SIEMENS

SINUMERIK

SINUMERIK 840D sl/828D
SINUMERIK Integrate Run
MyScreens (BE2)

プログラミングマニュアル

はじめに 1

導入ガイド 2

基礎事項 3

対話画面 4

変数 5

プログラミング命令 6

グラフィックおよびロジック項目 7

「Custom」操作エリア 8

対話画面選択 9

参照リスト A

適用:

CNCソフトウェア V4.5 SP2
SINUMERIK Operate V4.5 SP2

03/2013
A5E34416854

法律上の注意

警告事項

本書には、ユーザーの安全性を確保し製品の損傷を防止するうえ守るべき注意事項が記載されています。ユーザーの安全性に関する注意事項は、安全警告サインで強調表示されています。このサインは、物的損傷に関する注意事項には表示されません。以下に表示された注意事項は、危険度によって等級分けされています。

⚠危険
回避しなければ、直接的な死または重傷に至る危険状態を示します。

⚠警告
回避しなければ、死または重傷に至るおそれのある危険な状況を示します。

⚠注意
回避しなければ、軽度または中度の人身傷害を引き起こすおそれのある危険な状況を示します。

通知
回避しなければ、物的損傷を引き起こすおそれのある危険な状況を示します。

複数の危険レベルに相当する場合は、通常、最も危険度の高い事項が表示されることになっています。安全警告サイン付きの人身傷害に関する注意事項があれば、物的損傷に関する警告が付加されます。

有資格者

本書が対象とする製品 /
システムは必ず有資格者が取り扱うものとし、各操作内容に関連するドキュメント、特に安全上の注意及び警告が遵守されなければなりません。有資格者とは、訓練内容及び経験に基づきながら当該製品 /
システムの取り扱いに伴う危険性を認識し、発生し得る危害を事前に回避できる者をいいます。

シーメンス製品を正しくお使いいただくために

以下の事項に注意してください。

⚠警告
シーメンス製品は、カタログおよび付属の技術説明書の指示に従ってお使いください。他社の製品または部品との併用は、弊社の推奨もしくは許可がある場合に限ります。製品を正しく安全にご使用いただくには、適切な運搬、保管、組み立て、据え付け、配線、始動、操作、保守を行ってください。ご使用になる場所は、許容された範囲を必ず守ってください。付属の技術説明書に記述されている指示を遵守してください。

商標

®マークのついた称号はすべてSiemens AGの商標です。本書に記載するその他の称号は商標であり、第三者が自己の目的において使用した場合、所有者の権利を侵害することになります。

免責事項

本書のハードウェアおよびソフトウェアに関する記述と、実際の製品内容との一致については検証済みです。
しかしなお、本書の記述が実際の製品内容と異なる可能性もあり、完全な一致が保証されているわけではありません。 記載内容については定期的に検証し、訂正が必要な場合は次の版て更新いたします。

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
ドイツ

℗ 06/2014 変更する権利を留保

目次

1 はじめに 7

2 導入ガイド 11

2.1 はじめに 11

2.2 例 12
2.2.1 タスクの説明 12
2.2.2 設定ファイルの作成 15
2.2.3 OEMディレクトリへの設定ファイルの保存 18
2.2.4 オンラインヘルプの作成 19
2.2.5 OEMディレクトリへのオンラインヘルプの保存 20
2.2.6 OEMディレクトリへのeasyscreen.iniのコピー 21
2.2.7 easyscreen.iniへのCOMファイルの登録 21
2.2.8 プロジェクトのテスト 21

3 基礎事項 23

3.1 設定ファイルの構成 23

3.2 メニューツリーの構造 26

3.3 スタートソフトキーの定義 27
3.3.1 スタートソフトキーの機能 31

3.4 トラブルシューティング(ログブック) 33

3.5 作業者によるRun MyScreensへの切り替えに関する注意 34

4 対話画面 37

4.1 対話画面の構成と要素 37
4.1.1 対話画面の定義 37
4.1.2 対話画面プロパティの定義 40
4.1.3 対話画面要素の定義 45
4.1.4 例: 対話画面を開く 47
4.1.5 複数列のある対話画面の定義 48
4.1.6 表示イメージ/グラフィックの用途 49

4.2 ソフトキーメニューの定義 50
4.2.1 実行中のソフトキープロパティの変更 55
4.2.2 言語テキスト 57

4.3 オンラインヘルプの設定 59

5 変数 61

5.1 変数の定義 61

5.2 適用例 63
5.3 例1: 変数タイプ、テキスト、ヘルプ画面、色、ヒント欄の割り当て 65
5.4 例2: 変数タイプ、制限、属性、ショートテキスト位置のプロパティの割り当て 67
5.5 例3:
変数タイプ、初期設定、システム変数またはユーザー変数、入力/出力フィールド位置のプロパティの割り当て 68
5.6 切り替えフィールドとイメージの表示の例 69
5.7 変数パラメータ 71
5.8 変数タイプに関する詳細 75
5.9 切り替えフィールドに関する詳細 79
5.10 初期設定に関する詳細 80
5.11 ショートテキストの位置、入力/出力フィールドの位置に関する詳細 82
5.12 文字列の用途 83
5.13 CURPOS変数 85
5.14 CURVER変数 85
5.15 ENTRY変数 87
5.16 ERR変数 88
5.17 FILE_ERR変数 89
5.18 FOC変数 90
5.19 S_CHAN変数 91
6 プログラミング命令 93
6.1 演算子 93
6.1.1 算術演算子 93
6.1.2 ビット演算子 96
6.2 メソッド 98
6.2.1 CHANGE 99
6.2.2 FOCUS 101
6.2.3 LOAD 102
6.2.4 LOAD GRID 102
6.2.5 UNLOAD 104
6.2.6 OUTPUT 104
6.2.7 PRESS 105
6.2.8 OUTPUTブロックを使用したバージョン管理例 107
6.3 機能 109
6.3.1 ブロックの定義(//B) 110

6.3.2 サブプログラム呼び出し(CALL) 111
6.3.3 変数のチェック(CVAR) 112
6.3.4 プログラムファイルのコピー機能(CP) 114
6.3.5 プログラムファイルの削除機能(DP) 115
6.3.6 プログラムファイルの存在確認機能(EP) 117
6.3.7 プログラムファイルの移動機能(MP) 119
6.3.8 プログラムファイルの選択機能(SP) 120
6.3.9 対話画面行(DLGL) 121
6.3.10 評価(EVAL) 122
6.3.11 対話画面の終了(EXIT) 123
6.3.12 終了、ソフトキーの読み込み(EXITLS) 126
6.3.13 機能(FCT) 126
6.3.14 コードの作成(GC) 129
6.3.15 配列の読み込み(LA) 133
6.3.16 ブロックの読み込み(LB) 134
6.3.17 対話画面の読み込み(LM) 135
6.3.18 ソフトキーの読み込み(LS) 137
6.3.19 NC/PLCの読み出し(RNP)、NC/PLCの書き込み(WNP) 138
6.3.20 複数のNC PLC読み出し(MRNP) 140
6.3.21 レジスタ(REG) 143
6.3.22 RETURN 145
6.3.23 再コンパイル 146
6.3.24 コメントなしの再コンパイル 148
6.3.25 前方に検索、後方に検索(SF、SB) 152
6.3.26 STRING機能 153
6.3.27 PIサービス 158

7 グラフィックおよびロジック項目 161

7.1 線と長方形 161

7.2 配列の定義 162
7.2.1 配列要素の値へのアクセス 163
7.2.2 配列要素へのアクセス例 166
7.2.3 配列要素の状態のスキャン 168

7.3 テーブルグリッド(表) 169
7.3.1 テーブルグリッドの定義 171
7.3.2 列の定義 172
7.3.3 テーブルグリッドのフォーカス制御 174

7.4 カスタムウィジェット 175
7.4.1 カスタムウィジェットの定義 175
7.4.2 カスタムウィジェットライブラリの構造 176
7.4.3 カスタムウィジェットインタフェースの構造 177
7.4.4 カスタムウィジェットとダイアログの間の相互作用 179

8 「Custom」操作エリア 181

8.1 「Custom」操作エリアの有効化の方法 ... 181
8.2 「Custom」ソフトキーの設定方法 ... 181
8.3 「Custom」操作エリアの設定方法 ... 183
8.4 「Custom」エリアのプログラミング例 ... 185
9 対話画面選択 ... 191
9.1 PLCソフトキーを使った対話画面選択 ... 191
9.2 PLCハードキーを使った対話画面選択 ... 192
9.3 NCによる対話画面の選択 ... 196
A 参照リスト ... 197
A.1 スタートソフトキー一覧 ... 197
A.1.1 旋削用スタートソフトキー一覧 ... 197
A.1.2 フライス削り用スタートソフトキー一覧 ... 200
A.2 色のリスト ... 203
A.3 ファイル名称に使用される言語コードのリスト ... 204
A.4 アクセス可能なシステム変数のリスト ... 205
用語集 ... 207
索引 ... 211

はじめに

1

概要

「Run MyScreens」を使って、工作機械メーカまたはユーザーが設計した機能拡張を表示する操作画面を作成したり、独自のレイアウトを実装できます。
当社または工作機械メーカが提供する設定済みの操作画面を、変更または置き換えることができます。

たとえば、パートプログラムをユーザーが作成した操作画面上で編集できます。
コントロールシステムで対話画面を直接作成することができます。

「Run MyScreens」は、インタープリタ、および操作画面を記述する設定ファイルによって実装されます。

「Run MyScreens」はASCIIファイルを使用して設定されます。
この設定ファイルには操作画面の記述が含まれています。
このファイルの作成に使用する構文については、後述の章に記載されています。

基本設定

「Run MyScreens」機能によって、工作機械メーカは独自の対話画面を設定することができます。
基本設定でも、オペレータメニューツリーに、またはお客様専用のサイクル対話画面用に、5つの対話画面を設定することができます。

ソフトウェアオプション

対話画面数を増やすには、次のソフトウェアオプションが必要です。

「SINUMERIK Integrate Run MyScreens」(6FC5800-0AP64-0YB0)

必要条件

下記の条件を満たしてください。

- 1つの操作エリアの中の対話画面の間のみ切り替え可能。
- 対話画面でユーザー、設定およびマシンデータを処理できます。
- ユーザー変数はシステム変数またはPLC変数と同じ名称を持つことはできません(リストマニュアル システム変数 /PGAsl/も参照してください)。
- PLCによって起動される対話画面は別の操作エリアを構成する(計測サイクル画面に類似しています)。

ツール

- UTF8対応のエディタ(例: ワードパッド)
- グラフィック/表示イメージの作成には、画像処理プログラムが必要です。

用途

後述の機能を実装できます。

下記の要素を含む対話画面を表示します。	• ソフトキー • 変数 • テキストとヘルプテキスト • グラフィックとヘルプ表示
下記の操作で対話画面を開きます。	• (スタート)ソフトキーを押す • PLC/NCからの選択
ダイナミックに対話画面を再構成します。	• ソフトキーの編集と削除 • 変数表示欄の定義と設計 • (言語に依存するまたは、言語に依存しない)表示テキストの挿入、変更、削除 • グラフィックの挿入、変更、削除
下記の操作に応じて動作をおこないます。	• 対話画面の表示 • 値(変数)の入力 • ソフトキーの選択 • 対話画面の終了

対話画面間でデータを交信します。	
変数	• 読み出し(NC、PLC、ユーザー変数) • 書き込み(NC、PLC、ユーザー変数) • 算術演算子、比較演算子、論理演算子による結合
下記の機能を実行します。	• サブプログラム • ファイル機能 • PIサービス
ユーザークラスに応じてアクセスレベルを適用します。	

導入ガイド 2

2.1 はじめに

以下の例を使って、Run MySreensでSINUMERIK
Operateの操作画面に独自の対話画面を挿入するのに必要なステップを学習します。
また、独自の対話画面を作成する方法、状況に応じたヘルプ画面とヘルプの呼び出しの挿入、ソフトキーの定義と対話画面間の移動方法も学習します。

注記

NCUでHMI
Operateを使用する場合、CFカードのすべてのファイル名称は小文字で記載しなければならない(com、png、txt)ことに注意してください。 これは、Linuxの条件です。
PCUでは、ファイル名称に大文字と小文字のいずれも使用できます。

2.2 例

2.2.1 タスクの説明

対話画面

以下の例では2つの対話画面を作成します。
最初の対話画面では、(0と1が)書き込めるR変数およびジオメトリ軸の名称(入力欄)が表示されます。 対応するヘルプ画面は、この2つのR変数にリンクされます。
状況に応じたヘルプは、ジオメトリ軸にリンクされます。

対話画面1: R変数とチャネルMDの表示

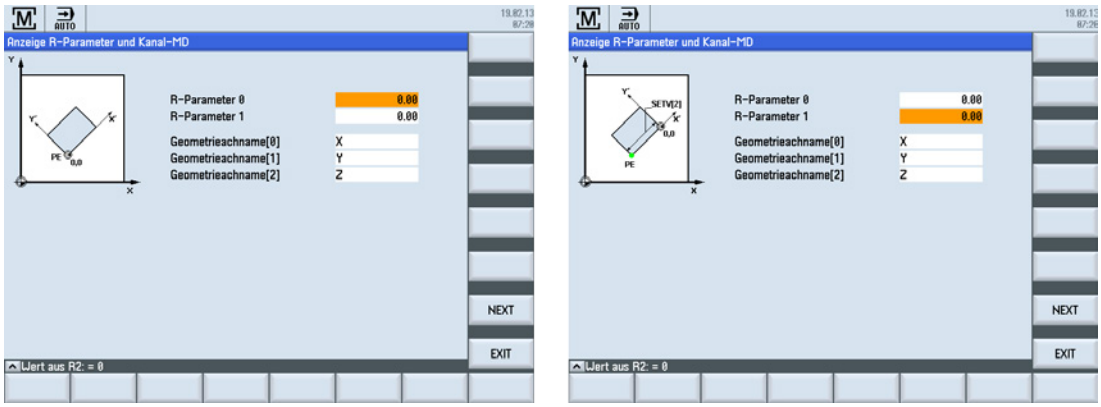


図 2-1 対話画面1: 状況に応じたヘルプ画面付きのR変数

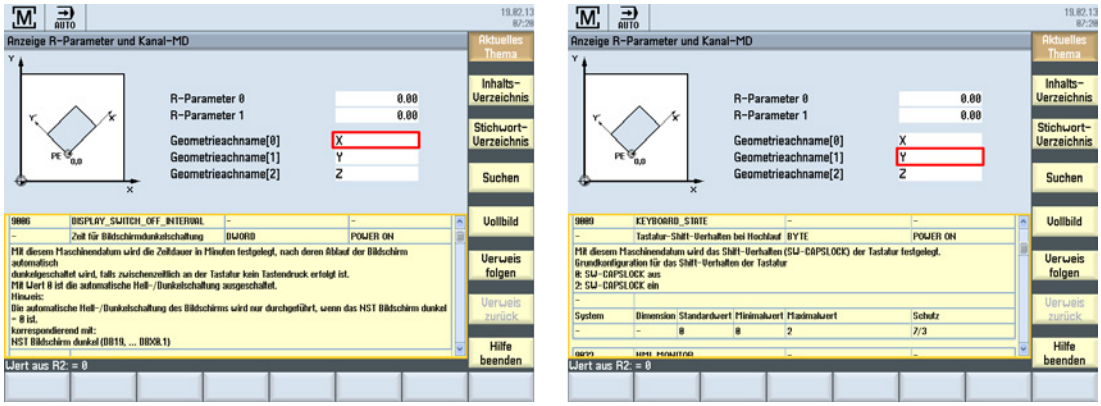


図 2-2 対話画面1: 状況に応じたオンラインヘルプ付きのジオメトリ軸名称

対話画面2: 軸値 WCSとMCS

MCSとWCS値は、2番目の対話画面に表示されます。

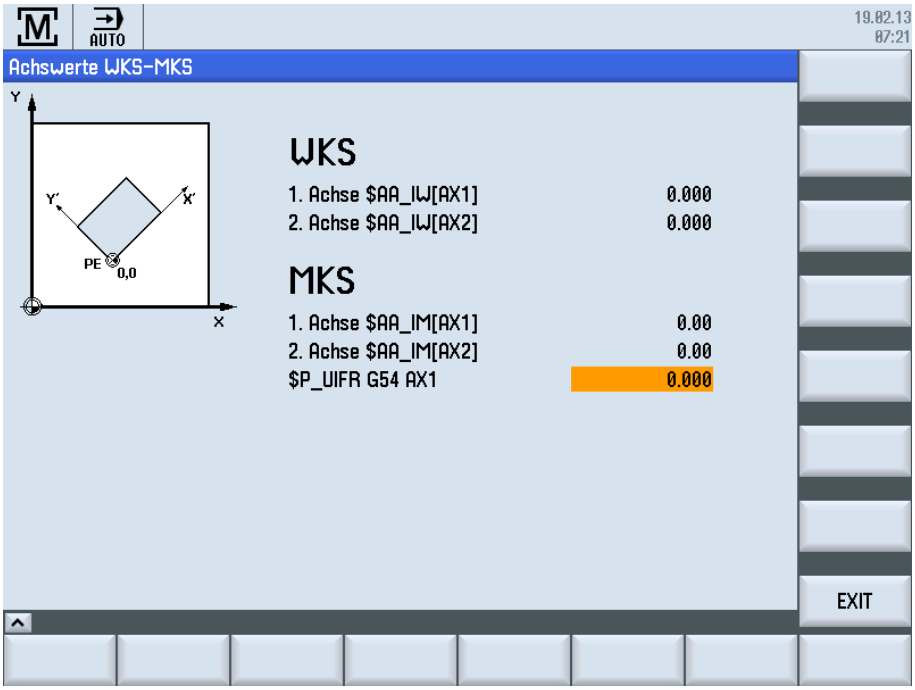


図 2-3　　対話画面2: 軸値の表示

ナビゲーション

最初の対話画面は、AUTOモードで機械操作エリアの[START]ソフトキーを使って呼び出します。 水平のSK6ソフトキーを使用します。

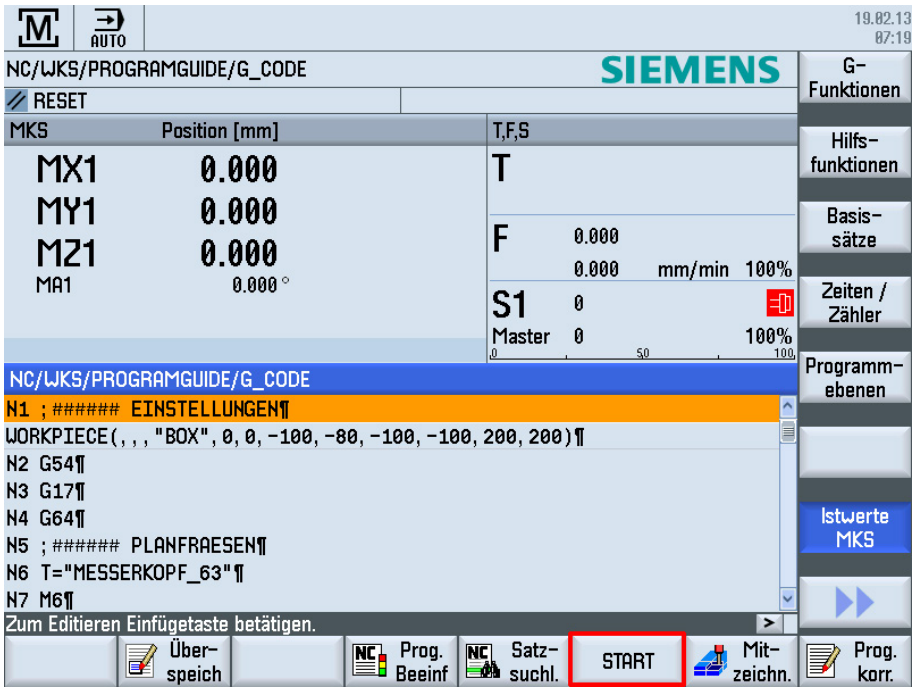


図 2-4 AUTOモードでの機械操作エリアのスタートソフトキー[„START“]

[NEXT]ソフトキーを使用して、1番目の対話画面から2番目の対話画面を呼び出すことができます。また、[EXIT]ソフトキーを使用して、操作エリアのルート画面に戻ることができます(上図参照)。
2番目の対話画面から[EXIT]ソフトキーを使って、操作エリアのルート画面に戻ることができます(上図参照)。

手順

必要な手順は、以下の章に記載されています。

1. 設定ファイル(COMファイル)の作成
2. OEMディレクトリへの設定ファイルの保存
3. オンラインヘルプの作成
4. OEMディレクトリへのオンラインヘルプの保存
5. OEMディレクトリへのeasyscreen.iniのコピー
6. easyscreen.iniへのCOMファイルの登録
7. プロジェクトのテスト

