	1.数字を数値に変換したとき、桁が多すぎます。 2.INPUT文によるASCII文字が数値に正しく変換できません。
26	存在しない機器を使用しようとしました。
27	存在しない入出力番号を指定しました。
28	FORループに対応するNEXT文がありません。
29	CASE文に対応するSELECT文がありません。
30	INPUT文によるデータの受信数が合いません。

<table>
<tr><th>エラーコード</th><th>内 容</th></tr>
<tr><td>31</td><td>接続またはボーレートを確認してください。</td></tr>
<tr><td>33</td><td>RS-232Cからターミネータのないデータが、バッファをオーバーして入力されました。</td></tr>
<tr><td>34</td><td>RS-232C通信のパリティ、オーバーラン、フレーミングエラーです。</td></tr>
<tr><td>35</td><td>DEFAULT文に対応するSELECT文がありません。</td></tr>
<tr><td>37</td><td>SELEND文に対応するSELECT文がありません。</td></tr>
<tr><td>38</td><td>SELECT文に対応するSELEND文がありません。</td></tr>
<tr><td>39</td><td>プログラム番号が見つかりません。</td></tr>
<tr><td>40</td><td>存在しないタスクを指定しています。</td></tr>
<tr><td>41</td><td>すでにタスクが起動されており、起動できません。</td></tr>
<tr><td>45</td><td>タスクの実行中に許されないコマンドを実行しました。</td></tr>
<tr><td>53</td><td>存在しないプログラムファイルを指定しています。</td></tr>
<tr><td>54</td><td>存在しないディレクトリを指定しています。</td></tr>
<tr><td>55</td><td>ライン移動先のラインナンバーがすでに存在しています。</td></tr>
<tr><td>57</td><td>同じ名前のプログラムファイルがすでに存在しています。</td></tr>
<tr><td>58</td><td>1.WHILE文に対応するWEND文がありません。
2.ファイル名が間違っています。（FILES）
3.同名のファイルがすでに存在しています。（NAME）</td></tr>
<tr><td>59</td><td>WEND文に対応するWHILE文がありません。</td></tr>
<tr><td>60</td><td>1.ディスクに空きスペースがありません。
2.メインメモリが足りません。（必要メモリサイズ471Kbyte）</td></tr>
<tr><td>62</td><td>プログラムファイルの名前が正しくありません。</td></tr>
<tr><td>63</td><td>ディスクの準備ができていません。</td></tr>
<tr><td>64</td><td>オブジェクト・データが違います。</td></tr>
<tr><td>65</td><td>1ファイルあたりのEXTERNのシンボル数が400個を越えました。</td></tr>
<tr><td>66</td><td>ディスクライトエラーです。</td></tr>
<tr><td>67</td><td>ドライブセレクトエラーです。</td></tr>
<tr><td>68</td><td>書き込みが禁止されているディスクに書き込みを行おうとしました。</td></tr>
<tr><td>70</td><td>GOSUB文がないのにRETURN文を実行しようとしました。</td></tr>
<tr><td>71</td><td>数値オーバーフローです。</td></tr>
<tr><td>79</td><td>ENDIF文に対応するIFB文がありません。</td></tr>
<tr><td>80</td><td>ELSE文に対応するIFB文がありません。</td></tr>
<tr><td>81</td><td>IFB文に対応するENDIF文がありません。</td></tr>
<tr><td>82</td><td>ELSEIF文に対応するIFB文がありません。</td></tr>
<tr><td>84</td><td>コンパイルされたオブジェクト文が大きすぎます。（中間コードの格納領域が足りません。）
<table>
<tr><th>メモリ ＼ ユニット</th><th>FP3H</th><th>FP3</th><th>FP1</th></tr>
<tr><td>プログラムエリア（byte）</td><td>128K</td><td>64K</td><td>12K</td></tr>
<tr><td>変数エリア（byte）</td><td>41K</td><td>18K</td><td>3K</td></tr>
</table>
</td></tr>
<tr><td>85</td><td>EXITするループが足りません。</td></tr>
<tr><td>88</td><td>DO文に対応するLOOP文がありません。</td></tr>
<tr><td>89</td><td>LOOP文に対応するDO文がありません。</td></tr>
</table>