2.2.2 設定ファイルの作成

設定ファイルの内容

UTF8対応のエディタを使って、2つの対話画面の設定ファイルma_auto.com を作成します。

```
; 開始識別子、開始ソフトキー
//S(START)
 ; 開始ソフトキー、テキストのみ
 HS6=("START")
 ; 開始ソフトキーの言語テキストとpng
 ;HS6=([$80792,"\\sk_ok.png"])

 ; pressメソッド
 PRESS(HS6)
 ; LMまたはLS機能
 LM("Screen form1")
 ; LM、comファイル指定
 ;LM("screen form1","test.com")
 END_PRESS
; スタートソフトキーの終了識別子
//END

; 対話画面1のヘッダとR変数0の画面を定義
```

```
//M(Maske1/"Display R parameters and channel MD"/"mz961_01.png")
 ; 変数の定義
 def var1 = (R2///,"R parameter 0"///"$R[0]"/200,50,150/400,50,100,)
 ; ヘルプ画面付き
 def var2 = (R2///,"R parameter 1"//"mz961_02.png"/"$R[1]"/200,70,150/400,70,100)
 ; オンラインヘルプ付き
 def achs_nam1 = (s///,"geometry axis
name[0]"////200,100,150/400,100,100//"sinumerik_md_1.html","9006")
 ; オンラインヘルプ付き
 def achs_nam2 = (s///,"geometry axis
name[1]"////200,120,150/400,120,100//"sinumerik_md_1.html","9009")
 def achs_nam3 = (s///,"geometry axis name[2]"////200,140,150/400,140,100)

 ; 対話画面のソフトキー定義
 HS1=("")
 HS2=("")
 HS3=("")
 HS4=("")
 HS5=("")
 HS6=("")
 HS7=("")
 HS8=("")
 VS1=("")
 VS2=("")
 VS3=("")
 VS4=("")
 VS5=("")
 VS6=("")
 VS7=("NEXT")
 VS8=("EXIT")

 ; LOADブロックの定義
 LOAD
 ; RNPによる値の読み取り
 ACHS_NAM1 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[0]")
 ACHS_NAM2 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[1]")
 ACHS_NAM3 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[2]")
 ; 対話画面行の出力
 DLGL("Value from R2: = " << RNP("$R[2]"))
 ; WNPの値への書き込み
 WNP("$R[3]",var0)
```

```
 END_LOAD

 ; pressメソッド
 press(vs7)
 ; 追加対話画面のロード
 LM("Screen form2")
 ; pressメソッドの終了識別子
 end_press

 ; pressメソッド
 PRESS(VS8)
 ; 対話画面の終了
 EXIT
 ; pressメソッドの終了識別子
 end_press
; 対話画面1の終了識別子
//END

; 対話画面2のヘッダと画面を定義
//M(Maske2/「Axis values WCS-MCS」/「mz961_01.png」)
 ; 変数の定義
 def TEXT1 = (I///,"WKS"/WR0,fs2///200,30,120/,,1)
 def Var1 = (R3///,"1st axis $AA_IW[AX1]"/WR1//"$AA_IW[AX1]"/200,70,150/400,70,100)
 def Var2 = (R3///,"2nd axis $AA_IW[AX2]"/WR1//"$AA_IW[AX2]"/200,90,150/400,90,100)
 def TEXT2 = (I///,"MCS"/WR0,fs2///200,120,120/,,1)
 def var3 = (R2///,"1st axis $AA_IM[AX1]"/WR1//"$AA_IM[AX1]"/200,160,150/400,160,100)
 def var4 = (R2///,"2nd axis $AA_IM[AX2]"/WR1//"$AA_IM[AX2]"/200,180,150/400,180,100)
 def var5 = (R3///,"$P_UIFR G54 AX1"///"$P_UIFR[1,AX1,TR]"/200,200,150/400,200,100)

 ; 対話画面のソフトキー定義
 HS1=("")
 HS2=("")
 HS3=("")
 HS4=("")
 HS5=("")
 HS6=("")
 HS7=("")
 HS8=("")
 VS1=("")
 VS2=("")
 VS3=("")
```

```
VS4=("")
VS5=("")
VS6=("")
VS7=("")
VS8=("EXIT")

; Changeメソッド
CHANGE(var5)
; PI_START機能
pi_start("/NC,201,_N_SETUFR")
; changeメソッド終了識別子
END_CHANGE

; pressメソッド
press(recall)
; 呼び出し画面に戻る
LM("Screen form1")
; pressメソッドの終了識別子
end_press

; pressメソッド
PRESS(VS8)
; 対話画面を終了し標準アプリケーションに復帰
exit
; pressメソッドの終了識別子
END_PRESS
; 対話画面終了識別子
//END
```

2.2.3 OEMディレクトリへの設定ファイルの保存

保存パス

設定ファイルma_auto.comを次のパスの下に保存します。

```
card/oem/sinumerik/hmi/proj
```

2.2.4 オンラインヘルプの作成

オンラインヘルプの内容

HTMLファイルsinumerik_md_1.htmlを作成します。
name="9006"とname=9009"で呼び出しをおこないます。
本マニュアルではオンラインヘルプの作成について扱わないため、ヘルプファイルについてはコメントしません。
オンラインヘルプの組み込みについての注釈は、「オンラインヘルプの設定 (ページ 59)」の章にあります。

```
<html><head><meta http-equiv="Content-Type" content="text/html; charset="UTF-
8"/><title></title></head>
   <body>
      <table border="1"  cellspacing="0" cellpadding="2" width="100%"><tr><td><a
name="9006"><b>9006</b></a></td>
         <td colspan="3">DISPLAY_SWITCH_OFF_INTERVAL</td>
         <td>-</td>
         <td>-</td></tr>
         <tr><td>-</td>
         <td colspan="3">スクリーンセーバーの時間</td>
         <td>DWORD</td>
         <td>電源投入</td></tr>
         <tr>
         <td
colspan="6">このマシンデータを使って、継続してキーボードのキーが押されない場合、自動的にスクリーンセーバーが起動するまでの<br />
時間を分で定義します。<br />
値が0の場合、自動スクリーンセーバーは無効になります。<br />
注:<br />
自動スクリーンセーバーは、NST screen dark = 0の場合のみ有効化されます。<br />
これは次に対応します:<br />
NST screen dark (DB19, ... DBX0.1)</td>
         </tr>
         <tr><td>-</td><td colspan="5"> </td>
         </tr>
         <tr><td width="16*">システム</td><td width="16*">寸法</td><td
width="16*">初期値</td><td width="16*">最小値</td><td width="16*">最大値</td><td
width="16*">保護</td></tr><tr><td>-</td><td>
-</td>
         <td>60</td>
         <td>0</td>
         <td>180</td>
         <td>7/3</td></tr>
      </table><p></p>


      <table border="1"  cellspacing="0" cellpadding="2" width="100%"><tr><td><a
name="9009"><b>9009</b></a></td>
         <td colspan="3">KEYBOARD_STATE</td>
```

```
          <td>-</td>
          <td>-</td></tr>
          <tr><td>-</td>
          <td colspan="3">電源投入時のキーボードシフト動作</td>
          <td>BYTE</td>
          <td>電源投入</td></tr>
          <tr>
     <td colspan="3">このマシンデータは、シフト動作(SW-CAPSLOCK)を定義するために使用します。<br />
キーボードシフト動作の基本設定<br />
 0: SW-CAPSLOCK off<br />
 2: SW-CAPSLOCK ein</td></tr>
          <tr><td>-</td><td colspan="5"> </td>
          </tr>
          <tr><td width="16*">システム</td><td width="16*">寸法</td><td
width="16*">標準値</td><td width="16*">最小値</td><td width="16*">最大値</td><td
width="16*">保護</td></tr><tr><td>-</td><td>
-</td>
          <td>0</td>
          <td>0</td>
          <td>2</td>
          <td>7/3</td></tr>

....

</body></html>
```

2.2.5 OEMディレクトリへのオンラインヘルプの保存

保存パス

ドイツ語のヘルプの場合は、HTMLファイルsinumerik_md_1.htmlを以下のパスに保存します。

```
/card/oem/sinumerik/hmi/hlp/deu
```

追加言語(例: chs、eng、esp、fra、ita …)の場合は、フォルダを作成してください。

言語コードのリストは、付録にあります。

2.2.6 OEMディレクトリへのeasyscreen.iniのコピー

保存パス

ファアイル easyscreen.iniを以下のディレクトリからコピーします。

```
card/siemens/sinumerik/hmi/cfg
```

次のディレクトリにコピーします。

```
card/oem/sinumerik/hmi/cfg
```

2.2.7 easyscreen.iniへのCOMファイルの登録

easyscreen.iniでの適合

OEMディレクトリのeasyscreen.iniで以下の変更をおこないます。
これを実行することで、ma_auto.com設定ファイルを登録します。

```
[STARTFILES]

;<=============================================================
>

;< 水平のMachine-AutomaticメニューのOEMソフトキー                 >

;< SOFTKEY位置="6"                                               >

;<=============================================================
>

StartFile03 = area := AreaMachine, dialog := SlMachine, menu :=
SlMaAutoMenuHU, startfile := ma_auto.com
```

2.2.8 プロジェクトのテスト

対話画面呼び出しのテスト

機械操作エリアで、AUTOモードに変更します。
[START]ソフトキーをクリックします。

設定ファイルの解釈時に「Run
MyScreens」が異常を検出した場合、これらの異常は、easyscreen_log.txt

ASCIIファイルに書き込まれます(「トラブルシューティング(ログブック) (ページ 33)」の章を参照してください)。

状況に応じたヘルプ画面とオンラインヘルプのテスト

状況に応じたヘルプ画面とオンラインヘルプのテストします。
異常が発生した場合、ファイルが保存されているパスをチェックします。

基礎事項 3

3.1 設定ファイルの構成

はじめに

設定ファイルには新しい操作画面の定義データを設定します。
このファイルは自動的に解釈され、結果が画面に表示されます。
設定ファイルは提供されたソフトウェアには設定されていませんので、ユーザーが準備してください。

ASCIIエディタ(例:
Notepadまたは操作画面に組み込まれたエディタ)を使って、設定ファイルを作成します。 コメントを使用して、説明を含めることもできます。
「;」は、すべての説明の前にコメント文字として挿入されます。

注記

COMファイルは、必ずASCIIファイルとして保存することもできます。
ソフトキーラベルなどのテキストをテキストIDを使用するのではなく(言語ファイルから読み出すため)、設定に直接入力した場合、COMファイルはUTF-8で保存してください。こうしないと、テキストが正しく表示されません。
ただし、キーワード(これもUTF8エンコードのCOMファイルにあります)は、ASCII文字セットに含まれる文字のみを使用してください。
これを守らない場合、解釈、および画面/対話画面の表示は保証されません。
言語ファイルはテキストのみを含むため、必ずUTF-8で保存します。

注記

NCUでHMI
Operateを使用する場合、CFカードのすべてのファイル名称は小文字で記載しなければならない(com、png、txt)ことに注意してください。 これは、Linuxの条件です。
PCUでは、ファイル名称に大文字と小文字のいずれも使用できます。

設定

各HMIアプリケーションには固定スタートソフトキーがあり、それを使用して新しく作成した対話画面へアクセスできます。

設定ファイルで「画面の読み込み」(LM)または「ソフトキーメニューの読み込み」(LS)が呼び出される際に、呼び出されるオブジェクトを含んだ新しいファイル名称を指定することができます。
これにより、設定を構成することができます(たとえば、独立した設定ファイルの、1つの操作レベルのすべての機能)。

設定ファイルの構造

設定ファイルは下記の要素から構成されます。

1. スタートソフトキーの記述
2. 対話画面の定義
3. 変数の定義
4. ブロックの記述
5. ソフトキーメニューの定義

注記

手順

設定ファイルの指定された手順は、守ってください。

例:

```
//S (START)                    ; startソフトキーの定義(オプション)
....
//END
//M (.....)                    ; 対話画面の定義
DEF .....                      ; 変数の定義
LOAD                           ; ブロックの記述
...
END_LOAD
UNLOAD
...
END_UNLOAD
...
//END
//S (...)                      ; ソフトキーメニューの定義
//END
```

設定ファイルの保存先

設定ファイルは、/user/sinumerik/hmi/proj
ディレクトリにあり、これに応じてadd_onとoemディレクトリにも保存されます。

他のHMIアプリケーションのテキストの変換

コードページコーディングを備えたテキストファイルをテキストコーディングUTF-8に変換する手順

1. PG/PCのテキストファイルをテキストエディタで開きます。
2. 保存時には、UTF-8エンコードを設定します。

コードページコードを介した読み取りメカニズムは依然としてサポートされています。

3.2 メニューツリーの構造

メニューツリーの原則

相互にリンクした複数の対話画面がメニューツリーを作成します。
ある対話画面から別の対話画面に切り替えることができた場合、リンクが形成されています。
この対話画面で新たに定義された水平/垂直ソフトキーを使用して、前の対話画面、またはその他の対話画面を呼び出すことができます。

メニューツリーを各スタートソフトキーの後に作成することができます。

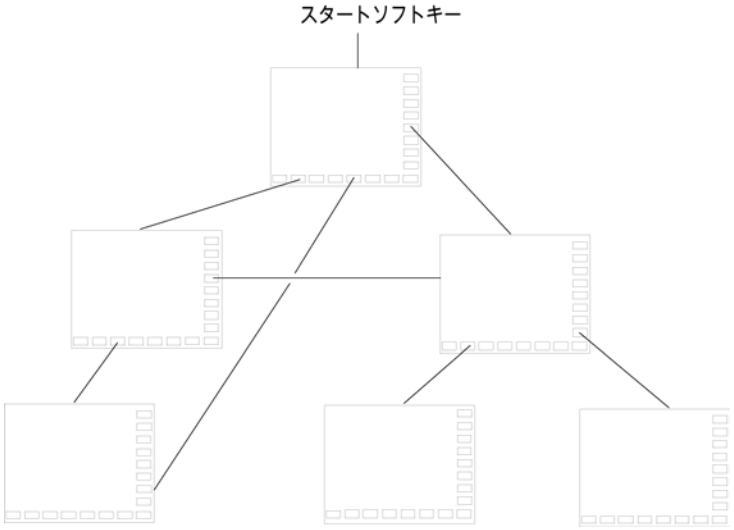


図 3-1 メニューツリー

スタートソフトキー

独自の動作処理を開始するために使用する複数のソフトキー(スタートソフトキー)は、easyscreen.iniで指定された設定ファイルで定義します。

専用の対話画面の読み込みはソフトキー定義または他のソフトキーメニューと関連しています。 これらを使用して、以降の動作を実行します。

スタートソフトキーを押すと、割り当てられた対話画面が読み込まれます。
これによって、この対話画面関連のソフトキーも有効になります。
特定の位置が設定されていなければ、変数は標準位置に出力されます。

標準アプリケーションへの復帰

新たに作成した操作画面を終了して、標準アプリケーションに戻ることができます。

[RECALL]キーを他の処理に設定していなければ、このキーを使用して新しい操作画面を閉じることができます。

注記

PLCユーザープログラムを使用した対話画面の呼び出し

対話画面はソフトキーと同様にPLCから選択できます。
インタフェース信号として、PLCとHMIの間の信号交信用にDB19.DBB10を使用できます。

3.3 スタートソフトキーの定義

対話画面に依存しないソフトキー

スタートソフトキーは対話画面からは呼び出されない、対話画面に依存しないソフトキーですが、新しい最初の対話画面の**前に**設定されています。
対話開始画面またはスタートソフトキーメニューにアクセスするには、スタートソフトキーを定義してください。

プログラミング

スタートソフトキーの定義ブロックの構成は次の通りです。

```
//S(Start)                ; スタートソフトキーの開始識別子
HS1=(...)                 ; スタートソフトキーの定義: 水平SK 1
PRESS(HS1)                ; メソッド
  LM...                   ; LMまたはLS機能
END_PRESS                 ; メソッドの終了
//END                     ; スタートソフトキーの終了識別子
```

使用できるスタートソフトキーの位置

操作エリアでは下記の「Run MyScreens」のスタートソフトキー位置が使用できます。

操作エリア	位置
マシン	HSK6
パラメータ	HSK7
プログラム	HSK6、HSK15 計測サイクル: HSK13、HSK14
プログラムマネージャ	HSK2-8、HSK12-16, ドライブ装置に割り当てられていない場合のみ使用できます。
診断	HSK7
セットアップ	HSK7

スタートソフトキーは特別なファイルに設定します。
このファイル名称はeasyscreen.iniファイルに宣言されています。
通常、操作エリア固有の名称になっています(たとえば、スタートアップエリアの場合、startup.com等です)。
運転モード固有のファイル(ma_jog.com、ma_auto.com)がある運転操作エリアには適用されません。

スタートソフトキーのあるソフトキーメニューは[Start]で呼び出します。
スタートソフトキーの既存の設定は引き続き使用することができます。
スタートソフトキーメニューのなかのスタートソフトキーが各HMIアプリケーション(操作エリア)のソフトキーと統合される機能はサポートされていません。
つまり、最初の対話画面呼び出しがおこなわれるまで、言い換えると、全機能を使用できる(たとえば、PRESSブロックの実行)ときまで、メニューまたはソフトキーメニューを他の全てと置き換えることができません。

標準アプリケーションのメニューには、XML設定の一部として「easyscreenmode」メニュープロパティが付いています。
これは、当該メニューが「Run MyScreens」のスタートソフトキーの使用を許可する(= easyscreen)か許可しない(= off)かを示します。

```
<SCREEN name="SlEasyScreenTest">
  <FORM ... >
      <PROPERTY ... > ... </PROPERTY>
  </FORM>
    <MENU name="menu_horiz" softkeybar="hu" easyscreenmode="easyscreen" />
    <MENU name="menu_vert" softkeybar="vr" easyscreenmode="off" />
</SCREEN>
```

例

個別のスタートソフトキーメニューを水平メニューと垂直メニュー用に定義することができます。 このために、「MENU」属性を使用します。

新しいメニューがHMIアプリケーションに表示され、このメニューが設定(easyscreenmode = "easyscreen")に応じてスタートソフトキーの使用を許可する場合、まずスタートソフトキーメニューの設定の「MENU」属性に対して検索されます。

- 「MENU」属性を備えたスタートソフトキーメニューの設定が検出され、「MENU」属性に現在表示されているメニューの名称が含まれている場合(例:では: "menu_horiz")、このスタートソフトキーメニューが表示されます。

  [menu_horiz]メニューは水平メニューバーを含むので、ここでは水平ソフトキーのみ考慮されます。

- 特定のメニュー用のメニュー専用のソフトキーメニューがない(すなわち「MENU」属性を使用できない)場合、初期設定のスタートソフトキーメニューが読み込まれます。

```
//S(Start)
  MENU="menu_horiz"
  HS2=("Contour",ac6,se3)
    PRESS(HS2)
      LS("Contour")
    END_PRESS
  ...
//END
```

設定用のテンプレート

スタートソフトキーのすべての許容可能な位置と、それらの設定の詳細な説明は、次のディレクトリのeasyscreen.iniファイルに存在します。

```
/card/siemens/sinumerik/hmi/cfg
```

このファイルは、ユーザー自身の設定用テンプレートとして使用されます。

下記を参照してください。

スタートソフトキーのリスト (ページ 197)

3.3.1 スタートソフトキーの機能

対話画面に依存しないソフトキーの機能

特定の機能のみスタートソフトキーで実行できます。

下記の機能が使用できます。

- **LM機能**は他の対話画面を読み込むために使用します。 **LM***("識別子"[,"ファイル"])*
- **LS機能**を使用して、他のソフトキーメニューを表示することができます。
  LS*("識別子"[, "ファイル"][, マージ])*
- **「EXIT」機能**を使用して、新たに設定した操作画面を終了して、標準アプリケーションに戻ることができます。
- **「EXITLS」機能**を使用して、現在の操作画面を終了して、定義されたソフトキーメニューを読み込むことができます。

PRESSメソッド

ソフトキーは定義ブロックの中で定義され、「LM」または「LS」の機能はPRESSメソッドで割り当てられます。

スタートソフトキー定義がコメントに記述されるか(行の先頭のセミコロン(;))、または設定ファイルが削除された場合、スタートソフトキーは機能しません。

```
//S(Start)                          ; 開始識別子
HS6=("1st screen form")             ; 水平SK 6は[1st screen form]と表記
PRESS(HS6)                          ; 水平SK 6用のPRESSメソッド
  LM("Screen form1")                ; Screen form1機能を読み込みます。Screen
                                    form1は同じこのファイルの中で定義してください。
END_PRESS                           ; PRESSメソッドの終了
HS7=("2nd screen form")             ; 水平SK 7は[2nd screen form]と表記
PRESS(HS7)                          ; 水平SK 7用のPRESSメソッド
  LM("Screen form2")                ; Screen form2機能を読み込みます。Screen
                                    form2は同じこのファイルの中で定義してください。
END_PRESS                           ; PRESSメソッドの終了
//END                               ; 起動ブロックの終了識別子
```

例

```
HS1 = ("new softkey menu")
HS2=("no function")
PRESS(HS1)
  LS("Menu1")                          ; 新しいソフトキーバーを読み込みます。
END_PRESS
PRESS (HS2)                            ; 空のPRESSメソッドです。
END_PRESS
```

スタートソフトキーの各種設定

スタートソフトキーの各種設定が統合されます。
この場合、解釈するファイルの名称がまずeasyscreen.iniから読み出されます。comの拡張子が付いたファイルの検索が下記のディレクトリでおこなわれます。