エラーコード	内　容
90	IFB,SELECT,LOOP文等のネストオーバーが発生しました。 IFB文のネスティングは最大9段まで可能。 SELECT文のネスティングは最大10段まで可能。 SELECT内のCASEとDEFAULTの数は合計13個まで可能。 LOOP文ネスティングは最大10段まで可能。
93	関数に与えるパラメータの数が合いません。
100	デバイス通信エラーです。
134	実数の演算エラーが発生しました。
135	タグが異常です。
136	与えられた関数が存在しません。
137	規定範囲外のパラメータが指定されました。
204	1.スタック・アンダーフローです。 2.BASICデータスタックのアンダーフローです。
208	直接命令の入力データが大きすぎます。（プリントデータが長すぎます。58文字まで。）
258	オブジェクトが存在しません。
259	ラベル情報が大きすぎます。
260	FUNCTION数が大きすぎます。（Ver2.3以降100個まで、Ver2.3未満50個まで）
261	メモリーが足りません。
1002	タスク起動中のXQTです。（タスクを2重起動しようとしました。）
1003	HALT中の再起動異常です。
1004	WAIT I/Oのタイムアウトです。
1005	指定されたシンボルがありません。
1006	モードが違います。 （RUNモードでDWNLD,UPLDしようとしました。PROGモードでXQTしようとしました。）
1007	スタックが空です。
1008	コマンドの使用方法が間違っています。
1009	パラメータが間違っています。（DT_、LD_のビット操作）
1010	未定値演算です。
1011	割り込み中のサブルーチンコールです。
1012	指定タスクがPAUSE状態ではありません。
1013	メモリーROMエラーです。
1014	メモリープロテクトエラーです。
1015	MEWNETデータリンクエラーです。
1016	MEWNETデータリンクがアクティブではありません。
1017	指定されたスロットに共有RAMがありません。
1018	WAITで他のデバイスの入力が指定されました。
1019	WAIT I/OのI/O数が多すぎます。(FP3H BASICは64個まで、FP1 BASICは24個までです。)
1020	タスク指定が不適当です。
1098	I/O種別エラーです。
1099	メモリアロケーションエラーです。
1101	リモートI/Oの設定異常です。
1200	数値をあらわす文字列として正しくありません。

エラーコード	内　容
1201	変数のタイプが等しくありません。
1202	データを最後まで読み込んだ後、再びDREAD文を実行しました。
1203	データ文が見つかりません。
1250	親局が見つかりません。
1251	親局が最大親局数（4）を越えています。
1252	子局が最大子局数（32）を越えています。
1253	スロットが最大スロット数を越えています。
1254	スロット機能番号が正しくありません。
1255	親局の総スロット数が最大親局総スロット数（64）を越えています。
1256	全体の総スロット数が最大（全体）総スロット数（128）を越えています。
1257	リモートI/Oのメモリアクセスエラーです。
1300	ユニットIDが正しくありません。
1301	ユニット種別が正しくありません。
1302	チャンネル番号が正しくありません。
1303	平均回数が正しくありません。
1304	終端コードタイプが正しくありません。
1305	入力フィルタ定数が正しくありません。
1306	ユニットタイプが正しくありません。
1307	軸番号が正しくありません。
1308	JOB番号が正しくありません。
1309	始動データNo.が正しくありません。
1310	データ長が正しくありません。
1311	IDアドレスが正しくありません。
1312	ID番号が正しくありません。
1313	クリアID番号が正しくありません。
1314	エラーレスポンスを受け取りました。
1315	シリアルポート付きのハードウェアが無いのにシリアルポートをアクセスするコマンドを使用しようとしました。
1316	高機能ユニットエラーです。
1400	ファイル名の指定が間違っています。
1401	オープンしていないファイルを参照しようとしました。
1402	すでにオープンされているファイルを再度オープンしようとしました。
1403	ファイル中のすべてのデータを読みつくした後、さらに読み込もうとしました。
2002	I/Oの2度読み異常です。
2003	I/O点数異常です。
2004	BASICスタックオーバーフローです。 （GOSUB,CALL～FUNCTION,PRIVATEを合わせた数が多すぎます。MAP命令を参照し、このエラーの発生したタスクのスタック領域を増やしてください。）
2005	ONERR処理中のエラーです。
2006	IFレベルオーバーです。
2007	内部タグコードエラーです。
3000	通信シーケンスエラーです。