- /user/sinumerik/hmi/proj/
- /oem/sinumerik/hmi/proj/
- /addon/sinumerik/hmi/proj/
- /siemens/sinumerik/hmi/proj/

スタートソフトキー用の複数の設定が統合されて1つの設定が構成されます。すなわち、個々のソフトキーが比較されます。
1つのソフトキーに対して2つ以上の設定がある場合、常に上位の順番が統合バージョンに採用されます。

複数のソフトキーメニューまたは対話画面が含まれている可能性があれば、これは無視されます。 ソフトキーにファイル情報を含まない命令(たとえば、LM ("test"))がある場合でも、必要なソフトキーメニュー、または対話画面が同じファイルに含まれるため、対応するファイル名称は内部統合バージョンで追加され、変更の必要はありません。 その後、統合設定を含めて表示します。

3.4 トラブルシューティング(ログブック)

概要

設定ファイルの解釈時に「Run MyScreens」が異常を検出すると、これらの異常はASCIIファイルのeasyscreen_log.txtに書き込まれます。
このファイルは操作画面が再起動する毎に削除されます。

このファイルは下記の内容を示します。

- エラーが発生したときの動作
- 問題のある最初の文字の行と列の番号
- 設定ファイルの問題のある行全体

easyscreen-log.txtの保存

easyscreen_log.txtファイルは下記のディレクトリに保存されます。

```
/user/sinumerik/hmi/log/
```

構文

スタートソフトキーが定義され、定義行と、開始と終了の識別子を含む対話画面が設定されるまで、システムは構文の解釈を開始しません。

```
//S(Start)
HS6=("1st screen")
PRESS(HS6)
  LM("Maske1")
END_PRESS
//END

//M(Maske1)
  DEF Var1=(R)
 DEF VAR2 = (R)
 LOAD
 VAR1 = VAR2 + 1                ;
                                VAR2に値がなかったため、ログブック内にエラーメッセージがあります。
...
```

```
//END


;
適切な入力は、たとえば以下のように
なります。
//M(Maske1)
 DEF Var1=(R)
 DEF VAR2 = (R)

 LOAD
 VAR2 = 7
 VAR1 = VAR2 + 1 ;
...
```

3.5 作業者によるRun MyScreensへの切り替えに関する注意

注記

NCUでHMI
Operateを使用する場合、CFカードにすべてのファイル名称が小文字で保存される(com、png、txt)ことに注意してください。

画像ファイル

画像ファイルは、PNG形式(xxx.png)で保存してください。

たとえばOEM修正のデータは、以下に保存してください。

/oem/sinumerik/hmi/ico/[解像度]

設定ファイルの適用

設定ファイルの以下のポイントをチェックします。

- スタートソフトキーと現在許容されているソフトキー(付録のスタートソフトキーのリストを参照してください)と比較し、必要に応じてそれらを設定します。
- 上記の「画像ファイル」のポイントに対応して、組み込まれた画像ファイルの名称を変更します。

たとえばOEM修正のデータは、以下に保存してください。

```
/oem/sinumerik/hmi/proj
```

ヘルプファイルの設定

すべてのヘルプファイルはUTF8で保存してください。
既存のファイルをチェックし、適切なエディタを使いこれに従って再保存します。

HTMLファイルは、たとえばドイツ語の場合、以下のディレクトリに保存されます。

```
/card/user/sinumerik/hmi/hlp/deu
/card/oem/sinumerik/hmi/hlp/deu
/card/addon/sinumerik/hmi/hlp/deu
```

言語識別子に対応する追加言語のディレクトリを作成してください(付録を参照)。

Run MyScreensライセンスのチェック

入力した対話画面の数が 基本の最大5つの対話画面を超えていないかチェックします。

対話画面の数を追加するには、以下のソフトウェアオプションが必要です。
「SINUMERIK Integrate Run MyScreens」(6FC5800-0AP64-0YB0)

4 対話画面

4.1 対話画面の構成と要素

4.1.1 対話画面の定義

定義

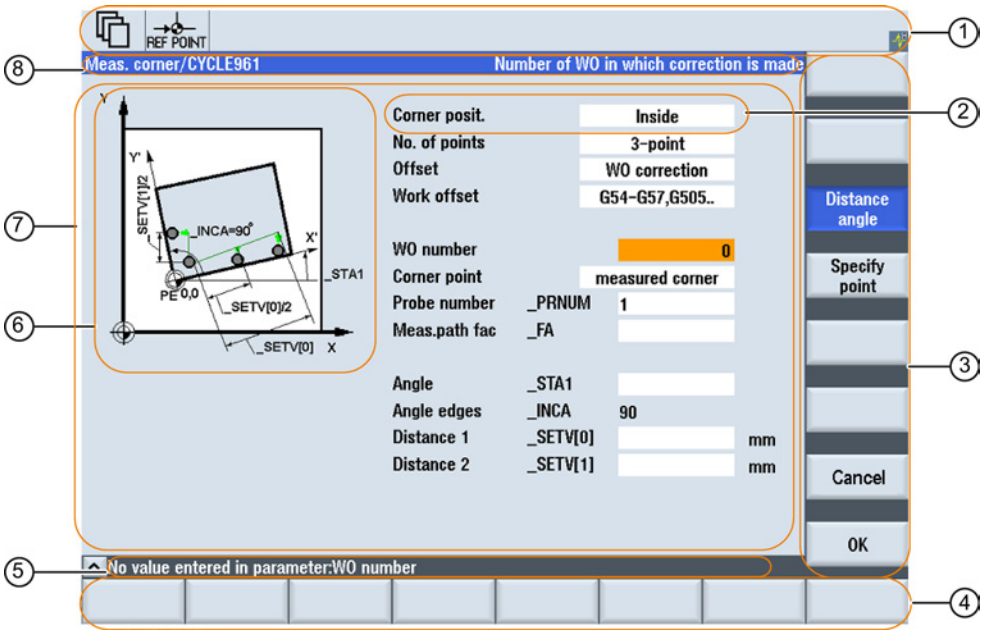
対話画面は操作画面の一部であり、表示行、対話画面要素かグラフィックまたは両方、メッセージの出力行、8個の水平ソフトキーと8個の垂直ソフトキーから構成されます。

対話画面要素には下記の項目があります。

- 変数
  - 制限/切り替えフィールド
  - 変数の初期設定
- ヘルプ表示
- テキスト
- 属性
- システム変数またはユーザー変数
- ショートテキストの位置
- 入力/出力フィールドの位置
- 色

対話画面のプロパティ

- ヘッダー
- グラフィック
- 寸法
- システム変数またはユーザー変数
- グラフィックの位置
- 属性



① 機械の状態表示(「ヘッダー」)
② 対話画面要素
③ 8個の垂直ソフトキー
④ 8個の水平ソフトキー
⑤ 診断メッセージの出力
⑥ グラフィック
⑦ 対話画面
⑧ ヘッダーとロングテキストのある対話画面のヘッダー行

図 4-1 対話画面の構成

概要

対話画面の定義(定義ブロック)の基本構成は次の通りです。

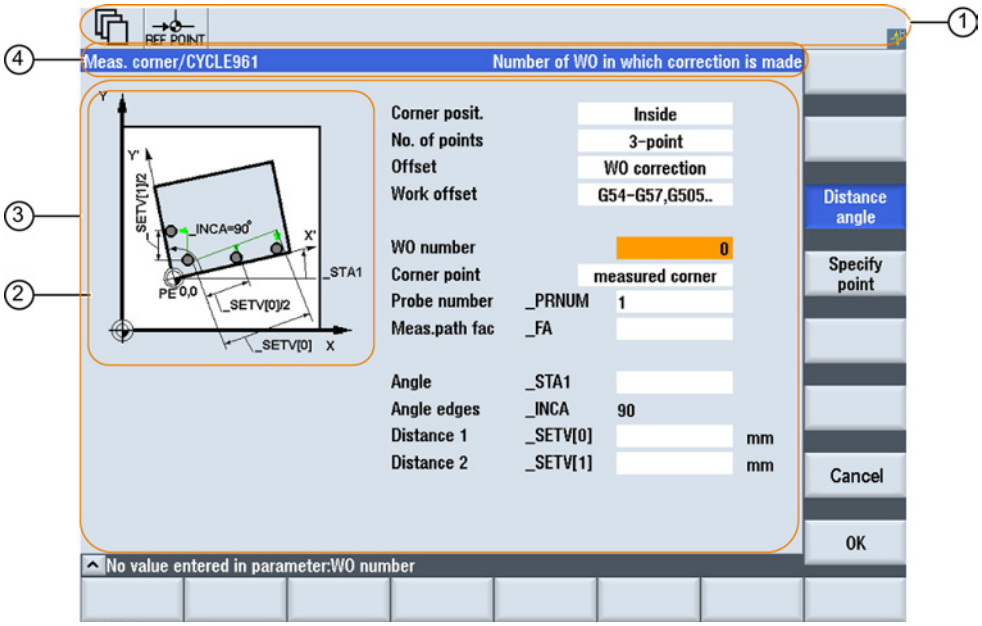
定義ブロック	コメント	参照する章
//M...	; 対話画面の開始識別子	
DEF Var1=... ...	; 変数	「変数」の章を参照してください
HS1=(...) ...	; ソフトキー	「ソフトキーメニューの定義」の章を参照してください
PRESS(HS1) LM... END_PRESS	; メソッド開始識別子 ; 動作 ; メソッド終了識別子	「メソッド」の章を参照してください
//END	; 対話画面終了識別子	

対話画面定義ブロックの中では、対話画面に対話画面要素として表示される各種変数、水平ソフトキー、垂直ソフトキーが最初に定義されます。
続いて各種動作がメソッドに設定されます。

4.1.2 対話画面プロパティの定義

概要

対話画面のプロパティは対話画面の識別子の開始行に定義します。



① 機械の状態表示(「ヘッダー」)
② グラフィック
③ 対話画面
④ ヘッダーとロングテキストのある対話画面のヘッダー行

図 4-2 対話画面のプロパティ

プログラミング

構文:	//M(識別子/[ヘッダー]/[グラフィック]/[寸法]/[システム変数またはユーザー変数]/[グラフィック位置]/[属性])	
説明:	対話画面を定義します。	
パラメータ:	識別子	対話画面の名称
	ヘッダー	テキストとしての対話画面ヘッダー、または言語テキストファイルからのテキスト($85011等)の呼び出し
	グラフィック	ダブルクォーテーションマークで囲まれたパスを含むグラフィックファイル
	寸法	ピクセル単位の対話画面の位置とサイズ(左側からの距離、右側からの距離、幅、高さ)です。画面左上のコーナーを基準とします。 項目はコンマで区切られます。
	システム変数またはユーザー変数	現在のカーソル位置が割り当てられるシステム変数またはユーザー変数です。 システム変数またはユーザー変数によって、NCまたはPLCへカーソル位置が与えられます。 最初の変数はインデックス1です。この順序は変数の設定順序と一致します。
	グラフィックの位置	ピクセル単位のグラフィックの位置(左側からの距離、右側からの距離)です。対話画面左上のコーナーを基準とします。 上部からの最小すきまは18ピクセルです。 項目はコンマで区切られます。
	属性	属性の指定はコンマで区切られます。 指定可能な属性は次の通りです。
	CMx	列モード: 列の配置
	CM0	初期設定: 列の配置は行毎に個別におこなわれます。
	CM1	最も多くの列を含んだ行の列配置が、すべての行に適用されます。

	CB	CHANGEブロック: 対話画面を開いたときの動作: 変数定義で変数に指定されたcb属性は、対話画面定義での初期設定より優先されます。
	CB0	初期設定: 対話画面が開くと、対話画面に関連したすべてのCHANGEブロックが処理されます。
	CB1	関連する値が変更される場合のみ、CHANGEブロックが処理されます。

対話画面プロパティへのアクセス

下記の対話画面プロパティへの読み出しアクセスと書き込みアクセスはメソッドの中で許可されます(PRESSブロック等)。

- Hd = ヘッダー
- Hlp = ヘルプ表示
- Var = システム変数またはユーザー変数

例

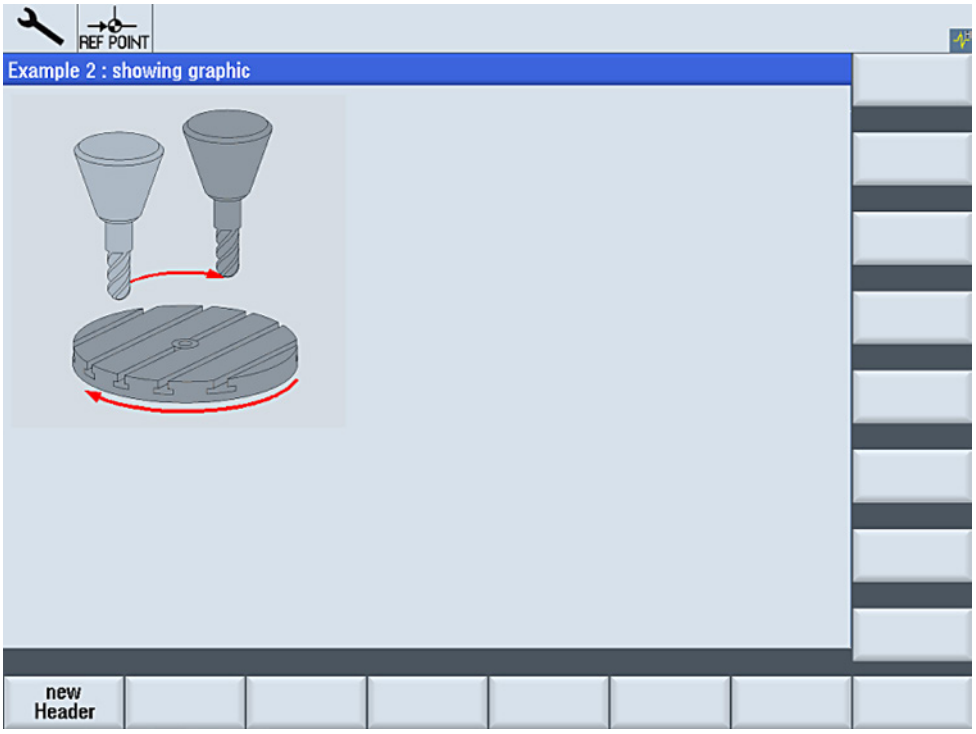


図 4-3 "Example 2: showing graphic"

```
//S(Start)
HS7=("Example", se1, ac7)

PRESS(HS7)
  LM("Mask2")
END_PRESS

//END
//M(Mask2/"Example 2 : showing graphic"/"example.png")
HS1=("new%nHeader")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
```

```
VS1=("")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("")
VS8=("")

PRESS(HS1)
  Hd= "new Header"
END_PRESS
...
//END
```

下記も参照

「Custom」エリアのプログラミング例 (ページ 185)

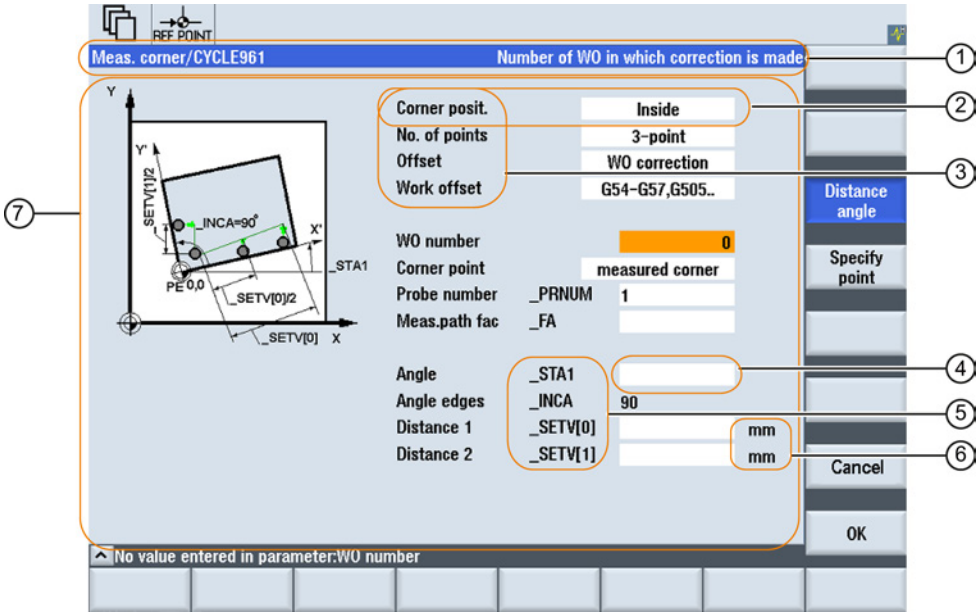
4.1.3 対話画面要素の定義

対話画面要素

「対話画面要素」とは変数の表示部分、すなわちショートテキスト、グラフィックテキスト、入力/出力フィールド、単位テキストのことです。
対話画面要素は対話画面の本体の行にあります。
各行に1つ以上の対話画面要素を定義できます。

変数のプロパティ

すべての変数は動作中の対話画面でのみ有効です。
変数が定義されるときにプロパティが変数に割り当てられます。
対話画面プロパティの値はメソッドの中でのみアクセスできます(PRESSブロック等)。



① ヘッダーとロングテキストのある対話画面のヘッダー行
② 対話画面要素
③ ショートテキスト
④ 入力/出力フィールド
⑤ グラフィックテキスト
⑥ 単位のテキスト
⑦ 対話画面の本体

図 4-4 対話画面要素

プログラミング - 一覧

コンマで区切られる単独のパラメータは丸括弧で囲んでいます。

DEF *識別子* =	**識別子 = 変数の名称**
	変数タイプ
	/[制限値または切り替えフィールド]
	/[初期設定]
	/[テキスト(ロングテキスト, ショートテキスト|イメージ, グラフィックテキスト, 単位テキスト)]
	/[属性]
	/[ヘルプ表示]
	/[システム変数またはユーザー変数]
	/[ショートテキストの位置]
	/[I/Oフィールドの位置(左, 上, 幅, 高さ)]
	/[色]
	/[オンラインヘルプ] (ページ 59)

下記を参照してください。

変数パラメータ (ページ 71)

4.1.4 例: 対話画面を開く

プログラミング

新しい[Example]の対話画面は、「スタートアップ」操作エリアから[Example]のスタートソフトキーで呼び出します。

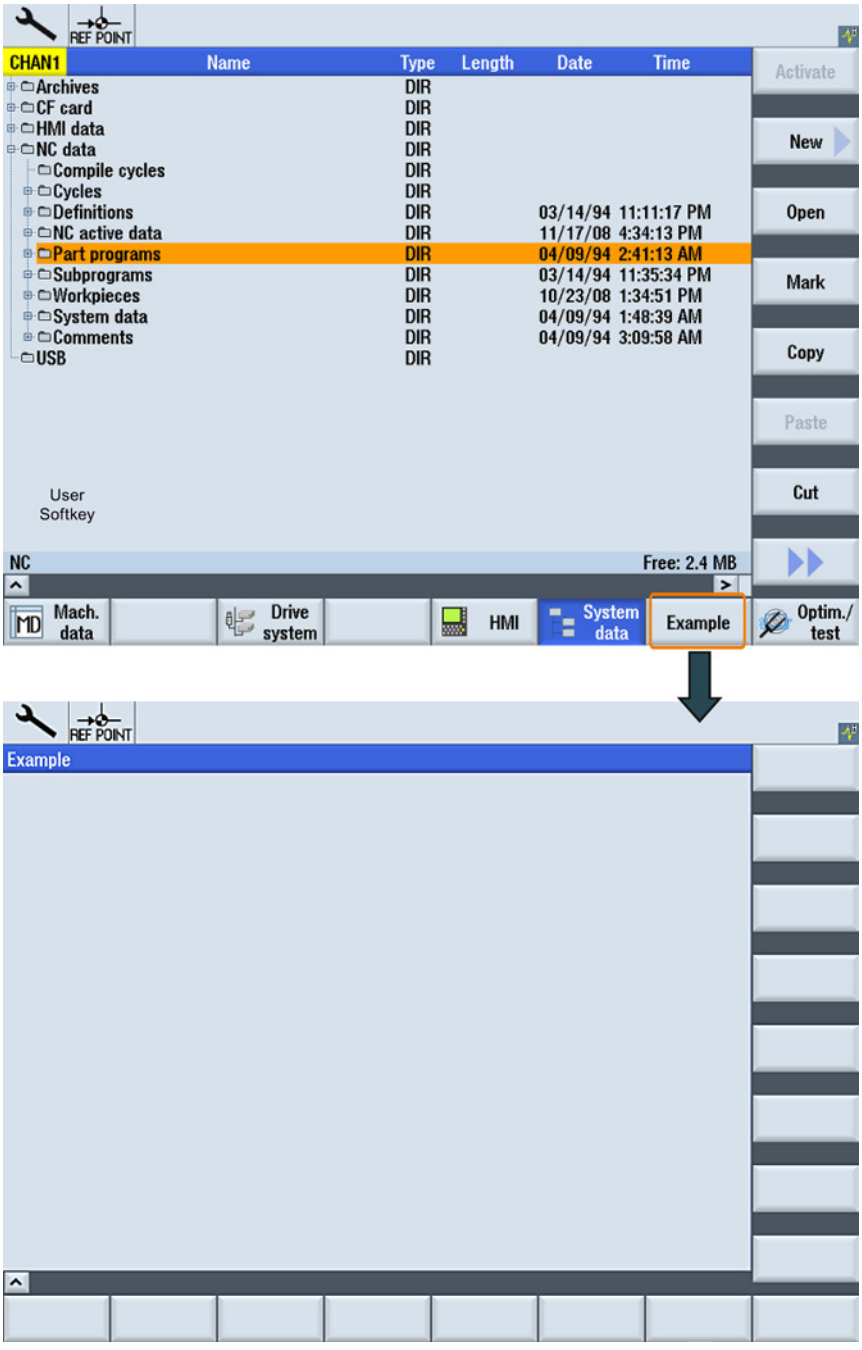


図 4-5　例: 新しい対話画面の呼び出し

```
//S(Start)
HS7=("Example", ac7, se1)

PRESS(HS7)
  LM("Maske1")
END_PRESS

//END
//M(Maske1/"Example")
HS1=("")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("")
VS8=("")
...                                          ;  メソッド
//END
```

4.1.5　複数列のある対話画面の定義

概要

複数の変数を対話画面の一行に表示することもできます。
この場合、変数はすべて設定ファイルで1つの定義行に定義します。

```
DEF VAR11 = (S///"Var11"), VAR12 = (I///"Var12")
```

設定ファイルの個々の変数をさらに読みやすくするため、定義行は各変数定義と、それに続くコンマの後で改行することができます。

キーワード「DEF」は常に新しい行の開始を示します。

```
DEF Tnr1=(I//1/"","T ","/wr1///,,10/20,,50),

  TOP1=(I///,"Type="/WR2//"$TC_DP1[1,1]"/80,,30/120,,50),

  TOP2=(R3///,"L1="/WR2//"$TC_DP3[1,1]"/170,,30/210,,70),

  TOP3=(R3///,"L2="/WR2//"$TC_DP4[1,1]"/280,,30/320,,70),

  TOP4=(R3///,"L3="/WR2//"$TC_DP5[1,1]"/390,,30/420,,70)

DEF Tnr2=(I//2/"","T ","/wr1///,,10/20,,50),

  TOP21=(I///,"Typ="/WR2//"$TC_DP1[2,1]"/80,,30/120,,50),

  TOP22=(R3///,"L1="/WR2//"$TC_DP3[2,1]"/170,,30/210,,70),

  TOP23=(R3///,"L2="/WR2//"$TC_DP4[2,1]"/280,,30/320,,70),

  TOP24=(R3///,"L3="/WR2//"$TC_DP5[2,1]"/390,,30/420,,70)

...
```

注記

複数列の対話画面を設定する場合、多数の列はシステムの速度低下の原因になる場合があることに注意してください!