3-8
自己診断エラーコード一覧

PCのCPU本体からのエラーメッセージです。

番　号	内　容
23	RAM異常1エラー
25	RAM異常2エラー
26	ユーザーROMサムチェックエラー
27	特殊ユニット装着制限エラー
28	パラメータメモリ異常エラー
29	メモリ管理テーブル異常エラー
30	割り込み異常1エラー
31	割り込み異常2エラー
33	マルチパラメータ照合チェックエラーコード
35	スレーブ上に組み合わせ禁止のユニットが実装されている
36	リモートI/O仕様制限エラー（重複あるいは範囲オーバー）
38	I/Oターミナルボード登録エラー
40	出力ユニットヒューズ断線
41	特殊ユニット暴走エラー
42	I/O照合エラー
45	演算エラー発生
46	リモートI/O交信エラー
47	スレーブ上のI/O属性エラー
50	電池電圧低下エラー
51	リモートI/O終端局エラー
53	ラダー用I/Oマップ照合チェックワーニングコード
60	ESB受信バッファオーバーフローエラー

3-9
高機能ユニットパラメーター覧表

(1)FP3　A/D変換ユニット(AFP3400)

■共有メモリ割り付け表

ワードアドレス	バイトアドレス	項目			内容	
0	0	平均処理	平均処理CHに指定		平均処理を行なうCHを指定	*1
1	2		平均をとる回数（CH0）		CH0入力についてサンプル回数を設定	設定範囲：3～4000回 *2
2	4		平均をとる回数（CH1）		CH1入力についてサンプル回数を設定	
3	6		平均をとる回数（CH2）		CH2入力についてサンプル回数を設定	
4	8		平均をとる回数（CH3）		CH3入力についてサンプル回数を設定	
5	10	変換値が設定範囲外の時の警報信号発生（サンプリング処理時のみ有効）	警報信号発生CHに指定		変換値が設定範囲外の時、警報を発生させるCHを指定	*3,*4
6	12		CH0	上限値	CH0入力について上限値を設定	設定範囲：−32768～32767回 ただし、下限値＜上限値
7	14			下限値	CH0入力について下限値を設定	
8	16		CH1	上限値	CH1入力について上限値を設定	
9	18			下限値	CH1入力について下限値を設定	
10	20		CH2	上限値	CH2入力について上限値を設定	
11	22			下限値	CH2入力について下限値を設定	
12	24		CH3	上限値	CH3入力について上限値を設定	
13	26			下限値	CH3入力について下限値を設定	
14	28	デジタルスケーリング処理	スケーリング処理CHの指定		デジタルスケーリング処理を行なうCHを指定	*3
15	30		CH0	オフセット値	CH0入力についてオフセット値を設定	設定範囲：−32768～32767回 ただし、オフセット値＜フルスケール値
16	32			フルスケール値	CH0入力についてフルスケール値を設定	
17	34		CH1	オフセット値	CH1入力についてオフセット値を設定	
18	36			フルスケール値	CH1入力についてフルスケール値を設定	
19	38		CH2	オフセット値	CH2入力についてオフセット値を設定	
20	40			フルスケール値	CH2入力についてフルスケール値を設定	
21	42		CH3	オフセット値	CH3入力についてオフセット値を設定	
22	44			フルスケール値	CH3入力についてフルスケール値を設定	
23	46	A/D変換後のデジタル出力値	CH0入力を変換した出力値（読出用）		CH0入力に指定の処理を行なって変換した値が格納されます	*5
24	48		CH1入力を変換した出力値（読出用）		CH1入力に指定の処理を行なって変換した値が格納されます	*5
25	50		CH2入力を変換した出力値（読出用）		CH2入力に指定の処理を行なって変換した値が格納されます	*5
26	52		CH3入力を変換した出力値（読出用）		CH3入力に指定の処理を行なって変換した値が格納されます	*5
27	54	エラーコード（読出用）			各種エラーに対応したビットが1になります	

注) *1　平均処理に指定しないCHについてはサンプリング処理になります。
*2　平均をとる回数が指定範囲外の時は、サンプリング処理になります。
*3　指定方法:指定するCHに対応するビットを1にしてください。
*4　警報の発生は、各CHに割り当てられているCPUユニットの入力XをONにすることによって行ないます。
I/O割り付けをご参照ください。
*5　出力値の範囲は、レンジおよびスケールによって異なります。
(1)デフォルト値のスケールでは次のようにします。
・1～5V/4～20mAレンジ:0～4000
・0～±10V/0～±20mAレンジ:0～±2000
(2)デジタルスケーリング処理を行なう場合は、設定したオフセット値からフルスケール値までの範囲です。