4.1.6 表示イメージ/グラフィックの用途

グラフィックの用途

数種類のタイプがあります。

- グラフィックエリアの表示イメージ、またはグラフィックです。
- たとえば個々の変数を例示するヘルプ表示、それをグラフィックエリアに重ね合わせます。
- ショートテキストまたは入力/出力フィールドの代わりに、さらにヘルプ表示を設定することができます。この位置を自由に選択できます。

ファイルの保存先

対応する解像度のディレクトリで、以下の順で、接続されたモニタに合った解像度の画像が検索されます。

```
/user/sinumerik/hmi/ico/ico<Resolution>
/oem/sinumerik/hmi/ico/ico<Resolution>
/addon/sinumerik/hmi/ico/ico<Resolution>
```

画像が表示されない、または見つからなかった場合、以下の640 x 480ピクセルの解像度のディレクトリのいずれかにそれをコピーします。

```
/user/sinumerik/hmi/ico/ico640
/oem/sinumerik/hmi/ico/ico640
/addon/sinumerik/hmi/ico/ico640
```

注記
異なるパネルの解像度で、画像は均等に配置されます。

4.2 ソフトキーメニューの定義

定義

ソフトキーメニューという用語は、画面に表示されるすべての水平ソフトキーと垂直ソフトキーを指します。
既存のソフトキーメニュー以外に、他のメニューを定義することができ、これによって既存のメニューを部分的または全体的に上書きします。

ソフトキーの名称を事前に定義します。
すべてのソフトキーを割り当てる必要はありません。

HSx　x 1 - 8, 水平ソフトキー1～8

VSy　y 1 - 8, 垂直ソフトキー1～8

ソフトキーメニューの定義(ソフトキーメニュー定義ブロック)の基本構成は次の通りです。

定義ブロック	コメント	参照する章
//S...	; ソフトキーメニューの開始識別子	
HSx=...	; ソフトキーの定義	
PRESS(HSx) LM... END_PRESS	; メソッド開始識別子 ; 動作 ; メソッド終了識別子	「メソッド」の章を参照してください
//END	; ソフトキーメニューの終了識別子	

概要

ソフトキーメニューの定義のときにプロパティがソフトキーに割り当てられます。

プログラミング

構文:	*//S(識別子)* ... //END	; ソフトキーメニューの開始識別子 ; ソフトキーメニューの終了識別子
説明:	ソフトキーメニューの定義	
パラメータ:	識別子	ソフトキーメニューの名称
	テキストまたは画像ファイル名称	
構文:	**SK =** *(テキスト[, アクセスレベル][, 状態])*	
説明:	ソフトキーを定義します。	

パラメータ:	SK	ソフトキー(HS1 ～ HS8、VS1 ～ VS8)	
	テキスト	テキストの入力	
	表示ファイルの名称	"\\my_pic.png" または別のテキストファイル$85199を介した名称、たとえば(言語固有の)テキストファイルのなかの下記のテキスト等: 85100 0 0 "\\my_pic.png" ソフトキーに表示できるイメージのサイズは、使用しているOP(操作パネル)によって変わります。	
		OP 08: OP 010: OP 012: OP 015: OP 019:	640 x 480 mm → 25 x 25ピクセル 640 x 480 mm → 25 x 25ピクセル 800 X 600 mm → 30 x 30ピクセル 1024 X 768 mm → 40 x 40ピクセル 1280 x 1024 mm → 72 x 72ピクセル
	アクセスレベル	ac0 ～ ac7 (ac7: 初期設定)	
	状態	se1: 表示可(初期設定) se2: 表示不可(グレー表示テキスト) se3: ハイライト表示(最後に使用されたソフトキー)	

注記

改行するためには、ソフトキーテキストに%nを入力します。
最大2行(行あたり9文字)まで入力できます。

アクセスレベルの割り当て

オペレータはこれ以下のアクセスレベルの情報にのみアクセスできます。
各アクセスレベルの意味は次の通りです。
ac0は最上位のアクセスレベル、ac7は最下位のアクセスレベルです。

アクセスレベル	ロック方法	範囲
ac0	当社用に予約済	
ac1	パスワード	工作機械メーカ
ac2	パスワード	サービス
ac3	パスワード	ユーザー
ac4	キーロックスイッチ位置 3	プログラマ、機械セットアップ担当者
ac5	キーロックスイッチ位置 2	有資格者
ac6	キーロックスイッチ位置 1	教育受講者
ac7	キーロックスイッチ位置 0	非熟練オペレータ

例

```
//S(Menu1)                          ; ソフトキーメニューの開始識別子
HS1=("NEW", ac6, se2)               ;
                                    ソフトキーHS1の定義、ラベル「NEW」、保護レベル6、および状態「無効」を割り当て
HS2=("\\image1.png")                ; ソフトキーへのグラフィックの割り当て
HS3=("Exit")


VS1=("sub screen form")
VS2=($85011, ac7, se2)              ;
                                    ソフトキーVS2の定義、言語ファイルのテキスト、保護レベル1、および状態「無効」を割り当て
VS3=("Cancel", ac1, se3)            ;
                                    ソフトキーVS3の定義、ラベル「Cancel」、保護レベル1、および状態「ハイライト表示」を割り当て
VS4=("OK", ac6, se1)                ;
                                    ソフトキーVS4の定義、ラベル「OK」、保護レベル6、および状態「表示可」を割り当て
VS5=(SOFTKEY_CANCEL,,se1)           ;
```

	キャンセル標準ソフトキーVS5の定義と状態「表示可」の割り当て
VS6=(SOFTKEY_OK,,se1)	; OK標準ソフトキーVS6の定義と状態「表示可」の割り当て
VS7=(["\\image1.png","OEM text"],,se1)	; ソフトキーVS7の定義、イメージの割り当て、ラベル「OEM Text」および状態「表示可」を割り当て
VS8=(["\\bild1.png", $83533],,se1)	; ソフトキーVS8の定義、イメージの割り当て、言語ファイルのテキストおよび状態「表示可」を割り当て
PRESS(HS1)	; メソッド開始識別子
HS1.st="Calculate"	; ソフトキーへラベルテキストを割り当て
...	
END_PRESS	; メソッド終了識別子
PRESS(RECALL)	; メソッド開始識別子
LM("Screen form21")	; 対話画面の読み込み
END_PRESS	; メソッド終了識別子
//END	; ソフトキーメニューの終了識別子

4.2.1 実行中のソフトキープロパティの変更

概要

ソフトキープロパティのテキスト、アクセスレベル、状態を実行中にメソッドで変更することができます。

プログラミング

構文:	SK.st = "テキスト" SK.ac = アクセスレベル SK.se = 状態		; ラベルを含むソフトキー ; アクセスレベルを含むソフトキー ; 状態を含むソフトキー
説明:	プロパティの割り当て		
パラメータ:	テキスト		ダブルクォーテーションマークで囲んだラベルテキスト
	アクセスレベル		値の範囲: 0 ... 7
	状態	1: 2: 3:	表示可、オペレータ制御可能 表示不可(グレー表示テキスト) ハイライト表示(最後に使用されたソフトキー)

例

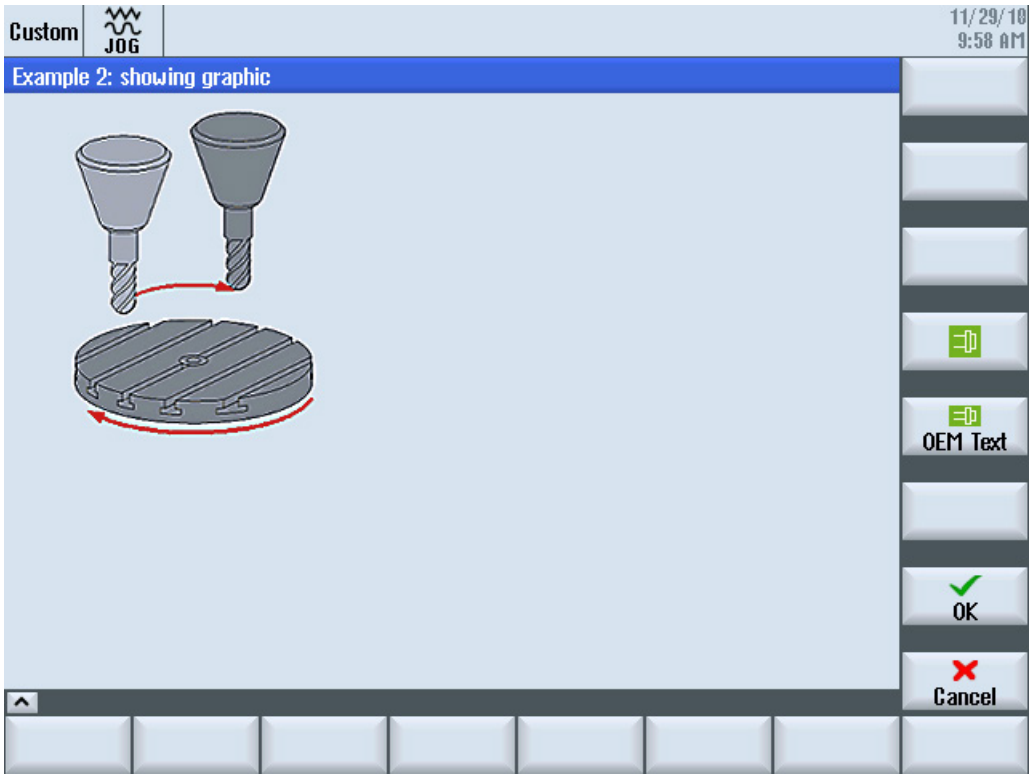


図 4-6　　例3: グラフィックとソフトキー

```
//S(Start)
HS7=("Example", ac7, sel)

PRESS(HS7)
  LM("Maske3")
END_PRESS

//END

//M(Maske3/"Example 2: showing graphic"/"example.png")
HS1=("")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
```

```
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
VS4=("\\sp_ok.png",,SE1)
VS5=(["\\sp_ok_small.png","OEM Text"],,SE1)
VS6=("")
VS7=(SOFTKEY_OK,,SE1)
VS8=(SOFTKEY_CANCEL,,SE1)
PRESS(VS4)
  EXIT
END_PRESS
PRESS(VS5)
  EXIT
END_PRESS
PRESS(VS7)
  EXIT
END_PRESS
PRESS(VS8)
  EXIT
END_PRESS
//END
```

4.2.2 言語テキスト

概要

言語テキストは次の目的に使用します。

- ソフトキーラベル
- 見出し
- ヘルプテキスト
- それ以外のテキスト

対話画面用の言語テキストはテキストファイルに設定されています。

このテキストファイルは下記のディレクトリに保存されています。

- /user/sinumerik/hmi/lng/
- /oem/sinumerik/hmi/lng/
- /addon/sinumerik/hmi/lng/

注記

テキストファイルは、プロジェクトファイルと同じ方法で保存してください。
例:
/user/sinumerik/hmi/lng/[テキストファイル]
/user/sinumerik/hmi/proj/[設定ファイル]

alsc.txt　当社の標準サイクル用の言語テキストを含みます。

almc.txt　メーカーサイクル用の言語テキストを含みます。

aluc.txt　ユーザー用の言語テキストです。

プログラム実行のときに使用されるテキストファイルはeasyscreen.iniファイルに指定されています。

```
[LANGUAGEFILES]

LngFile01 = alsc.txt ;->alsc<_xxx>.txt (例: alsc_eng.txt)

LngFile02 = user.txt ;->user<_xxx>.txt (例: user_eng.txt)
```

この場合は、テキストファイルの例としてuser.txtファイルが選択されています。
名称は、常に自由に選択できます。
ファイル内のテキストの言語に応じて、下記の構文を使用して、該当の言語コードを追加してください。 名称の後に、アンダーバーとそれに続く言語識別子を付けます。例: user_eng.txt。

下記を参照してください。

ファイル名称に使用される言語コードのリスト (ページ 204)

テキストファイルのフォーマット

テキストファイルはUTF-8フォーマットで保存してください。

たとえば、テキストファイルの作成にNotepadを使用する場合、[ファイル|名前を付けて保存]を選択して、UTF-8文字コードを選択します。

テキスト項目のフォーマット

構文:	8xxxx 0 0 "テキスト"		
説明:	ファイル内のテキスト番号にテキストを割り当てます。		
パラメータ:	xxxx	5000-9899	ユーザー用に予約済みのテキスト識別番号の範囲 固有の番号を割り当ててください。
	"テキスト"		対話画面に表示されるテキスト
	%n		改行するためのテキスト内の制御文字

パラメータ2と3はブランクで区切られ、アラームテキスト出力用の制御文字として作用します。
テキストフォーマットがアラームテキストのフォーマットと一致するように、この2つのパラメータは必ずゼロに設定してください。

アラームの例

```
85000 0 0 "イニシャル点"

85001 0 0 "穴あけ深さ"

85002 0 0 "ピッチ"

85003 0 0 "ポケット半径"
```

4.3 オンラインヘルプの設定

オンラインヘルプ

設定した対話画面、および要素のオンラインヘルプはHTMLフォーマットで作成できます。

オンラインヘルプの構文と処理はSINUMERIK Operateの場合と基本的に同じです。

入力フィールド用にオンラインヘルプを設定する場合、ユーザー専用オンラインヘルプの表示にオンラインヘルプの標準画面が使用されます。

```
DEF RFP=(R//1/,"RFP","RFP"///////"sinumerik_md_1.html","9006")
```

注記

LINUXであるため、HTMLファイルは小文字で書き込んでください!

HTMLファイルは、たとえばドイツ語の場合下記のディレクトリに保存されます。

```
/card/user/sinumerik/hmi/hlp/deu
/card/oem/sinumerik/hmi/hlp/deu
/card/addon/sinumerik/hmi/hlp/deu
```

言語識別子に対応する追加言語のディレクトリを作成してください(付録を参照してください)。

参照先

セットアップマニュアル『ベースソフトウェアとオペレーティングソフトウェア』(IM9)、「OEM用オンラインヘルプ」の章

変数 5

5.1 変数の定義

変数値

変数の最も基本的なプロパティはその値です。

変数値は下記の方法で割り当てることができます。

- 変数を定義するときの初期設定
- システム変数またはユーザー変数への割り当て
- メソッド

プログラミング

構文:	識別子.**val** = 変数値 識別子 = 変数値	
説明:	変数値 val (value(値))です。	
パラメータ:	識別子:	変数の名称
	変数値:	変数の値
例:	`VAR3 = VAR4 + SIN(VAR5)` `VAR3.VAL = VAR4 + SIN(VAR5)`	

変数状態

「変数状態」プロパティを使用して、実行中に変数の内容が有効であるかどうかをスキャンできます。 このプロパティは、FALSE値 = 0で読み出しと書き込みができます。

プログラミング

構文:	識別子.**vld**	
説明:	変数状態 vld (validation(有効性の確認))です。	
パラメータ:	識別子:	変数の名称
		スキャン結果は下記のようになります。
	FALSE =	無効な値
	TRUE =	有効な値

例:	`IF VAR1.VLD == FALSE`
	`  VAR1 = 84`
	`ENDIF`	

変数: プロパティの変更

変更の際、*識別子. プロパティ=*
*値*という表記で、変数に新しい値が割り当てられます。
等号の右側の式が評価され、これが変数または変数のプロパティに割り当てられます。

例:

識別子.**ac** = アクセスレベル	(ac: access level(アクセスレベル))
識別子.**al** = テキストの配置	(al: alignment(配置))
識別子.**bc** = 背景色	(bc: back color(背景色))
識別子.**fc** = 前面色	(fc: front color(前面色))
識別子.**fs** = フォントサイズ	(fs: font size(フォントサイズ))
識別子.**gt** = グラフィックテキスト	(gt: graphic text(グラフィックテキスト))
識別子.**hlp** = ヘルプ表示	(hlp: help(ヘルプ))
識別子.**htx** = ヘルプテキスト	(htx: help text(ヘルプテキスト))
識別子.**li** = 制限	(li: limit(制限))

識別子.**lt** = ロングテキスト (lt: long text(ロングテキスト))

識別子.**max** = 最大値 (max: maximum(最大))

識別子.**min** = 最小値 (min: minimum(最小))

識別子.**st** = ショートテキスト (st: short text(ショートテキスト))

識別子 .**typ** = 変数タイプ (typ: type(タイプ))

識別子.**ut** = 単位テキスト (ut: unit text(単位テキスト))

識別子.**val** = 変数値 (val: value(値))

識別子.**var** = システム変数またはユーザー変数 (var: variable(変数))

識別子.**vld** = 変数状態 (vld: validation(有効性の確認))

識別子.**wr** = 入力モード (wr: write(書く))

5.2 適用例

ヘルプ変数

ヘルプ変数は内部算術変数です。
算術変数は他の変数と同様に定義されますが、変数値と状態以外のプロパティがありません。そのため、ヘルプ変数は対話画面に表示されません。
ヘルプ変数はVARIANTタイプです。

プログラミング

構文:	DEF *識別子*	
説明:	VARIANTタイプの内部算術変数です。	
パラメータ:	識別子:	ヘルプ変数の名称
例:	`DEF OTTO ; ヘルプ変数の定義`	

構文:	識別子.val = *ヘルプ変数値* 識別子 = *ヘルプ変数値*	
説明:	メソッドで値がヘルプ変数に割り当てられます。	
パラメータ:	識別子:	ヘルプ変数の名称
	ヘルプ変数値:	ヘルプ変数の内容

例:

```
LOAD
  OTTO = "Test"              ; 値"Test" がヘルプ変数Ottoに割り当てられます。
END_LOAD
LOAD
  OTTO = REG[9].VAL          ; レジスタの値がヘルプ変数Ottoに割り当てられます。
END_LOAD
```

変数の計算

(ENTERキーまたは
TOGGLEキーを押して)I/Oフィールドを終了する毎に、変数が計算されます。
計算は、値が変更される毎に処理されるCHANGEメソッドに設定されています。

次の通り、変数状態をスキャンして、変数値の有効性を確認することができます。

```
IF VAR1.VLD == FALSE
   VAR1 = 84
ENDIF
```

システム変数の間接的アドレス指定

システム変数は別の変数の関数として間接的にアドレス指定することもできます。

```
PRESS(HS1)

  AXIS=AXIS+1

  WEG.VAR="$AA_DTBW["<<AXIS<<"]"    ;変数で軸のアドレスを指定。

END_PRESS
```