参考　A/D変換ユニット　処理方法

使用する機能	設定する項目	サンプリング処理（アドレス0:OFF）	平均処理（アドレス0:ON）	デジタルスケーリング処理（アドレス14:ON）	上・下限値制限（警報付）（アドレス15−22）
サンプリング	設定なし	●	×	−	−
サンプリング/警報	上下限値	●	×	−	●
サンプリング/スケーリング	オフセット/フルスケール	●	×	●	−
サンプリング/スケーリング/警報	オフセット/フルスケール/、上下限値	●	×	●	●
平均	回数	×	●	−	×
平均/スケーリング	回数、オフセット/フルスケール	×	●	●	×

■I/O割り付け表

・占有I/O点数は「入力16点」です。出力はありません。(機能番号220)

・基本マザーボードのスロット0に装着した時の割り付けです。

入力（A/D変換ユニット→CPUユニット）

I/O	内容		ON条件
X_&M0	設定完了フラグ		電源投入時および共有メモリ書き込み時に、全CHからの入力がサンプリング処理で変換されてメモリに格納された時 （設定内容に関係なく上記の内容で行なわれます）
X_&M1	A/D変換完了フラグ（CH0）		CH0からの入力が指定の処理方法に基づいて変換されてメモリに格納された時
X_&M2	A/D変換完了フラグ（CH1）		CH1からの入力が指定の処理方法に基づいて変換されてメモリに格納された時
X_&M3	A/D変換完了フラグ（CH2）		CH2からの入力が指定の処理方法に基づいて変換されてメモリに格納された時
X_&M4	A/D変換完了フラグ（CH3）		CH3からの入力が指定の処理方法に基づいて変換されてメモリに格納された時
X_&M5	未使用		
X_&M6	未使用		
X_&M7	未使用		
X_&M8	CH0	上限値越え警報フラグ	CH0を警報信号発生CHに指定している時に、変換値＞上限値の場合
X_&M9		下限値越え警報フラグ	CH0を警報信号発生CHに指定している時に、変換値＜下限値の場合
X_&MA	CH1	上限値越え警報フラグ	CH1を警報信号発生CHに指定している時に、変換値＞上限値の場合
X_&MB		下限値越え警報フラグ	CH1を警報信号発生CHに指定している時に、変換値＜下限値の場合
X_&MC	CH2	上限値越え警報フラグ	CH2を警報信号発生CHに指定している時に、変換値＞上限値の場合
X_&MD		下限値越え警報フラグ	CH2を警報信号発生CHに指定している時に、変換値＜下限値の場合
X_&ME	CH3	上限値越え警報フラグ	CH3を警報信号発生CHに指定している時に、変換値＞上限値の場合
X_&MF		下限値越え警報フラグ	CH3を警報信号発生CHに指定している時に、変換値＜下限値の場合

(2)D/A変換ユニット(AFP3410,3411)

■共有メモリ割り付け表

ワードアドレス	バイトアドレス	項目			内容	
0	0	データが設定範囲外の時の警報信号発生	警報信号発生CHの指定		データが設定範囲外の時、警報を発生させるCHを指定 ＊1,＊2	
1	2		CH0	上限値	CH0に出力するデータの上限値を設定	設定範囲： ＊3 -2000～2000回（AFP3410） 0～4000回（AFP3411） ただし、下限値＜上限値
2	4			下限値	CH0に出力するデータの下限値を設定	
3	6		CH1	上限値	CH1に出力するデータの上限値を設定	
4	8			下限値	CH1に出力するデータの下限値を設定	
5	10	アナログ出力に変換するデータの設定		CH0	D/A変換するデータを書き込む CH0からアナログ出力されます	設定範囲： ＊3 -2000～2000回（AFP3410） 0～4000回（AFP3411）
6	12			CH1	D/A変換するデータを書き込む CH1からアナログ出力されます	
7	14	エラーコード（読出用）			各種エラーに対応したビットが1になります	

注）＊1指定方法:指定するCHに対応するビットを1にしてください。
＊2警報の発生は、各CHに割り当てられているCPUユニットの入力XをONにすることによって行ないます。
I/O割り付けをご参照ください。
＊3変換元データの範囲は、D/A変換ユニットのタイプによって異なります。
・AFP3410（0～±10V/0～±20mAレンジ）　:－2000～2000
・AFP3411（1～±5V/4～20mAレンジ）　:0～4000

■I/O割り付け表

・占有I/O点数は「入力16点」です。出力はありません。(機能番号220)

・基本マザーボードのスロット0に装着した時の割り付けです。

入力（D/A変換ユニット→CPUユニット）

I/O	内容		ON条件
X_&M0	上限値・下限値設定完了フラグ		警報信号発生処理のために、CH指定および上限値・下限値の設定が共有メモリに書き込まれた時 ＊1
X_&M1	書き込みデータ無効フラグ（CH0）		CH0から出力するように書き込まれた変換元データが指定範囲外の時
X_&M2	書き込みデータ無効フラグ（CH1）		CH1から出力するように書き込まれた変換元データが指定範囲外の時
X_&M3	CH0	上限値越え警報フラグ	CH0から出力するように書き込まれた変換元データ＞上限値の時 ＊2 ＊2
X_&M4		下限値越え警報フラグ	CH0から出力するように書き込まれた変換元データ＜下限値の時 ＊2 ＊2
X_&M5	CH1	上限値越え警報フラグ	CH1から出力するように書き込まれた変換元データ＞上限値の時 ＊2 ＊2
X_&M6		下限値越え警報フラグ	CH1から出力するように書き込まれた変換元データ＜下限値の時 ＊2 ＊2
X_&M7～X_&MF	未使用		

注）＊1フラグがONになっていても、警報信号発生CHの指定および上限値・下限値の設定を新たに行った場合は、一旦クリアされます。
＊2フラグがONになっていても、警報信号発生CHの指定および上限値・下限値の設定を新たに行った場合は、クリアされます。

(3) シリアルデータユニット(AFP3460)

■共有メモリ割り付け表

ワードアドレス	バイトアドレス	項目		内容	
1～250	2～500	CH1	送信用バッファ	CH1から送信するデータ（最大500文字）を設定してください。 出力Y_&M10をONにすると、設定したデータがCH1に接続している RS232C機器に送信されます。	
251～500	502～1000		受信用バッファ（読出用）	CH1で受信したデータ（最大500文字）が設定されます。 CH1に接続しているRS232C機器から、正常なデータを受信する と、入力X_&M1がONになります。	
501～750	1002～1500	CH2	送信用バッファ	CH2から送信するデータ（最大500文字）を設定してください。 出力Y_&M12をONにすると、設定したデータがCH1に接続している RS232C機器に送信されます。	
751～1000	1502～2000		受信用バッファ（読出用）	CH2で受信したデータ（最大500文字）が設定されます。 CH2に接続しているRS232C機器から、正常なデータを受信する と、入力X_&M2がONになります。	
1001	2002	CH1終端コード（読出用）		CH1から送信するデータの終端コード （1バイト）を任意に設定。	ディップスイッチで、終端コードの任意設定を選択してください。（*1）
1002	2004	CH2終端コード（読出用）		CH2から送信するデータの終端コード （1バイト）を任意に設定。	

注）*1ディップスイッチの設定と終端コードの対応は次の通りです。

CH1	DSW2-1	DSW2-2	終端コード	説明
CH2	DSW3-5	DSW3-6		
	OFF	OFF	任意設定	共有メモリに終端コード（下位1バイト）を設定してください。 設定した終端コードは、送信時データの末尾に自動的に付加されます。
	ON	OFF	C_R	指定した終端コードをデータ末尾に付加して送信してください。 自動的には付加されませんのでご注意ください。
	OFF	ON	$C_R + LF$	
	ON	ON	ETX	

■I/O割り付け表

・占有I/O点数は「入力16点・出力16点」です。(機能番号222)

・基本マザーボードのスロット0に装着した時の割り付けです。

入力（シリアルデータユニット→CPUユニット）

I/O	内容		説明
X_&M0	CH1	送信完了フラグ	CH1からの送信が完了した時、ON 送信用バッファがクリアされるとOFFします。
X_&M1		受信完了フラグ	CH1に接続しているRS232C機器から正常なデータを受信した時、ON 受信用バッファがクリアされるとOFFします。
X_&M2	CH2	送信完了フラグ	CH2からの送信が完了した時、ON 送信用バッファがクリアされるとOFFします。
X_&M3		受信完了フラグ	CH2に接続しているRS232C機器から正常なデータを受信した時、ON 受信用バッファがクリアされるとOFFします。
X_&M4	CH1	フレーミングエラー	受信データのデータ長、ストップビット長が設定と異なる時
X_&M5	エラーフラグ	パリティエラー	受信データのパリティチェックが設定と異なる時
X_&M6	（*1）	バッファフルエラー	受信バッファがいっぱいの時
X_&M7		受信メッセージ長エラー	受信データのフレーム長が500バイトを越える時
X_&M8		送信メッセージ長エラー	送信データのフレーム長が500バイトを越える時
X_&M9	CH1	フレーミングエラー	受信データのデータ長、ストップビット長が設定と異なる時
X_&MA	エラーフラグ	パリティエラー	受信データのパリティチェックが設定と異なる時
X_&MB	（*1）	バッファフルエラー	受信バッファがいっぱい時
X_&MC		受信メッセージ長エラー	受信データのフレーム長が500バイトを越える時
X_&MD		送信メッセージ長	送信データのフレーム長が500バイトを越える時
X_&ME	ソフトリセット（イニシャライズ）完了フラグ		ソフトリセットを実行し、イニシャライズが終了した時
X_&MF	未使用		