ソフトキーラベルの変更

例:

```
HS3.st = "New Text"   ; ソフトキーラベルの変更
```

5.3 例1: 変数タイプ、テキスト、ヘルプ画面、色、ヒント欄の割り当て

例1a

変数タイプ、テキスト、ヘルプ表示、色のプロパティの割り当て

DEF Var1 = (R///,"Actual value",,"mm"//"Var1.png"////8,2)	
変数タイプ:	REAL
制限または切り替えフィールドの項目:	なし
初期設定:	なし
テキスト:	
ロングテキスト:	なし
ショートテキスト:	Actual value
グラフィックテキスト:	なし
単位テキスト:	mm
属性:	なし
ヘルプ画面:	Var1.png
システム変数またはユーザー変数:	なし
ショートテキストの位置:	データなし(初期設定位置)
入力/出力フィールドの位置:	データなし(初期設定位置)
色:	
前面色:	8
背景色:	2

例1b

ヒント欄の割り当て

DEF Var2 = (I///5/"","value","","","," Tooltiptext"/wr2///20,250,50)	
変数タイプ:	INTEGER
制限または切り替えフィールドの項目:	なし
初期設定:	5
テキスト:	
ショートテキスト:	値(可能な言語テキストID)
ヒント欄:	ヒント欄テキスト
属性:	
入力モード	読み出しと書き込み
ヘルプ画面:	なし
ショートテキストの位置:	
左側からの距離	20
上からの距離	250
幅:	50
色:	データなし(初期設定)

下記も参照

変数パラメータ (ページ 71)

5.4 例2: 変数タイプ、制限、属性、ショートテキスト位置のプロパティの割り当て

例2

変数タイプ、制限、属性、ショートテキスト位置のプロパティの割り当て

DEF Var2 = (I/0,10///wr1,al1///,,300)	
変数タイプ:	INTEGER
制限または切り替えフィールドの項目:	最小値: 0 最大値: 10
初期設定:	なし
テキスト:	なし
属性:	
入力モード	読み出し専用
ショートテキストの配置	右揃え
ヘルプ画面:	なし
システム変数またはユーザー変数:	なし
ショートテキストの位置:	
左側からの距離	なし
上からの距離	なし(左上からの初期設定距離)
幅:	300
入力/出力フィールドの位置:	データなし(初期設定位置)
色:	データなし(初期設定)
ヘルプ:	なし

下記も参照

変数パラメータ (ページ 71)

5.5 例3: 変数タイプ、初期設定、システム変数またはユーザー変数、入力/出力フィールド位置のプロパティの割り当て

例3

変数タイプ、初期設定、システム変数またはユーザー変数、入力/出力フィールド位置のプロパティの割り当て

DEF Var3 = (R//10////"$R[1]"//300,10,200//)	
変数タイプ:	REAL
制限または切り替えフィールドの項目:	なし
初期設定:	10
テキスト:	なし
属性:	なし
ヘルプ画面:	なし
システム変数またはユーザー変数:	$R[1] (R変数1)
ショートテキストの位置:	入力/出力フィールドに対しての初期設定位置
入力/出力フィールドの位置:	
左側からの距離	300
上からの距離	10
幅:	200
色:	データなし(初期設定)

下記も参照

変数パラメータ (ページ 71)

5.6 切り替えフィールドとイメージの表示の例

例4

切り替えフィールドの各種項目

制限または切り替えフィールドの項目:

```
DEF Var1 = (I/* 0,1,2,3)
DEF Var2 = (S/* "In", "Out")
DEF Var3 = (B/* 1="In",                ;1と0の値で「In」と「Out」が表示されます。
0="Out")                               ; ARR1は配列の名称です。
DEF Var4 = (R/* ARR1)
```

例5

ショートテキストの代わりにイメージを表示:
イメージのサイズと位置は「I/Oフィールドの位置(左, 上, 幅, 高さ)」で定義されます。

DEF VAR6= (V///,"\\image1.png" ////160,40,50,50)	
変数タイプ:	VARIANT
制限または切り替えフィールドの項目:	なし
初期設定:	なし
テキスト:	
ショートテキスト:	image1.png
属性:	なし
ヘルプ画面:	なし
システム変数またはユーザー変数:	なし
ショートテキストの位置:	
左側からの距離:	160
上からの距離:	40
幅:	50
高さ:	50
入力/出力フィールドの位置:	該当なし
色:	データなし(初期設定)

5.7 変数パラメータ

パラメーター一覧

下記の一覧は変数パラメータの簡単な説明です。
詳細は以降の章に記載されています。

パラメータ	説明
変数タイプ (ページ 75)	変数タイプを指定します。
	R[x]: REAL(+ 小数位の桁数) I: INTEGER S[x]: STRING(+ 文字列長の桁数) C: CHARACTER(個別の文字) B: BOOL V: VARIANT
制限 (ページ 67)	最小値、最大値 初期設定: 空 制限値はコンマで区切られます。 制限は、10進数形式ではタイプI、C、Rで指定するか、「A」、「F」の形式の文字で指定します。
初期設定 (ページ 80)	初期設定がなく、変数に対してシステム変数またはユーザー変数が割り当てられていない場合、切り替えフィールドの1番目の要素が割り当てられます。 切り替えフィールドが定義されていない場合、初期設定はありません。つまり、変数状態は「計算されない」ことになります。 初期設定: 初期設定なし
切り替えフィールド (ページ 79)	入力/出力フィールドに予め定義された項目を含むリスト: リストは*で始まり、項目はコンマで区切られます。 項目には値を割り当てることができます。 切り替えフィールドでは、制限用の項目はリストとして解釈されます。 * が1つだけ入力された場合のみ、変数の切り替えフィールドが作成されます。 初期設定: なし

パラメータ	説明	
テキスト (ページ 65)	順序が指定されます。 ショートテキストの代わりに、イメージを表示することもできます。 初期設定: 空	
	ロングテキスト: ショートテキスト: グラフィックテキスト: 単位テキスト:	表示行のテキスト 対話画面要素の名称 テキストはグラフィック内の用語を表します 対話画面要素の単位
	ヒント欄 (ページ 65)	画面設定内の、表示および切り替えフィールド用の簡単な情報です。 この情報は、プレーンテキストおよび言語テキストIDによって設定されます。
属性 (ページ 67)	属性は下記のプロパティに作用します。 • 入力モード • アクセスレベル • ショートテキストの配置 • フォントサイズ • 制限 • 対話画面がCHANGEブロックで開いたときの動作 属性はコンマで区切られ、任意の順序で表示されます。 属性は切り替えフィールドには適用されません。 各成分に対して定義することができます。	
	入力モード	wr0: I/Oフィールドは表示不可、ショートテキストは表示可 wr1: 読み出し(入力用のフォーカスは不可) wr2: 読み出しと書き込み(行は白色表示) wr3: フォーカス付きのwr1 wr4: 変数要素はすべて表示不可、フォーカス不可 wr5: 入力値は、キー操作毎にすぐ保存されます(wr2との違いは、wr2では、フィールドを終了するか、RETURNが押されたときのみ保存されます)。 初期設定: wr2

パラメータ	説明	
	アクセスレベル	空: 常に書き込みできます。 ac0...ac7: アクセスレベル アクセスレベルが不適切な場合、最初の行がグレーで表示されます。(初期設定): ac7
	ショートテキストの配置	al0: 左揃え al1: 右揃え al2: 中央揃え 初期設定: al0
	フォントサイズ	fs1: 初期設定のフォントサイズ (8 pt.) fs2: ダブルフォントサイズ 初期設定: fs1 行間のすきまを定義します。 初期設定のフォントサイズでは、対話画面は16行になります。 グラフィックと単位のテキストは初期設定のフォントサイズでのみ設定できます。
	制限	この結果として、変数値が指定された最大値と最小値の範囲内かどうかをチェックできます。 初期設定: 指定された制限により特定 li0: チェックなし li1: 最小値に対してチェック li2: 最大値に対してチェック li3: 最小値と最大値に対してチェック
	対話画面を開くときの特性	変数定義で変数に対して指定されたcb属性は、対話画面定義でのcb初期設定より優先されます。 複数の属性はコンマで区切ります。
	cb0	この変数用に定義されたCHANGEブロックは、対話画面を開くときに処理されます(初期設定)。 複数の属性はコンマで区切ります。
	cb1	この変数用に定義されたCHANGEブロックは、変数値が変更された場合のみ処理されます。
ヘルプ表示 (ページ 65)	ヘルプ表示ファイル	pngファイルの名称 初期設定: 空

パラメータ	説明
	ヘルプ表示ファイルの名称はダブルクォーテーションマークで囲まれて表示されます。 カーソルが変数上にあれば、(以前のグラフィックに代わり)この表示が自動的におこなわれます。
システム変数またはユーザー変数 (ページ 68)	この変数には、NC/PLCのシステムまたはユーザーデータを割り当てることができます。 システム変数またはユーザー変数はダブルクォーテーションマークで囲まれて表示されます。 参照先: リストマニュアルシステム変数、/PGAsl/
ショートテキストの位置 (ページ 82)	ショートテキストの位置(左側からの距離、上からの距離、幅) この位置はピクセル単位で入力し、対話画面の本体の左上コーナーを基準にしています。 項目はコンマで区切られます。
入力/出力フィールドの位置 (ページ 82)	入力/出力フィールドの位置(左側からの距離、上からの距離、幅、高さ) この位置はピクセル単位で入力し、対話画面の本体の左上コーナーを基準にしています。 項目はコンマで区切られます。 この位置が変更されると、ショートテキスト、グラフィックテキスト、単位テキストの位置も変更されます。
色 (ページ 65)	前面色、背景色: 色はコンマで区切られます。 色の設定は入力/出力フィールドにのみ適用されます。その他のテキストに色を指定することはできません。 データ範囲: 1...10 初期設定: 前面色: 黒、背景色: 白 入力/出力フィールドの初期設定の色は、書き込みモードによって特定されます。 「wr」は書き込みモードを意味します。

5.8 変数タイプに関する詳細

変数タイプ INTEGER

「INTEGER」タイプでは、入力/出力フィールドの表示とメモリタイプを特定するために、下記の拡張ができます。

拡張データタイプの**2番目の文字**

表示フォーマット	
B	2進数
D	符号付き10進数
H	16進数
データなし	符号付き10進数

拡張データタイプの**3番目および/または4番目の文字**

メモリタイプ	
B	バイト
W	ワード
D	ダブルワード
BU	バイト、符号なし
WU	ワード、符号なし
DU	ダブルワード、符号なし

INTEGERデータタイプの文字の順序

1. 「I」 INTEGERの略称
2. 表示フォーマット
3. メモリタイプ
4. 「U」 符号なし

有効なINTEGERタイプの指定	
IB	2進数表記の32ビットの整数変数
IBD	2進数表記の32ビットの整数変数
IBW	2進数表記の16ビットの整数変数
IBB	2進数表記の8ビットの整数変数
I	10進数表記の32ビットの整数変数、符号付き
IDD	10進数表記の32ビットの整数変数、符号付き
IDW	10進数表記の16ビットの整数変数、符号付き
IDB	10進数表記の8ビットの整数変数、符号付き
IDDU	10進数表記の32ビットの整数変数、符号なし
IDWU	10進数表記の16ビットの整数変数、符号なし
IDBU	10進数表記の8ビットの整数変数、符号なし
IH	16進数表記の32ビットの整数変数
IHDU	16進数表記の32ビットの整数変数
IHWU	16進数表記の16ビットの整数変数
IHBU	16進数表記の8ビットの整数変数

VARIANT変数タイプ

VARIANT変数タイプは最後に割り当てた値のデータタイプによって特定されます。
これはISNUM機能またはISSTR機能でスキャンできます。
VARIANTタイプは主に変数名称または数値をNCコードに書き込む目的に適しています。

プログラミング

変数のデータタイプはチェックすることができます。

構文:	**ISNUM** *(VAR)*	
パラメータ:	VAR	データタイプがチェックされる変数の名称です。
		スキャン結果は下記のようになります。
	FALSE =	変数は数値ではありません(データタイプ = STRING)
	TRUE =	変数は数値です(データタイプ = REAL)

構文:	**ISSTR** *(VAR)*	
パラメータ:	VAR	データタイプがチェックされる変数の名称です。
		スキャン結果は下記のようになります。
	FALSE =	変数は数値です(データタイプ = REAL)
	TRUE =	変数は数値ではありません(データタイプ = STRING)
例:	IF ISNUM(VAR1) == TRUE IF ISSTR(REG[4]+2) == TRUE	

変数の表示モードは変更することができます。

- INTEGERの場合、表示タイプを変更することができます。

B	2進数
D	符号付き10進数
H	16進数
符号なし	
符号なしの場合はUを付けます。	

- REALデータタイプの場合、小数点以下の桁数のみ変更できます。

  タイプを変更した場合は許可されていないため、easyscreen_log.txtファイルにエラーメッセージが作成されます。

  例:

```
Var1.typ = "IBW"

Var2.typ = "R3"
```

数値フォーマット

数値は2進数、10進数、16進数、または指数表記で表すことができます。

2進数	B01110110
10進数	123.45
16進数	HF1A9
指数	-1.23EX-3
例:	
	`VAR1 = HF1A9`
	`REG[0]= B01110110`
	`DEF VAR7 = (R//-1.23EX-3)`

注記

「GC」機能でコードを作成する場合、10進数または指数表記の数値のみ処理されます。2進数または16進数の数値は処理されません。

下記も参照

変数パラメータ (ページ 71)

5.9 切り替えフィールドに関する詳細

概要

切り替えフィールド拡張機能を使用して、NC/PLC変数に応じてテキスト(切り替えフィールドの項目)を表示することができます。
切り替えフィールド拡張を利用する変数は読み出し専用です。

プログラミング

構文:	**DEF 識別子 =(変数タイプ /+ $テキスト番号 | * 値="\\イメージ"[値="\\image2.png"][, ...]** /[初期設定] /[テキスト(ロングテキスト, ショートテキスト, グラフィックテキスト, 単位テキスト)] /[属性] /[ヘルプ表示] /[システム変数またはユーザー変数] /[ショートテキストの位置] /[入力/出力フィールドの位置(左, 上, 幅, 高さ)] /[色]	
説明:	対話画面が開くとき、テキスト番号$85015の内容がI/Oフィールドに表示されます。 システム変数DB90.DBB5に初期値15が入力されています。 システム変数DB90.DBB5に保存されている値が変更されると、表示されるテキスト番号$(85000 + DB90.DBB5>)が、その都度再計算されます。	
パラメータ:	変数タイプ	システム変数またはユーザー変数に指定されている変数タイプ
	テキスト番号	基本番号として有効な言語テキストの(基本)番号
	システム変数またはユーザー変数	最後のテキスト番号(基本 + オフセット)の表示に使用されるシステム変数またはユーザー変数(オフセット)
例:	`DEF VAR1=(IB/+ $85000/15////"DB90.DBB5")`	

変数切り替えフィールド

変数の切り替えフィールドを対話画面要素に割り当てることができます。すなわち、切り替えキーを押すと、CHANGEメソッドで設定された値が変数に割り当てられます。

変数を定義する際、変数切り替えフィールドを識別するために、アスタリスク*を制限または切り替えフィールドのプロパティに入力します。

例: `DEF VAR1=(S/*)`

切り替えフィールド関連の表示イメージ

切り替えフィールドはグラフィックと重ね合わせます。グラフィックはフラグバイトの値に応じて変わります。
フラグバイトの値が1の場合、「image1.png」が表示されます。
値が2の場合、「image2.png」が表示されます。

```
DEF VAR1=(IDB/*1="\\image1.png",

              2="\\image2.png"//,$85000/wr1//"MB[0]"//160,40,50,50)
```

イメージのサイズと位置は「I/Oフィールドの位置(左, 上, 幅, 高さ)」で定義されます。

下記も参照

変数パラメータ (ページ 71)

5.10 初期設定に関する詳細

概要

変数は、変数フィールド(I/Oフィールドまたは切り替えフィールド)に初期値が割り当てられているか、またはシステム変数/ユーザー変数の両方またはいずれかが割り当てられているかに応じて、さまざまな状態になります(計算されない場合:
有効な値が変数に割り当てられた場合のみ切り替えが可能です)。

初期設定の範囲

条件			結果
フィールドタイプ	初期設定	システム変数またはユーザー変数	フィールドタイプの応答
I/Oフィールド	yes	yes	初期値をシステム変数またはユーザー変数に書き込みます
	No	yes	システム変数またはユーザー変数を初期値として使用します
	Fault	yes	計算なし。システム変数またはユーザー変数へ書き込みません、または使用しません
	yes	No	初期設定
	No	No	計算なし
	Fault	No	計算なし
	yes	Fault	計算なし
	No	Fault	計算なし
	Fault	Fault	計算なし
切り替え	yes	yes	初期値をシステム変数またはユーザー変数に書き込みます
	No	yes	システム変数またはユーザー変数を初期値として使用します
	Fault	yes	計算なし システム変数またはユーザー変数へ書き込みません、または使用しません
	Yes	No	初期設定
	No	No	初期設定 = 1番目の切り替えフィールド要素
	Fault	No	計算なし
	Yes	Fault	計算なし
	No	Fault	計算なし
	Fault	Fault	計算なし

下記も参照

変数パラメータ (ページ 71)

5.11 ショートテキストの位置、入力/出力フィールドの位置に関する詳細

概要

ショートテキストとグラフィックテキスト、入力/出力フィールドと単位テキストは、それぞれペアのように扱われます。すなわち、ショートテキストの位置設定はグラフィックテキストに適用され、入力/出力フィールドの設定は単位テキストに適用されます。

プログラミング

設定された位置項目は初期値に上書きします。つまり、変更できるのは1つの値だけです。
後続の画面要素に位置設定が設定されていない場合、前の画面要素の位置設定が適用されます。

対話画面要素に位置が設定されていない場合、初期設定が適用されます。
初期設定では、ショートテキストと入力/出力フィールドの列幅は、各行に対して列数と最大行幅に基づいて計算されます。列幅 = 最大行幅/列数となります。

グラフィックと単位のテキストの幅は予め定義され、プログラミングサポートの要求に合うように最適化されます。
グラフィックまたは単位テキストが設定されている場合、ショートテキストまたはI/Oフィールドの幅はそれに応じて縮小します。

ショートテキストとI/Oフィールドの順序は位置設定によって切り替えることができます。

下記も参照

変数パラメータ (ページ 71)

5.12 文字列の用途

文字列

文字列は設定の一部として使用できます。
これによって、テキストをダイナミックに表示したり、コード作成のために異なるテキスト同士を結合したりできます。

規則

文字列変数に関して下記の規則に従ってください。

- 論理演算は左から右に処理されます。
- ネスティングされた式は内側から外側に処理されます。
- アルファベットの大文字と小文字は区別されません。
- 文字列変数は、通常、左揃えで表示されます。

文字列はブランク文字列を割り当てるだけて削除できます。

演算子「**<<**」を使用して、等号の後に文字列を付加することができます。
文字列内のダブルクォーテーションマーク(")は、2つの連続するクォーテーションマークシンボルで表します。 IF命令で文字列が等しいかどうかをチェックできます。

例

下記の例の初期設定

```
VAR1.VAL = "This is an"

VAR8.VAL = 4

VAR14.VAL = 15

VAR2.VAL = "Error"

$85001 = "This is an"

$85002 = "Alarm text"
```

文字列の編集

- 文字列の結合

```
VAR12.VAL = VAR1 << " Error."   ;結果: This is an error
```

- 変数の削除

```
VAR10.VAL = ""   ;結果: ブランク文字列
```

- テキスト変数を使用した変数の設定

```
VAR11.VAL = VAR1.VAL   ;結果: This is an
```

- データタイプの一致:

```
VAR13.VAL ="This is the " << (VAR14 - VAR8) << ". error"

            ;結果: This is the 11th error
```

- 数値の処理

```
VAR13.VAL = "Error" << VAR14.VAL << ": " << $85001 << $85002

         ;結果: Error 15: This is an alarm text

IF VAR15 == "Error"   ; IF命令の文字列

  VAR16 = 18.1234

         ; 結果: VAR16 = 18.1234,

         ; VAR15 の内容が"Error"の場合

ENDIF
```

- 文字列の中のダブルクォーテーションマーク

```
VAR2=" Hello, this is a " Test""

   ; 結果: Hello, this is a " Test"
```

- 変数の内容に関連するシステム変数またはユーザー変数の文字列

```
VAR2.Var = "$R[" << VAR8 << "]"   ;結果: $R[4]
```

下記を参照してください。

STRING機能 (ページ 153)

5.13 CURPOS変数

説明

CURPOS変数を使用すると、現在の対話画面で有効な入力フィールドでカーソルの位置を表示したり、操作したりできます。
この変数はカーソルの前の文字数を示します。
カーソルが入力フィールドの先頭にある場合、CURPOSの値は0になります。CURPOSの値が変更されると、カーソルは入力フィールドの当該の位置にあります。

変数値の変更に対応できるようにするため、CHANGEブロックを使用して変更を監視することができます。
CURPOSの値が変更されると、CHANGEブロックにジャンプし、そこに含まれている命令が実行されます。

5.14 CURVER変数

説明

CURVER (CURrent
VERsion(現在のバージョン))のプロパティによって、異なるバージョンでも処理できるようプログラミングを適応させることができます。
CURVER変数は読み出し専用です。