注）*1エラーフラグは、正常なデータを受信するか、リセットスイッチを押すとOFFします。

出力（CPUユニット→シリアルデータユニット）

I/O	内容		説明
Y_&M10	CH1	送信指示	ONすると、CH1送信用バッファ（共有メモリのアドレス1～250）に設定したデータをCH1から送信します。 *1 OFFすると、送信用バッファをクリアします。
Y_&M11		受信用バッファクリア	ONすると、CH1受信用バッファ（共有メモリのアドレス251～500）をクリアします。 *2 バッファのクリアが終了（X_&M1:OFF）したら、OFFしてください。
Y_&M12	CH2	送信指示	ONすると、CH2送信用バッファ（共有メモリのアドレス501～750）に設定したデータをCH2から送信します。 *1 OFFすると、送信用バッファをクリアします。
Y_&M13		受信用バッファクリア	ONすると、CH2受信用バッファ（共有メモリのアドレス751～1000）をクリアします。 *2 バッファのクリアが終了（X_&M3:OFF）したら、OFFしてください。
Y_&M14～Y_&M1C	未使用		
Y_&M1D	ソフトリセット		ONすると、イニシャライズします（約100ms間）。 イニシャライズが終了（XE:OFF）したら、OFFしてください。
Y_&M1E	CH1終端コードなしモード切り換え		ON時、終端コードを送信しないモード *3
Y_&M1F	CH2終端コードなしモード切り換え		ON時、終端コードを送信しないモード *3

注）*1送信用バッファのクリアは、送信完了(X_&M0:ONまたはX_&M2:ON)後に行なってください。

*2受信用バッファのクリアは、共有メモリからデータを読み出した後に行なってください。

*3終端コードを送信しないモードの時に、送信指示(Y_&M10:ONまたはY_&M12:ON)を行なうと、終端コードが送られません。プリンタ出力時に改行を防ぐ場合などに利用できます。

(4)高速カウンタユニット(AFP3622)

■共有メモリ割り付け表

ワードアドレス	バイトアドレス	項目		内容	
0	0	CH0	出力パルス数　経過値（32ビットデータ）	高速カウンタでの経過値が格納されます。任意の値に書き替えることもできます。	計数範囲（設定範囲）：-16,777,216～16,777,215
1	2				
2	4		出力パルス数　目標値（32ビットデータ）	C=P一致出力モード時、一致出力を行なう目標値を設定。	設定範囲：-16,777,216～16,777,215
3	6				
4	8		入力時定数（32ビットデータ）	カウント入力のレンジを設定。	設定範囲：0～3
5	10				
6	12	未使用			
7	14				
8	16	CH1	出力パルス数　経過値（32ビットデータ）	高速カウンタでの経過値が格納されます。任意の値に書き替えることもできます。	計数範囲（設定範囲）：-16,777,216～16,777,215
9	18				
10	20		出力パルス数　目標値（32ビットデータ）	C=P一致出力モード時、一致出力を行なう目標値を設定。	設定範囲：-16,777,216～16,777,215
11	22				
12	24		入力時定数（32ビットデータ）	カウント入力のレンジを設定。	設定範囲：0～3
13	26				

■I/O割り付け表

・占有I/O点数は「入力16点・出力16点」です。(機能番号222)

・基本マザーボードのスロット0に装着した時の割り付けです。

入力（高速カウンタユニット→CPUユニット）

I/O	内容		説明
X_&M0	CH0	目標値一致出力	C=PモードではCH0の経過値=目標値の時 C=0モードではCH0の経過値=0の時
X_&M1		比較出力	CH0の経過値>目標値の時
X_&M2		オーバーフロー/アンダーフローフラグ	内部カウンタの加算または減算が計数範囲を越えて行われた時
X_&M3	未使用		
X_&M4	CH1	目標値一致出力	C=PモードではCH1の経過値=目標値の時 C=0モードではCH1の経過値=0の時
X_&M5		比較出力	CH1の経過値>目標値の時
X_&M6		オーバーフロー/アンダーフローフラグ	内部カウンタの加算または減算が計数範囲を越えて行われた時
X_&M7～X_&MF	未使用		

出力（CPUユニット→高速カウンタユニット）

I/O	内容		説明
Y_&M10	CH0	リセット　＊1	ONの時、経過値・目標値クリア、各出力およびフラグ：OFF
Y_&M11		出力許可	ONの時、目標値一致出力および比較出力が可能
Y_&M12	CH1	リセット　＊1	ONの時、経過値・目標値クリア、各出力およびフラグ：OFF
Y_&M13		出力許可	ONの時、目標値一致出力および比較出力が可能
Y_&M14～Y_&M1F	未使用		

＊1　外部入力RSTと同じ働きをします。

(5)パルス出力ユニット(AFP3480)

■共有メモリ割り付け表

ワードアドレス	バイトアドレス	項目	内容	
0	0	出力パルス数　経過値（32ビットデータ）	パルス出力ユニット内部の高速カウンタでの経過値が格納されます。任意の値に書き換えることができます。	計数範囲（設定範囲）： -16,777,216～16,777,215
1	2			
2	4	出力パルス数　目標値（32ビットデータ）	C=P一致出力モード時、一致出力を行なう目標値を設定します。	計数範囲（設定範囲）： -16,777,216～16,777,215
3	6			

■I/O割り付け表

・占有I/O点数は「入力16点・出力16点」です。(機能番号222)

・基本マザーボードのスロット0に装着した時の割り付けです。

入力（パルス出力ユニット→CPUユニット）

I/O	内容	ON条件
X_&M0	目標値一致出力	C=PモードではCH0の経過値=目標値の時 C=0モードではCH0の経過値=0の時
X_&M1	比較出力	CH0の経過値>目標値の時
X_&M2	オーバーフロー/アンダーフローフラブ	内部カウンタの加算または減算が計数範囲を越えて行なわれた時
X_&M3	未使用	
X_&M4	原点復帰中フラグ	原点復帰動作中　*1
X_&M5～X_&MF	未使用	

*1　原点復帰動作は、Y14:ONで原点復帰モードに切り替わった状態でY11がONすると開始されます。原点入力(ZERO)がONになると停止します。

出力（CPUユニット→パルス出力ユニット）

I/O	内容	動作
Y_&M10	リセット　*1	ONの時、パルス出力停止、経過値・目標値クリア、各フラグOFF
Y_&M11	パルス出力スタート	ONの時、パルス出力開始
Y_&M12	パルス出力方向切り換え	OFFの時、正転（CW）/ONの時、逆転（CCW）
Y_&M13	パルス周波数切り換え　*2	OFFの時、低速パルス/ONの時、高速パルス
Y_&M14	原点復帰モード切り換え	ONの時、原点復帰モードに切り換え
Y_&M15～Y_&M1F	未使用	

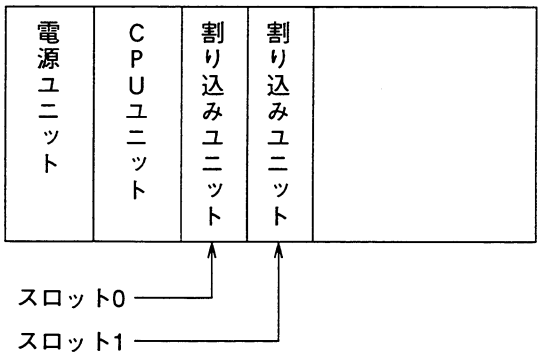
*1　外部入力RSTと同じ働きをします。
*2　外部入力P.CNTと同じはたらきをします。

(6)割り込みユニット(AFP3452)

■I/O割り付け表

割り込みプログラムNo.は、装着スロットNo.の小さい方がINT0～INT7、大きい方がINT8～INT15となります。
割り込みユニットの割り込み入力接点と割り込みプログラムの関係は、各入力が割り込みプログラムと1:1に対応しています。
右図のように第0、第1スロットに割り込みユニットを装着すると以下のようになります。