注記

古いバージョンで以前に再コンパイルされた場合でも、コードは自動的に最新バージョンで作成されます。「GC」命令は常に最新バージョンを作成します。
作成されたバージョンを示す追加の識別子は、バージョン >
0で作成されたコードのユーザーコメントに挿入されます。

規則

常に、最新の対話画面がすべての変数と一緒に表示されます。

- 以前使用されていた変数は変更できないことがあります。
- 新しい変数は任意の順序で既存の(サイクル)プログラミングに挿入されます。
- あるバージョンからその次のバージョンに移行する際に、対話画面の変数は削除できません。
- 対話画面には全バージョンの全変数が含まれていなければなりません。

例

```
(IF CURVER==1 ...)                    ; コードを再コンパイルするとき、CURVERは、再コンパイルされるコードのバージョンを自動的に割り当てます。
```

5.15 ENTRY変数

概要

ENTRY変数を使用して、対話画面がどのメソッドによって呼び出されたかを確認することができます。

プログラミング

構文:	**ENTRY**	
説明:	ENTRY変数は読み出し専用の変数です。	
戻り値:		スキャン結果は下記のようになります。
	0 =	プログラミングサポートなし
	1 =	プログラミングサポートあり(対話画面はプログラミングサポートによって呼び出されました)
	2 =	プログラミングサポート + 以前の対話画面(サブ対話画面)からの初期設定
	3 =	プログラミングサポート + 再コンパイル
	4 =	プログラミングサポート + 作成されたコメント付きの再コンパイル(#記号あり)
	5 =	プログラミングサポート + 作成されたコメント付きの再コンパイル(#記号なし)

例

```
IF ENTRY == 0
 DLGL("The dialog was not called during programming")
ELSE
  DLGL("The dialog was called during programming")
ENDIF
```

5.16 ERR変数

概要

ERR変数を使用して、これより前の行が正しく実行されたかどうかを確認することができます。

プログラミング

構文:	**ERR**	
説明:	ERR変数は読み取り専用です。	
戻り値:		スキャン結果は下記のようになります。
	FALSE =	これより前の行はエラーなしで実行されました。
	TRUE =	これより前の行はエラー有りで実行されました。

例

```
VAR4 = Thread[VAR1,"CDM",3]              ; 配列からの出力値。
IF ERR == TRUE                           ; 値が配列内で見つかったかどうかを確認します。
  VAR5 = "Error accessing array"
                                         ; 値が配列内に見つからなかった場合、値「Error
                                         accessing array」が変数に割り当てられます。
ELSE
  VAR5 = "All OK"                        ; 値が配列内に見つかった場合、「All
                                         OK」の値が変数に割り当てられます。
ENDIF
```

5.17 FILE_ERR変数

概要

FILE_ERR変数を使用して、これより前のGCまたはCP命令が正しく実行されたかどうかを確認することができます。

プログラミング

構文:	**FILE_ERR**	
説明:	FILE_ERR変数は読み取り専用です。	
戻り値:		とりうる結果は下記のようになります。
	0 =	動作OK
	1 =	ドライブ/パスは使用不可
	2 =	パス/ファイルへのアクセスエラー
	3 =	ドライブ準備未完了
	4 =	不正なファイル名称
	5 =	ファイルはすでに開いています
	6 =	アクセス拒否
	7 =	目的のパスは使用不可または使用禁止
	8 =	コピーソースがターゲットと同じです
	10 =	内部エラー: FILE_ERR = 10は、エラーを他のカテゴリに分類できなかったことを意味します。

例

```
CP("D:\source.mpf","E:\target.mpf")
                                        ;
                                        source.mpfからE:\target.mpfにコピーします。
IF FILE_ERR > 0                         ; 異常が発生したか確認します。
   IF FILE_ERR == 1                     ;
```

```
CP("D:\source.mpf","E:\target.mpf")
                                                    特定のエラー番号を確認して、対応するエラーテキストを出力します。
      VAR5 = "Drive/path not available"
   ELSE
      IF FILE_ERR == 2
         VAR5 = "Path/file access error"
      ELSE
         IF FILE_ERR == 3
            VAR5 = "incorrect file name"
         ENDIF
      ENDIF
   ENDIF
ELSE
   VAR5 = "All OK"                                  ;
                                                    CP(またはGC)に異常が発生しなかった場合、「All OK」が出力されます。
ENDIF
```

5.18 FOC変数

概要

変数FOCを使用して、対話画面で、入力フォーカス(現在有効な入/出力フィールド)が制御されます。
左カーソル、右カーソル、上カーソル、下カーソルの効果は、PGUPやPGDNと同様に、事前定義されています。

注記

FOC機能は、ナビゲーションイベントの結果として、開始してはいけません。
ソフトキーPRESSブロック、CHANGEブロック、...では、カーソル位置だけを変更できます。
FOC機能は、入力モードwr = 0、wr = 4の変数、またはヘルプ変数には適用できません。

プログラミング

構文:	FOC	
説明:	この変数は、読み出しも書き込みもできます。	
戻り値:	読み出し	結果は、FOC機能に適用された変数の名称です。
	書き込み	文字列または数値を割り当てることができます。 文字列は変数名称として解釈され、数値は変数インデックスとして解釈されます。

例

```
IF FOC == "Var1"                ; フォーカスを読み取ります。
  REG[1] = Var1
ELSE
  REG[1] = Var2
ENDIF

FOC = "Var1"                    ; 入力フォーカスは変数1に割り当てられます。
FOC = 3                         ;
                                入力フォーカスはWR ≥ 2の3番目の対話画面要素に割り当てられます。
```

5.19 S_CHAN変数

説明

S_CHAN変数を使用して、表示または評価のために現在のチャネル番号を特定することができます。

プログラミング命令 6

6.1 演算子

概要

プログラミング時に下記の演算子を使用することができます。

- 算術演算子
- 比較演算子
- 論理(ブール)演算子
- ビット演算子
- 三角関数

6.1.1 算術演算子

概要

算術演算子	識別子
+	加算
-	減算
*	乗算
/	除算
MOD	モジュロ演算
( )	丸括弧
AND	AND演算子
OR	OR演算子
NOT	NOT演算子
ROUND	小数位のある数値の四捨五入

例: `VAR1.VAL = 45 * (4 + 3)`

ROUND

ROUND演算子は、対話画面設定の実行のときに、最大12桁の小数位がある数値の四捨五入に使用します。 変数フィールドの表示は小数位に対応できません。

用途

ROUNDは2つのパラメータで制御します。

```
VAR1 = 5,2328543

VAR2 = ROUND( VAR1, 4 )
```

結果: `VAR2 = 5,2339`

VAR1は四捨五入する桁数を含みます。
パラメータ「4」は結果の小数桁数を示し、これがVAR2に割り当てられます。

三角関数

三角関数	識別子
SIN(x)	xのサイン
COS(x)	xのCOS
TAN(x)	xのTAN
ATAN(x, y)	x/yのATAN
SQRT(x)	xの平方根
ABS(x)	xの絶対値
SDEG(x)	degへ変換
SRAD(x)	radへの変換

注記
これらの関数はrad単位で演算がおこなわれます。
このために、SDEG()とSRAD()の関数を使用して変換することができます。

例: `VAR1.VAL = SQRT(2)`

定数

定数	
PI	3.14159265358979323846
FALSE	0
TRUE	1

例: `VAR1.VAL = PI`

比較演算子

比較演算子	
==	等しい
<>	等しくない
>	より大きい
<	より小さい
>=	以上
<=	以下

例

```
IF VAR1.VAL == 1

  VAR2.VAL = TRUE

ENDIF
```

条件

ネスティング深さには制約がありません。

1つの命令を使用した条件:

```
IF
...
ENDIF
```

2つの命令を使用した条件:

```
IF
...
ELSE
...
ENDIF
```

6.1.2 ビット演算子

概要

ビット演算子	識別子
BOR	ビット単位論理和
BXOR	ビット単位排他的論理和
BAND	ビット単位論理積
BNOT	ビット単位論理否定
SHL	ビットを左にシフト
SHR	ビットを右にシフト

SHL演算子

ビットを左にシフトするにはSHL (SHIFT LEFT)演算子を使用します。
シフトされる値とシフト回数の両方を、直接または変数で指定できます。
データフォーマットの制限に到達しても、ビットは制限を超えてビットをシフトします。このときエラーメッセージは表示されません。

用途

構文:	変数 = *値* **SHL** *シフト量*	
説明:	左にシフトします。	
パラメータ:	値	シフトされる値
	シフト量	シフト回数

例

```
PRESS(VS1)
  VAR01 = 16 SHL 2            ; 結果 = 64
  VAR02 = VAR02 SHL VAR04     ;
                              VAR02の内容を符号なし32ビットに変換し、これを左にVAR04で指定したビット数だけシフトします。
                              続いて、32ビット値を変数VAR02のフォーマットに変換します。
END_PRESS
```

SHR演算子

ビットを右にシフトするにはSHR (SHIFT RIGHT)演算子を使用します。
シフトされる値とシフト回数の両方を、直接または変数で指定できます。
データフォーマットの制限に到達しても、ビットは制限を超えてビットをシフトします。このときエラーメッセージは表示されません。

用途

構文:	変数 = *値* **SHR** *シフト量*		
説明:	右にシフトします。		
パラメータ:	値	シフトされる値	
	シフト量	シフト回数	

例

```
PRESS(VS1)
  VAR01 = 16 SHR 2                ; 結果 = 4
  VAR02 = VAR02 SHR VAR04         ;
                                  VAR02の内容を符号なし32ビットに変換し、これを右にVAR04で
                                  指定したビット数だけシフトします。続いて、32ビット値を変数V
                                  AR02のフォーマットに変換します。
END_PRESS
```

6.2 メソッド

概要

さまざまなタイプのイベント(入力フィールド終了、ソフトキー操作)によって、対話画面および対話画面関連のソフトキーメニュー(新たに設定された対話画面から呼び出されるソフトキーメニュー)で特定の動作が開始されます。
これらの動作はメソッドで設定されます。

下記の表にはメソッドをプログラム指令するために適用される基本原則を示しています。

定義ブロック	コメント	参照する章
`PRESS(HS1)`	; メソッド開始識別子	
`LM...` `  LS...`	; 機能	「機能」の章を参照してください。
`Var1.st = ...`	; プロパティの変更	「ソフトキーメニュー」の章と 「対話画面の構成と要素」の章を参照してください。

`Var2 = Var3 + Var4`
`...`
`EXIT`
	; 変数の計算	「変数の定義」の章を参照してください。

`END_PRESS`	; メソッド終了識別子	

6.2.1 CHANGE

概要

変数値が変更されると、CHANGEメソッドが実行されます。
すなわちCHANGEメソッドの中に、変数値の変更直後に実行される変数計算が設定されています。

要素専用とグローバルの2種類のCHANGEメソッドがあります。

- **要素専用のCHANGEメソッド**は、指定された変数値が変更された場合に実行されます。
  変数にシステム変数またはユーザー変数が割り当てられている場合、変数値の周期的更新をCHANGEメソッドに設定することができます。
- **グローバルなCHANGEメソッド**は、任意の変数値が変更され、要素専用のCHANGEメソッドが設定されていない場合に実行されます。

「要素専用」のプログラミング

構文:	CHANGE*(識別子)* ... END_CHANGE	
説明:	特定の変数値を変更します。	
パラメータ:	識別子	変数の名称

例

```
DEF VAR1=(I//////"DB20.DBB1")          ; Var1にシステム変数が割り当てられます。
CHANGE(VAR1)
  IF VAR1.Val <> 1
  VAR1.st="Tool OK!"                   ; システム変数値 ≠
                                       1の場合、変数状態のショートテキストは次の通りになります
                                       。 Tool OK!
    otto=1
  ELSE
    VAR1.st="Attention: Error!"        ; システム変数値 =
                                       1の場合、変数状態のショートテキストは次の通りになります
```

```
                                        。 Attention: Error!
    otto=2
  ENDIF
  VAR2.Var=2
END_CHANGE
```

「グローバル」のプログラミング

構文:	CHANGE() ... END_CHANGE
説明:	任意の変数値を変更します。
パラメータ:	- なし -

例

```
CHANGE()
  EXIT                                  ; 任意の変数値が変更された場合、対話画面は終了します。
END_CHANGE
```

6.2.2 FOCUS

概要

フォーカス(カーソル)が対話画面の別のフィールドにある場合、FOCUSメソッドが実行されます。

ナビゲーション操作で、FOC機能を実行することはできません。
ソフトキーPRESSブロック、CHANGEブロックでは、カーソル位置だけを変更できます。 カーソル動作の応答は予約されています。

注記

FOCUSブロックの中で、別の変数を選択したり、新しい対話画面を読み込むことはできません。

プログラミング

構文:	FOCUS ... END_FOCUS
説明:	カーソルを位置決めします。
パラメータ:	- なし -

例

```
FOCUS
  DLGL("The focus has been placed on variable" << FOC << ".)
END_FOCUS
```

6.2.3 LOAD

概要

LOADメソッドは、変数とソフトキーの定義(DEF Var1= ..., HS1= ...)が解釈された後に実行されます。 このとき、対話画面はまだ表示されません。

プログラミング

構文:	LOAD ... END_LOAD
説明:	ダウンロードします。
パラメータ:	- なし -

例

```
LOAD                          ; 開始識別子
  Screen form1.Hd =  $85111   ; 言語ファイルから対話画面ヘッダー用のテキストを割り当てます。
  VAR1.Min = 0                ; 変数の最小値を割り当てます。
  VAR1.Max = 1000             ; 変数の最大値を割り当てます。
END_LOAD                      ; 終了コード
```

下記も参照

線と長方形 (ページ 161)

6.2.4 LOAD GRID

概要

LGメソッドを使用して、表記述をLOADブロックの中でダイナミックに使用することができます。

LGメソッドを使用して表を割り当てるには、変数が表変数として定義されていて、既存の有効な表との間でクロスリファレンスが確立されていなければなりません。

プログラミング

構文:	LG *(表名称, 変数名称 [,ファイル名称])*	
説明:	表を読み込みます。	
パラメータ:	表名称	ダブルクウォーテーションマークで囲まれた表の名称
	変数名称	ダブルクウォーテーションマークで囲まれた表に割り当てられている変数の名称
	ファイル名称	ダブルクウォーテーションマークで囲まれた表が定義されているファイルの名称。 表が変数の定義を含むファイルの中にも定義されていない場合のみ、指定が必要です。

例

```
LOAD
 LG("grid1","VAR1","GRID2.COM")
END_LOAD
```

GRID2.COMの内容

```
//G(grid1/0/5/1,1)
 (I///,"GRID1"/wr1//"1"/80/1)
 (R3///"longtext1","R1-R4"/wr2//"$R[1]"/80/1)
 (R3///"longtext2","R5-R8"/wr2//"$R[5]"/80/1)
 (R3///"longtext3","grid1"/wr2//"$R[9]"/80/2)
//END
```

6.2.5 UNLOAD

概要

UNLOADメソッドは対話画面をアンロードする前に実行されます。

プログラミング

構文:	UNLOAD ... END_UNLOAD
説明:	アンロードします。
パラメータ:	- なし -

例

```
UNLOAD
  REG[1] = VAR1             ; レジスタに変数を保存します
END_UNLOAD
```

6.2.6 OUTPUT

概要

OUTPUTメソッドは、「GC」機能が呼び出されると実行されます。
変数とヘルプ変数は、OUTPUTメソッドでNCコードとして設定されます。
コード行の個別の要素はブランクで結合します。

注記
NCコードはファイル機能で別のファイルに作成し、NCに転送することができます。

プログラミング

構文:	OUTPUT *(識別子)* ... END_OUTPUT	
説明:	NCプログラムの変数を出力します。	
パラメータ:	識別子	OUTPUTメソッドの名称

ブロック番号とスキップ識別子

再コンパイルの際に、パートプログラムの行番号を保持して、動作中のプログラムサポートで直接設定されたマーキングを非表示にする場合は、OUTPUTブロックには行番号またはスキップ識別子を含めてはいけません。

パートプログラムでエディタを使って変更されると、下記の動作になります。

条件	動作
ブロック数は変化しません。	ブロック番号は維持されます。
ブロック数は減少します。	最大ブロック番号が削除されます。
ブロック数が増加します。	新しいブロックにブロック番号が付きません。

例

```
OUTPUT(CODE1)
  "CYCLE82(" Var1.val "," Var2.val "," Var3.val ","Var4.val "," Var5.val "," Var6.val ")"
END_OUTPUT
```

6.2.7 PRESS

概要

PRESSメソッドは対応するソフトキーを押すと実行されます。

プログラミング

構文:	PRESS*(ソフトキー)* ... END_PRESS		
識別子:	押したソフトキー		
パラメータ:	ソフトキー	ソフトキーの名称: HS1 - HS8、VS1 - VS8	
	RECALL	<RECALL>キー	
	PU	ページアップ	画面を上に移動
	PD	ページダウン	画面を下に移動
	SL	左スクロール	カーソルを左に移動
	SR	右スクロール	カーソルを右に移動
	SU	上スクロール	カーソルを上に移動
	SD	下スクロール	カーソルを下に移動

例

```
HS1 = ("another softkey menu")
HS2=("no function")
PRESS(HS1)
  LS("Menu1")                          ; 別のソフトメニューを読み込みます。
  Var2 = Var3 + Var1
END_PRESS
PRESS (HS2)
END_PRESS
PRESS(PU)
  INDEX = INDEX -7
  CALL("UP1")
END_PRESS
```

6.2.8 OUTPUTブロックを使用したバージョン管理例

概要

操作画面を拡張する際、既存の対話画面に変数を追加することができます。
変数名称に続いて、括弧に囲まれたバージョン識別子が追加する変数に付加されます。
: (0 = オリジナル、書き込みなし)、1 = バージョン1、2 = バージョン2等

例

```
DEF var100=(R//1)                ;  オリジナルです、バージョン0に相当します
DEF var101(1)=(S//"Hello")       ;  バージョン1への拡張です。
```

OUTPUTブロックを書き込む際、どの変数を書き込むかを特定のバージョン識別子を参照して指定することができます。

例

```
OUTPUT(NC1)                      ;  オリジナルバージョンの変数のみOUTPUTブロックで使用することができます。
OUTPUT(NC1,1)                    ;  オリジナルバージョンの変数とバージョン識別子1を使用した拡張が、OUTPUTブロックで使用できます。
```

オリジナルバージョンのOUTPUTブロックの場合はバージョン識別子は不要ですが、0を指定することもできます。OUTPUT(NC1)はOUTPUT(NC1,0)と同じ意味になります。 OUTPUTブロックのバージョン識別子nは、オリジナル0, 1, 2, ...
nのすべての変数を含みます。

バージョン識別子を使用したプログラミング

```
//M(XXX)                         ;  バージョン0(初期設定)
DEF var100=(R//1)
DEF var101=(S//"Hello")
DEF TMP
VS8=("GC")
PRESS(VS8)
   GC("NC1")
END_PRESS
```

```
OUTPUT(NC1)
var100",,"var101
END_OUTPUT

; ************ バージョン1、拡張された定義 ***************
//M(XXX)
DEF var100=(R//1)
DEF var101=(S//"Hello")
DEF var102(1)=(V//"HUGO")
DEF TMP
VS8=("GC")
PRESS(VS8)
   GC("NC1")
END_PRESS
...

OUTPUT(NC1)                          ;  オリジナルと追加の新しいバージョン
var100","var101
END_OUTPUT
...

OUTPUT(NC1,1)                        ;  バージョン1
var100","var101"," var102
END_OUTPUT
```

6.3 機能

概要

対話画面と対話画面関連のソフトキーメニューで、さまざまな機能を使用することができます。
これらは、特定のイベント(入力フィールド終了、ソフトキーの操作)により有効にできます。そしてこれらはメソッドで設定します。

サブプログラム

特定の操作のプロセスを定義して、繰り返し使用する命令等は、サブプログラムで設定することができます。
サブプログラムはいつでもメインプログラムまたは他のサブプログラムに読み込むことができ、必要な回数だけ実行できます。このため命令は繰り返し設定する必要がありません。
対話画面またはソフトキーメニューの定義ブロックはメインプログラムの構成要素です。

外部機能

外部機能を使用して、追加のユーザー専用機能を統合することができます。
外部機能はDLLファイルに設定され、設定ファイルの定義行の項目で識別されます。

PIサービス

PI_SERVICE機能を使用して、NC領域で、PLCからPIサービス(PI Services:Program Invocation Services(プログラム起動サービス))をスタートできます。

下記を参照してください。

機能(FCT) (ページ 126)
PIサービス (ページ 158)

6.3.1 ブロックの定義(//B)

概要

プログラムファイルで、サブプログラムはブロック識別子//Bで識別され、//ENDで終了します。 各ブロック識別子で複数のサブプログラムを定義できます。

注記

サブプログラムで使用する変数は、サブプログラムが呼び出される対話画面で定義してください。

プログラミング

ブロックの構成は次の通りです。

構文:	**//B**(*ブロック名称*) **SUB***(識別子)* END_SUB [SUB(*識別子*) ... END_SUB] ... //END	
説明:	サブプログラムを定義します。	
パラメータ:	ブロック名称	ブロック識別子の名称
	識別子	サブプログラムの名称

例

```
//B(PROG1)                          ; ブロックの開始
SUB(UP1)                            ; サブプログラムの開始
  ...
  REG[0] = 5                        ; 値5をレジスタ0に割り当てます。
  ...
END_SUB                             ; サブプログラム終了
SUB(UP2)                            ; サブプログラムの開始
  IF VAR1.val=="Otto"
    VAR1.val="Hans"
    RETURN
  ENDIF
  VAR1.val="Otto"
END_SUB                             ; サブプログラム終了
//END                               ; ブロック終了
```

6.3.2 サブプログラム呼び出し(CALL)

概要

CALL機能を使用して、メソッド内の任意の位置から、読み込まれたサブプログラムを呼び出すことができます。
サブプログラムのネスティングがサポートされています。つまり、サブプログラムから別のサブプログラムを呼び出すことができます。

プログラミング

構文:	CALL*("識別子")*	
説明:	サブプログラムを呼び出します。	
パラメータ:	識別子	サブプログラムの名称

例

```
//M(SCREEN FORM1)
VAR1 = ...
VAR2 = ...
LOAD
  ...
  LB("PROG1")                  ; ブロックを読み込みます。
  ...
END_LOAD
CHANGE()
  ...
  CALL("UP1")                  ; サブプログラムを呼び出して実行します。
  ...
END_CHANGE
...
//END
```

6.3.3 変数のチェック(CVAR)

概要

CVAR (Check Variable(変数のチェック))機能を使用して、スキャンを実行して、画面内のすべてまたは特定の変数、あるいはヘルプ変数が正常であるかどうかを確認することができます。

GC機能を使用したNCコード作成の前に、変数が有効な値を含むかどうかをチェックすることが有用な場合があります。

変数状態 識別子.vld = 1の場合、変数は正常です。

プログラミング

構文:	**CVAR**(*VarN*)	
説明:	変数が有効かどうかをチェックします。	
パラメータ:	VarN	チェックする変数のリスト それぞれコンマで区切られた最大29個の変数をチェックできます。 文字長さは500を超えてはいけません。 スキャン結果は下記のようになります。
	1 =	TRUE(すべての変数は有効です)
	0 =	FALSE(無効な変数が少なくとも1つあります)

例

```
IF CVAR == TRUE                    ; すべての変数をチェックします。
  VS8.SE = 1                       ; すべての変数が正常である場合、ソフトキーVS8が表示されます。
ELSE
  VS8.SE = 2                       ;
                                   変数に無効な値が含まれる場合、ソフトキーVS8は無効になります。
ENDIF


IF CVAR(”VAR1”, ”VAR2”) == TRUE
                                   ; 変数VAR1、VAR2をチェックします。
  DLGL ("VAR1 and VAR2 are OK")
                                   ; VAR1、VAR2の値が正常な場合、対話画面行に[VAR1 and VAR2
                                   are OK]と表示されます。
ELSE
 DLGL ("VAR1 and VAR2 are not
OK")
                                   ; VAR1、VAR2の値が無効な場合、対話画面行に[VAR1 and VAR2
                                   are not OK]と表示されます。
ENDIF
```

6.3.4 プログラムファイルのコピー機能(CP)

概要

CP(プログラムのコピー)機能は、HMIファイルシステム、またはNCファイルシステムの中でファイルをコピーします。

プログラミング

構文:	**CP***("ソースファイル", "ターゲットファイル")*	
説明:	ファイルをコピーします。	
パラメータ:	ソースファイル	ソースファイルの絶対パス
	ターゲットファイル	ターゲットファイルの絶対パスデータ

戻り値を使用して、機能が正常に動作したかどうかを確認することができます。

```
CP("\MPF.DIR\CFI.MPF","\WKS.DIR\123.WPD\CFI.MPF",VAR1)
```

例

戻り値を使用したアプリケーション

```
CP("//NC/MPF.DIR/HOHO.MPF","//NC/MPF.DIR/ASLAN.MPF",VAR3)
CP("//NC/MPF.DIR/hoho.MPF",VAR0,VAR3)
CP(VAR4,VAR0,VAR3)
CP("CF_CARD:/mpf.dir/myprog.mdf","//NC/MPF.DIR/HOHO.MPF",VAR3)
CP("//NC/MPF.DIR/HOHO.MPF",                    ; xyzが存在しなければなりません
"CF_CARD:/xyz/123.pmf",VAR3)
```

戻り値を使用しないアプリケーション

```
CP("//NC/MPF.DIR/HOHO.MPF","//NC/MPF.DIR/ASLAN.MPF")
CP("//NC/MPF.DIR/hoho.MPF",VAR0)
CP(VAR4,VAR0)
CP("CF_CARD:/mpf.dir/myprog.mdf","//NC/MPF.DIR/HOHO.MPF")
CP("//NC/MPF.DIR/HOHO.MPF",                          ; xyzが存在しなければなりません
"CF_CARD:/xyz/123.mpf")
```

下記を参照してください。

FILE_ERRのサポート: FILE_ERR変数 (ページ 89)

6.3.5 プログラムファイルの削除機能(DP)

概要

DP(Delete Program(プログラムの削除))機能は、パッシブなHMIファイルシステムまたはアクティブなNCファイルシステムからファイルを削除します。

プログラミング

構文:	**DP***("ファイル")*	
説明:	ファイルを削除します。	
パラメータ:	ファイル	削除するファイルの絶対パス名称

例

この機能には、下記のデータ管理構文を使用します。

- 戻り値あり

```
DP("//NC/MPF.DIR/XYZ.DIR ", VAR1)
```

VAR1 = 0　　ファイルは削除されました。

VAR1 = 1　　ファイルは削除されませんでした。

- 戻り値なし

```
DP("//NC/MPF.DIR/XYZ.DIR ")
```

```
DP("\MPF.DIR\CFI.MPF")
```

6.3.6 プログラムファイルの存在確認機能(EP)

概要

EP(Exist Program(プログラムの存在確認))機能は、NCファイルシステムまたはHMIファイルシステムの指定されたパスに特定のNCプログラムがあるかどうかをチェックします。

プログラミング

構文:	EP(*"ファイル"*)	
説明:	NCプログラムがあるかどうかをチェックします。	
パラメータ:	ファイル	NCファイルシステムまたはHMIファイルシステムのファイルの絶対パス
戻り値:	スキャン結果を割り当てる変数の名称	
		スキャン結果は下記のようになります。 • M = ファイルはHMIファイルシステムにあります。 • N = ファイルはNCファイルシステムにあります。 • 空の文字列 = ファイルはHMIにもNCにもありません。

EP機能は新しい構文、および(適用済みの構文の) 古い論理を扱うことができます。

適切な名称を使用して、ファイルを直接アドレス指定します。

```
//NC/MPF.DIR/XYZ.MPF
```

または

`CF_CARD: /MPF.DIR/XYZ.MPF` (/user/sinumerik/data/progを指します)

または

`LOC:` (CF_CARDに対応)

新しい構文

```
EP("//NC/MPF.DIR/XYZ.MPF ", VAR1)
EP("CF_CARD:/MPF.DIR/XYZ.MPF ", VAR1)
EP("LOC:/MPF.DIR/XYZ.MPF ", VAR1)
```

```
; 戻り値:
; VAR1 = 0                    ファイルは存在します。
; VAR1 = 1                    ファイルは存在しません。
```

古い構文

```
EP("/MPF.DIR/CFI.MPF", VAR1)
; 戻り値:
; VAR1 = M                    ファイルはHMIファイルシステムにあります。
; VAR1 = N                    ファイルはNCファイルシステムにあります。
; VAR1 = B                    ファイルはHMIファイルシステムとNCファイルシステムにあります。
```

例

```
EP("\MPF.DIR\CFI.MPF", VAR1)                ;
                                            CFI.MPFファイルがHMIファイルシステムにあるかどうかをチェックします。
IF VAR1 == "M"
  DLGL("File is located in the HMI file
system")
ELSE
  IF VAR1 == "N"
    DLGL("File is located in the NC file
system")
  ELSE
    DLGL("File is located neither in the
HMI nor in the NC file system")
  ENDIF
ENDIF
```

6.3.7 プログラムファイルの移動機能(MP)

概要

MP(Move Program(プログラムの移動))機能はHMIファイルシステムまたはNCファイルシステムの中でファイルをコピーします。

プログラミング

構文:	MP*("ソース", "ターゲット")*	
	MP("CF_CARD:/MPF.DIR/MYPROG.MPF","//NC/MPF.DIR")	
説明:	ファイルを移動します。	
パラメータ:	ソースファイル	絶対パスデータ
	ターゲットファイル	絶対パスデータ

例

```
MP("//NC/MPF.DIR/123.MPF","//NC/MPF.  // 絶対パス
DIR/ASLAN.MPF",VAR3)
MP("//NC/MPF.DIR/123.MPF","//NC/MPF.  // ファイル名称のないターゲット
DIR",VAR3)
MP("//NC/MPF.DIR/123.MPF",VAR0,VAR3)  // 変数によるターゲット
MP(VAR4,VAR0,VAR3)                    // 変数によるソースとターゲット
MP("CF_CARD:/mpf.dir/myprog.mdf","//  // CFカードからNCへ
NC/MPF.DIR/123.MPF",VAR3)
MP("//NC/MPF.DIR/HOHO.MPF","CF_CARD:  // NCからCFカードへ
/xyz/123.mpf",VAR3)
MP("USB:/mpf.dir/myprog.mdf","//NC/M  // USBからNCへ
PF.DIR",VAR3)//
```

6.3.8 プログラムファイルの選択機能(SP)

概要

SP(Select Program(プログラムの選択))機能は、実行のために、有効なNCファイルシステムのなかのファイルを選択します。すなわち、ファイルは事前にNCに読み込まれていなければなりません。

プログラミング

構文:	SP*("ファイル")*	
識別子:	プログラムの選択	
パラメータ:	"ファイル"	NCファイルの絶対パス名称

例

この機能には、下記のデータ管理構文を使用します。

- 戻り値あり

```
SP("//NC/MPF.DIR/MYPROG.MPF", VAR1)
```

VAR1 = 0　ファイルは読み込まれました。

VAR1 = 1　ファイルは読み込まれませんでした。

- 戻り値なし

```
SP("//NC/MPF.DIR/MYPROG.MPF")
```

```
//M(TestGC/"Code generation:")
DEF VAR1 = (R//1)
DEF VAR2 = (R//2)
DEF D_NAME
LOAD
  VAR1 = 123
  VAR2 = -6
END_LOAD
OUTPUT(CODE1)
```

```
//M(TestGC/"Code generation:")
  "Cycle123(" VAR1 "," VAR2 ")"
  "M30"
END_OUTPUT
PRESS(VS1)
  D_NAME =
"CF_CARD:/MPF.DIR/MESSEN.MPF"
  GC("CODE1",D_NAME)                ;
                                    OUTPUTメソッドからの
                                    コードをCF_CARD:/MP
                                    F.DIR/MESSEN.MPFフ
                                    ァイルに書き込みます
                                    。
END_PRESS
PRESS(HS8)
  MP("CF_CARD:/MPF.DIR/MESSEN.MPF", ;
"//NC/MPF.DIR")                     ファイルをNCに読み込
                                    みます。
  SP("\MPF.DIR\MESSEN.MPF")         ;
                                    ファイルを選択します
                                    。
END_PRESS
```

6.3.9 対話画面行(DLGL)

概要

特定の状況に対する動作として、対話画面行に出力するショートテキスト(メッセージまたは入力のヒント)を設定することができます。

初期設定のフォントサイズでの許容文字数: 約 50

プログラミング

構文:	**DLGL**(*"文字列"*)	
説明:	対話画面行にテキストを出力します。	
パラメータ:	文字列	対話画面行に表示されるテキスト

例

```
IF Var1 > Var2
  DLGL("Value too large!")    ; 変数1 > 変数2の場合、対話画面行に[Value too
                               large!]のテキストが表示されます。
ENDIF
```

6.3.10 評価(EVAL)

概要

EVAL機能は転送された式を評価し、これを実行します。
この機能で、実行中に式をプログラム指令することができます。
これは、たとえば変数へのインデックス付きアクセス操作に有用です。

プログラミング

構文:	**EVAL***(式)*	
説明:	式を評価します。	
パラメータ:	式	論理式

例

```
VAR1=(S)
VAR2=(S)
VAR3=(S)
VAR4=(S)
CHANGE()
  REG[7] = EVAL("VAR"<<REG[5])  ;
                                REG[5]の値が3の場合、括弧内の式はVAR3になります。したがって
                                、REG[7]にはVAR3が割り当てられます。
  IF REG[5] == 1
    REG[7] = VAR1
  ELSE
    IF REG[5] == 2
      REG[7] = VAR2
    ELSE
      IF REG[5] == 3
        REG[7] = VAR3
      ELSE
        IF REG[5] == 4
          REG[7] = VAR4
        ENDIF
      ENDIF
    ENDIF
  ENDIF
END_CHANGE
```

6.3.11 対話画面の終了(EXIT)

概要

EXIT機能を使用して、対話画面を終了し、マスター対話画面に戻ります。
マスター対話画面が見つからない場合、新たに設定した操作画面を終了して、標準アプリケーションに戻ります。

プログラミング(パラメータなし)

構文:	**EXIT**
説明:	対話画面を終了します。
パラメータ:	- なし -

例

```
PRESS(HS1)
  EXIT
END_PRESS
```

概要

現在の対話画面が転送変数付きで呼び出された場合、変数値を変更し、これを出力対話画面に送ることができます。

変数値はそれぞれ、「LM」機能を使用して、この出力対話画面から後続の対話画面に送られた変数に割り当てられます。
それぞれコンマで区切られた最大20個の変数値を送ることができます。

注記

変数または変数値の順序は、割り当て異常を回避するためLM機能に設定された転送値の順序と同じにしてください。 指定されていない変数値は転送時に変更されません。
変更された転送変数はLM機能実行時に出力対話画面で直ちに有効になります。

転送変数を使用したプログラミング

構文:	**EXIT***[(VARx)]*	
説明:	対話画面を終了し、1つ以上の変数を転送します。	
パラメータ:	VARx	ラベル変数

例

```
//M(Screen form1)
...
PRESS(HS1)
  LM("SCREEN
FORM2","CFI.COM",1, POSX,
POSY, DIAMETER)
                              ;
                              画面1を中断し、画面2を開きます。
                              このとき、変数POSX、POSY、DIAMETERを転送します。
  DLGL("Screen form2          ;
ended")                       画面2から戻った後、次のテキストが画面1の対話画面の行に表示されます。Screen form2
                              ended
END_PRESS
...
//END


//M(Screen form2)
...
PRESS(HS1)
EXIT(5, ,
CALCULATED_DIAMETER)
                              ;
                              画面2を終了し、LMの後の行で画面1に戻ります。
                              このとき、値5を変数POSXに割り当て、変数CALCULATED_DIAMETERの値をDIAMETER変数に割り当てます。
                              変数POSYは現在値を保持します。
END_PRESS
...
//END
```

6.3.12 終了、ソフトキーの読み込み(EXITLS)

概要

EXITLS機能を使用して、現在の操作画面を終了して、定義されたソフトキーメニューを読み込むことができます。

プログラミング

構文:	**EXITLS**(*"ソフトキーメニュー"[, "パス"]*)	
説明:	対話画面を終了し、ソフトキーメニューを読み込みます。	
パラメータ:	ソフトキーメニュー	読み込むソフトキーメニューの名称
	パス名称	読み込むソフトキーメニューのディレクトリパス

例

```
PRESS(HS1)
    EXITLS( "Menu1", "AEDITOR.COM" )
END_PRESS
```

6.3.13 機能(FCT)

概要

外部機能はDLLファイルに設定され、設定ファイルの定義行の項目で識別されます。

注記
外部機能には1つ以上の戻りパラメータが設定されていなければなりません。

プログラミング

構文:	**FCT***機能名称 =* *("ファイル"/戻りのタイプ/固定パラメータのタイプ/変数パラメータのタイプ)*	
	FCT InitConnection = ("c:\tmp\xyz.dll"/I/R,I,S/I,S)	
説明:	外部機能は、LOADブロックまたはPRESSブロック等で呼びすことができます。	
パラメータ:	機能名称	外部機能の名称
	ファイル	DLLファイルの絶対パス
	戻りのタイプ	戻り値のデータタイプ
	固定パラメータのタイプ	値パラメータ
	変数パラメータのタイプ	参照パラメータ
	データタイプはコンマで区切られます。	

外部機能は、LOADブロックまたはPRESSブロック等で呼びすことができます。

例:

```
press(vs4)

RET = InitConnection(VAR1,13,"Servus",VAR2,VAR17)

end_press
```

外部機能の構成

外部機能では特定の専用記号を考慮してください。

構文:	external "C" dllexport void InitConnection (ExtFctStructPtr FctRet, ExtFctStructPtr FctPar, char cNrFctPar)	
説明:	DLLのエクスポートです。(Windowsに実装している場合のみ) 指定される転送パラメータを厳密に定義します。 実際の呼び出しパラメータは転送の構成を使用して送られます。	
パラメータ:	cNrFctPar	呼び出しパラメータの個数 = FctPar内の構成要素の個数
	FctPar	構成要素フィールドに対するポインタです。これにはデータタイプ付きの特定の呼び出しパラメータを含みます。
	FctRet	データタイプ付きの機能戻り値の構成に対するポインタ

転送構成の定義

```
union CFI_VARIANT
 (
 char                                    b;
 short int                               i;
 double                                  r;
 char*                                   s;
 )
typedef struct ExtFctStructTag
 (
 char                                    cTyp;
 union CFI_VARIANT                       value;
 )ExtFctStruct;
typedef struct ExtFct* ExtFctStructPtr;
```

プラットフォーム(Windows、Linux)に依存しないで外部機能を開発する場合、キーワードのdeclspec(dllexport)は使用できません。

このキーワードはWindowsの場合のみ必要です。
たとえば、Qtでは下記のマクロを使用できます。

```
#ifdef Q_WS_WIN
       #define MY_EXPORT __declspec(dllexport)
#else
       #define MY_EXPORT
#endif
```

この機能は次の通りに宣言します。

```
extern "C" MY_EXPORT void InitConnection
(ExtFctStructPtr FctRet, ExtFctStructPtr FctPar,char cNrFctPar)
```

「Run MyScreens」で設定した画面をNCUとPCU/PCで使用する場合、バイナリファイルの拡張子は省略してください。

```
FCT InitConnection = ("xyz"/I/R,I,S/I,S)
```

絶対パス情報が省かれた場合、「Run MyScreens」はまず設定されたディレクトリでバイナリファイルを検索します。

6.3.14 コードの作成(GC)

概要

GC(Generate code(
コードの作成))機能はOUTPUTメソッドからNCコードを作成します。

プログラミング

構文:	**GC**("*識別子*"[,"*ターゲットファイル*"][,*オプション*],[*付加*])	
説明:	NCコードの生成	
パラメータ:	識別子	コードを作成するOUTPUTブロックの名称
	ターゲットファイル	HMIファイルシステムまたはNCファイルシステムのターゲットファイルのパス名称 ターゲットファイルが指定されていない場合(これはプログラミングサポートシステムの中でのみ可能です)、コードは現在開いているファイルの中のカーソル位置に書き込まれます。
	オプション	コメント作成オプション
	0:	(初期設定)再コンパイル用にコメント付きコードを作成します。
	1:	作成されたコードにコメントを作成しません。 **注:** このコードは再コンパイルできません(関連項目: コメントなしの再コンパイル (ページ 148))。
	付加	このパラメータはターゲットファイルが指定された場合のみ関連します。
	0:	(初期設定)すでにファイルが存在する場合、古い内容が削除されます。
	1:	すでにファイルが存在する場合、新しいコードはファイルの先頭に書き込まれます。
	2:	すでにファイルが存在する場合、新しいコードはファイルの末尾に書き込まれます。

例

```
//M(TestGC/"Code generation:")
DEF VAR1 = (R//1)
DEF VAR2 = (R//2)
DEF D_NAME
LOAD
  VAR1 = 123
  VAR2 = -6
END_LOAD
OUTPUT(CODE1)
  "Cycle123(" VAR1 "," VAR2 ")"
  "M30"
END_OUTPUT

PRESS(VS1)
  D_NAME = "\MPF.DIR\MESSEN.MPF"
  GC("CODE1",D_NAME)                    ;
                                        OUTPUTメソッドからコードをファイル\MPF.DIR\MESSEN.
                                        MPFに書き込みます。
                                        Cycle123(123, -6)
                                        M30

END_PRESS
```

再コンパイル

- **ターゲットファイルに項目なし**

  GC機能はプログラミングサポートシステムでのみ使用できます。そしてエディタで現在開いているファイルにNCコードを書き込みます。
  NCコードを再コンパイルすることができます。
  「Run MyScreens」で指定されるターゲットファイルなしでGC機能を設定すると、実行時にエラーメッセージが出力されます。

- **ターゲットファイルに項目あり**

  OUTPUTブロックから作成されたコードはターゲットファイルに転送されます。
  ターゲットファイルが存在しない場合、NCファイルシステムでセットアップされます。
  ターゲットファイルがHMIファイルシステムに保存される場合、このファイルはハードディスクに保存されます。