例



割り込み入力番号	割り込みプログラムNo.	先着順位
X_&M00	INT 0	高い
X_&M01	INT 1	
X_&M02	INT 2	
X_&M03	INT 3	
X_&M04	INT 4	
X_&M05	INT 5	
X_&M06	INT 6	
X_&M07	INT 7	
X_&M10	INT 8	
X_&M11	INT 9	
X_&M12	INT 10	
X_&M13	INT 11	
X_&M14	INT12	
X_&M15	INT13	
X_&M16	INT14	
X_&M17	INT 15	低い

例えば入力 X_&M11に信号が入ると、INT 9の割り込みプログラムが起動します。また、同時に複数の条件が成立した場合の処理優先順位は、INT 0が最も高く、INT 15が最も低くなっています。

■割り込みユニットの装着制限について

割り込みユニットはCPUユニット1台につき2台まで装着できます。(3台以上装着すると、エラーとなりプログラムを実行することはできません。)
基本マザーボード/増設マザーボードのいずれに対しても任意の位置に装着可能です。

改訂履歴

マニュアル番号は、表紙下に記載されています。

発行日付	マニュアル番号	改訂内容
1994年11月	FAF−191	初版

ご注文に際してのお願い

本資料に記載された製品および仕様は、製品の改良などのために予告なしに変更(仕様変更、製造中止を含む)することがありますので、記載の製品のご使用のご検討やご注文に際しては、本資料に記載された情報が最新のものであることを、必要に応じ当社窓口までお問い合わせのうえ、ご確認下さいますようお願いします。

なお、本資料に記載された仕様や環境・条件の範囲を超えて使用される可能性のある場合、または記載のない条件や環境での使用、あるいは鉄道・航空・医療用などの安全機器や制御システムなど、特に高信頼性が要求される用途への使用をご検討の場合は、当社窓口へご相談いただき、仕様書の取り交わしをお願いします。

【受入検査】

・ご購入品または納入品につきましては、速やかに受入検査を行っていただくとともに、本製品の受入検査前または検査中の扱いにつきましては、管理保全に十分なご配慮をお願いします。

【保証期間】

・本製品の保証期間は、別途に両社間で定めのない限りは、ご購入後あるいは貴社のご指定場所への納入後1年間とさせていただきます。

【保証範囲】

・万一、保証期間中に本製品に当社側の責による故障や瑕疵が明らかになった場合、当社は代替品または必要な交換部品の提供、または瑕疵部分の交換、修理を、本製品のご購入あるいは納入場所で、無償で速やかに行わせていただきます。

ただし、故障や瑕疵が次の項目に該当する場合は、この保証の対象範囲から除かせていただくものとします。

(1) 貴社側が指示した仕様、規格、取扱い方法などに起因する場合。
(2) ご購入後あるいは納入後に行われた当社側が係わっていない構造、性能、仕様などの改変が原因の場合。
(3) ご購入後あるいは契約時に実用化されていた技術では予見することが不可能な現象に起因する場合。
(4) カタログや仕様書に記載されている条件・環境の範囲を逸脱して使用された場合。
(5) 本製品を貴社の機器に組み込んで使用される際、貴社の機器が業界の通念上備えられている機能、構造などを持っていれば回避できた損害の場合。
(6) 天災や不可抗力に起因する場合。

また、ここでいう保証は、ご購入または納入された本製品単体の保証に限るもので、本製品の故障や瑕疵から誘発される損害は除かせていただくものとします。

以上の内容は、日本国内の取引および使用を前提とするものです。
日本以外での取引および使用に関し、仕様、保証、サービスなどについてのご要望、ご質問は当社窓口まで別途ご相談ください。

●このマニュアルに使われている用紙は古紙配合率100%の再生紙を使用しております。
●この印刷物は環境にやさしい植物性大豆油インキを使用しています。

古紙配合率100%再生紙を使用しています　大豆油を主成分としたインキで印刷しています

●在庫・納期・価格など販売に関するお問い合わせは

●技術に関するお問い合わせは

制御機器コールセンター
0120-101-550
※お問い合せ商品 / リレー・機器用センサ・スイッチ・コネクタ・
プログラマブルコントローラ・プログラマブル表示器・
画像処理装置・タイマ・カウンタ・温度調節器
※サービス時間／9:00ー17:00（11:30ー13:00、当社休業日除く）
●FAX …… 06-6904-1573（24時間受付）

松下電工株式会社　制御機器本部
制御デバイス事業部
〒571-8686 大阪府門真市門真1048
TEL.(06)6908-1131〈大代表〉

このマニュアルの記載内容は平成6年11月現在のものです。

■FAF-191　199411-8Z　a　　Printed in Japan