ユーザーコメント行(コードの再コンパイルに必要な情報)はセットアップされません。つまり、このコードは再コンパイルできません。

ターゲットファイル指定時の特別な注意項目

原則として、ターゲットファイルの指定方法は2種類あります。

- **NC表記** `/_N_MPF_DIR/_N_MY_FILE_MPF`

  ファイルはNCのMPFディレクトリに作成されます。

- **DOS表記** `d:\abc\my_file.txt`または`\\RemoteRechner\files\my_file.txt`

  ハードディスクまたはリモートPCでディレクトリを使用できる場合、ファイルはハードディスクまたは指定されたPCの、指定されたディレクトリに保管されます。

注記

無効な変数は、作成されたNCコードのなかでブランク文字列を作成し、読み込まれたときにログブックにエラーメッセージを作成します。

再コンパイルの特記事項

マスター対話画面からの変数はサブ対話画面で使用できるため、サブ対話画面でGC機能を呼び出すことはできません。
ただし、これらの変数は直接呼び出しに応えて使用することはできません。

作成されたコードをエディタを使って手動で処理する場合、コード作成プログラムで作成された値の文字数は変更しないでください。
この値を変更するとコードを再コンパイルできなくなるおそれがあります。

対策

1. 再コンパイル
2. 設定された対話画面を使用して変更。 (たとえば、99 → 101)
3. GC

下記も参照

再コンパイル (ページ 146)

6.3.15 配列の読み込み(LA)

概要

LA(Load Array(配列の読み込み))機能を使用して、他のファイルから配列を読み込むことができます。

プログラミング

構文:	**LA***(識別子 [, ファイル])*	
説明:	ファイルから配列を読み込みます。	
パラメータ:	識別子	読み込む配列の名称
	ファイル	配列が定義されたファイル

注記

現在の設定ファイル内の配列を他の設定ファイル内の配列で置き換える場合、両方の配列の名称は同じにしてください。

例

```
                                ; maske.comファイルから取り出します。
DEF VAR2 =
(S/*ARR5/"Out"/,"Toggle
field")
PRESS(HS5)
  LA("ARR5","arrayext.com")     ; arrayext.comファイルから配列ARR5を読み込みます。
  VAR2 = ARR5[0]                ;
                                VAR2切り替えフィールドには[Out]/[In]の代わりに"Above"/"Bel
                                ow"/"Right"/"Left"が
                                表示されます。
END_PRESS
//A(ARR5)
("Out"/"In")
//END
```

```
                                    ; arrayext.comファイルから取り出します。
//A(ARR5)
("Above"/"Below"/"Right"/"Lef
t")
//END
```

注記

LA機能を使用して他の配列を変数の切り替えフィールドに割り当てた後に、必ず変数に有効な値を割り当ててください。

6.3.16 ブロックの読み込み(LB)

概要

LB(Load Block(ブロックの読み込み))機能を使用して、実行中にサブプログラムを含むブロックを読み込むことができます。
読み込まれたサブプログラムをいつでも呼び出せるように、LBをLOADメソッドで設定します。

注記

サブプログラムを読み込まなくても済むように、対話画面で直接定義することもできます。

プログラミング

構文:	**LB***("ブロック名称"[,"ファイル"])*	
説明:	実行中にサブプログラムを読み込みます。	
パラメータ:	ブロック名称	ブロック識別子の名称
	ファイル	設定ファイルのパス名称 初期設定 = 現在の設定ファイル

例

```
LOAD
  LB("PROG1")              ; 現在の設定ファイルでブロック「PROG1」を検索し、読み込みます。
  LB("PROG2","XY.COM")     ; 設定ファイルXY.COMでブロック「PROG2」を検索し、読み込みます。
END_LOAD
```

6.3.17 対話画面の読み込み(LM)

概要

LM機能を使用して、新しい対話画面を読み込むことができます。

マスター対話画面/サブ対話画面

他の対話画面を呼び出すが、それ自体では終了しない対話画面は、マスター対話画面と呼ばれます。
マスター対話画面によって呼び出される対話画面はサブ対話画面と呼ばれます。

プログラミング

構文:	**LM***("識別子"[,"ファイル"] [,MSx [, VARx]])*	
説明:	対話画面の読み込み	
パラメータ:	識別子	読み込む対話画面の名称
	ファイル	設定ファイルのパス名称(HMIファイルシステムまたはNCファイルシステム); 初期設定:現在の設定ファイル
	MSx	対話画面変更モード

	0:	(初期設定) 現在の対話画面は拒否され、新しい対話画面が読み込まれて、表示されます。 EXITによって標準アプリケーションに戻ります。 MSxパラメータを使用して、対話画面変更時に現在の対話画面を終了するかどうかを特定することができます。 現在の対話画面を閉じない場合、変数を新しい対話画面に転送することができます。 MSxパラメータの長所は、対話画面変更時に、対話画面の再初期設定が必ずしも必要でないことです。その代わりに、現在の対話画面のデータとレイアウトが保持され、データ転送が容易におこなわれます。
	1:	LM機能を開始すると、現在のマスター対話画面が中断されます。 新しいサブ対話画面が読み込まれ、表示されます。 EXITはサブ対話画面を終了し、マスター対話画面が中断された位置まで戻ります。 マスター対話画面では、中断されている間にUNLOADブロックは処理されません。
	VARx	必要条件: MS1 マスター対話画面からサブ対話画面に転送できる変数のリストです。 それぞれコンマで区切られた最大20個の変数を転送できます。

注記

パラメータVARxは、それぞれの場合に、変数の値だけを転送しますが(すなわち、サブ対話画面での変数の読み出し/書き込みが可能ですが)、サブ対話画面に表示することはできません。
変数は、EXIT機能でサブ対話画面からマスター対話画面に戻すことができます。

例

```
PRESS(HS1)

  LM("SCREEN FORM2","CFI.COM",1, POSX, POSY, DIAMETER)
                                        ; 画面1を中断し、画面2を開きます。
                                        このとき、変数POSX、POSY、およびDIAMETERが転送されます。
  DLGL("Screen form2 ended")            ;
                                        画面2から戻った後、次のテキストが画面1の対話画面の行に表示されます。Screen form2 ended

END_PRESS
```

6.3.18 ソフトキーの読み込み(LS)

概要

LS機能を使用して、他のソフトキーメニューを表示することができます。

プログラミング

構文:	**LS***("識別子"[, "ファイル"][, マージ])*	
説明:	ソフトキーメニューを表示します。	
パラメータ:	識別子	ソフトキーメニューの名称
	ファイル	設定ファイルのパス(HMIファイルシステムまたはNCファイルシステム) 初期設定:現在の設定ファイル
	マージ	
	0:	既存のソフトキーがすべて削除され、新しく設定されたソフトキーが入力されます。
	1:	初期設定 新たに設定されたソフトキーのみ、既存のソフトキーに上書きします。 他のソフトキー(=HMIアプリケーションのソフトキー)はその機能とテキストが保持されます。

例

```
PRESS(HS4)
  LS("Menu2",,0)       ;
                       メニュー2で既存のソフトキーメニューを上書きし、表示されているソフトキーは
                       削除されます。
END_PRESS
```

注記

インタープリタが対話画面を表示していない限り(すなわち、まだLM機能が処理されていない限り)は、1つのLSまたは1つのLM命令のみ(他の動作は除く)を、スタートソフトキーまたはソフトキーメニュー用の定義ブロックのPRESSメソッドに設定することができます。

LSとLM機能は、ソフトキーPRESSブロックの中で呼び出すことはできますが、ナビゲーションキー(PU、PD、SL、SR、SU、SD)を押している場合は動作しません。

6.3.19 NC/PLCの読み出し(RNP)、NC/PLCの書き込み(WNP)

概要

RNP(Read NC PLC Variable(NC PLC変数の読み出し))命令を使用して、NCまたはPLC変数あるいはマシンデータを読み出すことができます。

プログラミング

構文:	**RNP** ("*システム変数またはユーザー変数*", *値*)	
説明:	NC変数またはPLC変数、あるいはマシンデータを読み出します。	
パラメータ:	システム変数またはユーザー変数	NC変数またはPLC変数の名称
	値	システム変数またはユーザー変数に書き込む値 値が文字列タイプの場合、ダブルクウォーテーションマークで囲んで記述します。

例

```
VAR2=RNP("$AA_IN[2]")          ; NC変数の読み出し
```

概要

WNP(Write NC PLC Variable(NC
PLC変数の書き込み))命令を使用して、NCまたはPLC変数あるいはマシンデータを書き込むことができます。

WNP機能を実行する毎に、新たにNC/PLC変数にアクセスします。つまり、NC/PLCアクセスは常にCHANGEメソッドでおこなわれます。
システム変数またはユーザー変数の値が頻繁に変更される場合、この方法を使用することを推奨いたします。
NC/PLC変数に一回だけアクセスする場合、LOADまたはUNLOADメソッドで設定してください。

プログラミング

構文:	**WNP***("システム変数またはユーザー変数", 値)*	
説明:	NC変数またはPLC変数、あるいはマシンデータを書き込みます。	
パラメータ:	システム変数またはユーザー変数	NC変数またはPLC変数の名称
	値	システム変数またはユーザー変数に書き込む値 値が文字列タイプの場合、ダブルクウォーテーションマークで囲んで記述します。

例

```
WNP("DB20.DBB1",1)          ; PLC変数の書き込み
```

6.3.20 複数のNC PLC読み出し(MRNP)

概要

MRNP命令を使用して、一回のレジスタアクセスで複数のシステム変数またはOPI変数を転送することができます。
このアクセス方法は、その個別にアクセスして読み出すよりはるかに速くおこなえます。 システム変数またはOPI変数は同じ領域のMRNP命令の中に含めてください。

システム変数またはOPI変数の領域は次の通りに構成されています。

- 一般的なNCデータ($MN..., $SN.., /nck/...)
- チャネル別NCデータ($MC..., $SC.., /channel/...)
- PLCデータ(DB..., MB.., /plc/...)
- 同じ軸の軸別NCデータ($MA..., $SA..)

プログラミング

構文:	MRNP*(変数名称1*変数名称2[* ...], レジスタインデックス)*
説明:	複数の変数を読み出します。
パラメータ:	変数名称で「*」は区切りを表します。 値は命令に現れる変数名称の順序でレジスタREG[レジスタインデックス]以降に転送されます。 したがって、下記が適用されます。 第1の変数値はREG[レジスタインデックス]にあります。 第2の変数値はREG[レジスタインデックス + 1]にあります(これ以降も同様)。

注記

レジスタ数には制限があり、変数のリストは500文字を超えてはならないことに注意してください。

例

```
MRNP("$R[0]*$R[1]*$R[2]*$R[3]",1)        ; 変数$R[0] ～ $R[3]の値がREG[1] ～
                                          REG[4]に書き込まれます。
```

表示マシンデータの読み出し

表示マシンデータはLOADブロックの中でRNP ($MM...)を使用して読み出すことができます。

「Run MyScreens」を使用して、表示マシンデータへの一般的な読み出し/書き込みアクセスをおこなうことはできません。

注記

ユーザー変数はシステム変数またはPLC変数と同じ名称を持つことはできません。

NC変数

すべてのマシンデータ、セッティングデータ、R変数、および特定のシステム変数のみが使用できます。(こちらを参照してください: アクセス可能なシステム変数のリスト (ページ 205)).

すべてのグローバルチャネル別ユーザー変数(GUD)にアクセスできます。
ただし、ローカル変数とグローバルプログラムユーザー変数は処理できません。

マシンデータ	
グローバルマシンデータ	$MN_...
軸マシンデータ	$MA_...
チャネルマシンデータ	$MC_...

セッティングデータ	
グローバルセッティングデータ	$SN_...
軸セッティングデータ	$SA_...
チャネルセッティングデータ	$SC_...

システム変数	
R変数1	$R[1]

PLC変数

すべてのPLCデータを使用することができます。

PLCデータ	
データブロックxのバイトyビットz	DBx.DBXy.z
データブロックxのバイトy	DBx.DBBy
データブロックxのワードy	DBx.DBWy
データブロックxのダブルワードy	DBx.DBDy
データブロックxの実数y	DBx.DBRy
フラグバイトxビットy	Mx.y
フラグバイトx	MBx
フラグワードx	MWx
フラグダブルワードx	MDx
入力バイトxビットy	Ix.yまたはEx.y
入力バイトx	IBxまたはEBx
入力ワードx	IWxまたはEWx
入力ダブルワードx	IDxまたはEDx
出力バイトxビットy	Qx.yまたはAx.y
出力バイトx	QBxまたはABx
出力ワードx	QWxまたはAWx
出力ダブルワードx	QDxまたはADx
データブロックxからの長さがzの文字列y	DBx.DBSy.z

6.3.21 レジスタ(REG)

レジスタの説明

レジスタは異なる対話画面間のデータ交信に必要です。
レジスタは各対話画面に割り当てられます。
レジスタは最初の対話画面が読み込まれたときに作成されて、値0またはブランク文字列が割り当てられます。

注記
レジスタは直接にNCコード作成用のOUTPUTブロックで使用しないでください。

プログラミング

構文:	**REG*[x]***	
説明:	レジスタを定義します。	
パラメータ:	x	x = 0...19のレジスタインデックス; タイプ: REALまたはSTRING = VARIANT 当社で、x ≥ 20のレジスタを割り当て済みです。

レジスタ値の説明

レジスタへの値の割り当てはメソッドで設定します。

注記
LM機能で既存の対話画面から新しい対話画面が作成された場合、同時にレジスタの内容が新しい対話画面に自動的に転送され、引き続き第2の対話画面で計算に使用することができます。

プログラミング

構文:	*識別子*.val = *レジスタ値* または *識別子*= *レジスタ値*	
説明:		
パラメータ:	識別子	レジスタの名称
	レジスタ値	レジスタの値

例

```
UNLOAD
  REG[0] = VAR1           ; 変数1の値をレジスタ0に割り当てます。
END_UNLOAD


UNLOAD
  REG[9].VAL = 84         ; 値84をレジスタ9に割り当てます。
END_UNLOAD
                          ;
                          これらのレジスタはその次の対話画面のメソッドで再度ローカル変数に割り当
                          てることができます。
LOAD
  VAR2 = REG[0]
END_LOAD
```

レジスタ状態の説明

状態プロパティを使用して、レジスタが有効な内容であるかどうかをスキャンできます。

レジスタスキャン機能のひとつの用途として、当該対話画面が「マスター対話画面」である場合のみ、レジスタに値を書き込むようにできます。

プログラミング

構文:	*識別子*.vld	
説明:	このプロパティは読み出し専用です。	
パラメータ:	識別子	レジスタの名称
戻り値:		スキャン結果は下記のようになります。
	FALSE =	無効な値
	TRUE =	有効な値

例

```
IF REG[15].VLD == FALSE     ; レジスタ値の有効性をスキャンします。
  REG[15] = 84
ENDIF
VAR1 = REG[9].VLD           ; REG[9]状態要求の値をVar1に割り当てます
```

6.3.22 RETURN

概要

RETURN機能を使用して、現在のサブプログラムを途中で終了し、最後のCALL命令の分岐点に戻ることができます。

サブプログラムにRETURN命令が設定されていない場合、サブプログラムは最後まで運転された後に分岐点に戻ります。

プログラミング

構文:	**RETURN**	
説明:	分岐点に戻ります。	
パラメータ:	- なし -	

例

```
//B(PROG1)                ; ブロックの開始
SUB(UP2)                  ; サブプログラムの開始
  IF VAR1.val=="Otto"
    VAR1.val="Hans"
    RETURN                ; 変数値 =
                          Ottoの場合、値「Hans」が変数に割り当てられ、サブプログラムはこの位置
                          で終了します。
  ENDIF
  VAR1.val="Otto"         ; 変数値 ≠ Ottoの場合、値「Otto」が変数に割り当てられます。
END_SUB                   ; サブプログラム終了
//END                     ; ブロック終了
```

6.3.23 再コンパイル

概要

プログラミングサポートシステムでは、GC機能で作成されたNCコードを**再コンパイル**すること、その後、対応する対話画面項目の入力/出力フィールドに変数値を再度表示することができます。

プログラミング

NCコードからの変数は対話画面に転送されます。
同時に、NCコードからの変数値は設定ファイルからの計算された変数値と比較されます。
値が一致しない場合、NCコード作成のときに値が変更されているため、エラーメッセージがログブックに書き込まれます。

NCコードが同じ変数を複数回含む場合、再コンパイルのときに最後に作成された位置で評価されます。 警告もログブックに書き込まれます。

コード作成のときにNCコードで使用されない変数は、ユーザーコメントとして設定されます。
「ユーザーコメント」とはコードの再コンパイルに必要となる全ての情報です。
ユーザーコメントは変更してはいけません。

注記

NCコードとユーザーコメントを構成要素とするブロックが、行の先頭で始まる場合のみ、再コンパイルすることができます。

例:

プログラムは下記のNCコードを含みます。

```
DEF VAR1=(I//101)
OUTPUT(CODE1)
  "X" VAR1 " Y200"
  "X" VAR1 " Y0"
END_OUTPUT
```

下記のコードがパートプログラムに設定されます。

```
;NCG#TestGC#\cus.dir\aeditor.com#CODE1#1#3#
X101 Y200
X101 Y0
;#END#
```

エディタは再コンパイルのときに下記を読み出します。

```
X101 Y200
X222 Y0          ; パートプログラムでXの値が変更されました (X101 → X222)。
```

入力対話画面でVAR1の下記の値が表示されます: `VAR1 = 222`

下記も参照

コードの作成(GC) (ページ 129)

6.3.24 コメントなしの再コンパイル

概要

プログラミングサポートシステムでは、GC機能で作成されたNCコードを**コメントなしで再コンパイル**し、対応する入力対話画面の入力/出力フィールドに変数値を再度表示することができます。

プログラミング

GC命令は、標準のコード生成で作成されるコメント行をマスクするために、次の方法で実行できます。

```
GC("CODE1",D_NAME,1)
```

通常、上の命令の結果のコードは再コンパイルできません。
この方法で生成されたサイクル呼び出しを再コンパイルするには、以下のステップが必要です。

- **easyscreen.iniの拡張**

  セクション[RECOMPILE_INFO_FILES]をeasyscreen.iniファイルに挿入します。
  このセクションでは、コメントなしで再コンパイルされたサイクルの記述を含むすべてのiniファイルがリスト表示されます。

```
 [RECOMPILE_INFO_FILES]

IniFile01 = cycles1.ini

IniFile02 = cycles2.ini
```

  名称が自由に選択可能な複数のiniファイルを指定できます。

- **サイクル記述用iniファイルの作成**

  サイクル記述付きのiniファイルは、ディレクトリ/sinumerik/hmi/cfgの/userまたは/oemに保存されます。 サイクルごとに、別のセクションが必要です。
  セクション名称は、サイクル名称に対応します。

```
[Cycle123]

Mname = TestGC

Dname = testgc.com

OUTPUT = Code1

Anzp = 3

Version = 0

Code_type = 1

Icon = cycle123.png

Desc_Text = This is describing text
```

Mname	画面名称
Dname	画面を定義するファイルの名前
OUTPUT	それぞれの出力ブロックの名称
Anzp	再コンパイルする画面のパラメータの数(DEF作成変数のすべてとヘルプ変数)
バージョン	(オプション)サイクルのバージョン仕様
Icon	(オプション)加工ステッププログラムの表示アイコン、フォーマット*.png 対応する解像度の画面サイズ 640 X 480 mm → 16 x 16ピクセル 800 X 600 mm → 20 x 20ピクセル 1024 X 768 mm → 26 x 26ピクセル 1280 X 1024 mm → 26 x 26ピクセル 1280 X 768 mm → 26 x 26ピクセル ファイルの位置: /sinumerik/hmi/ico/ico<解像度> 注: 1280の解像度の場合、1024 x 768 mm用のフォルダが使用されます(加工ステッププログラムだけに対応)。
Desc_Text	(オプション)加工ステッププログラムの表示用説明テキスト、最大17文字列(加工ステッププログラムだけに対応)

例

```
//M(TestGC/"Code generation:")
DEF VAR1 = (R//1)
DEF VAR2 = (R//2)
DEF D_NAME
LOAD
  VAR1 = 123
  VAR2 = -6
END_LOAD
OUTPUT(CODE1)
  "Cycle123(" VAR1 "," VAR2 ")"
  "M30"
END_OUTPUT

PRESS(VS1)
  D_NAME = "\MPF.DIR\MESSEN.MPF"
  GC("CODE1",D_NAME)                    ;
                                        OUTPUTメソッドからNCコードをファイル\MPF.DIR
                                        \MESSEN.MPFに書き込みます。
                                        Cycle123(123, -6)
                                        M30

END_PRESS
```

下記も参照

コードの作成(GC) (ページ 129)

6.3.25 前方に検索、後方に検索(SF、SB)

概要

SF、SB(Search Forward, Search Backward(前方に検索、後方に検索))
機能を使用して、エディタで現在選択しているNCプログラムの現在のカーソル位置から文字列を検索したり、その値を出力したりできます。

プログラミング

構文:	**SF***("文字列")*	
識別子:	**前方に検索**: 現在のカーソル位置から前方に検索します	
構文:	**SB***("文字列")*	
識別子:	**後方に検索**: 現在のカーソル位置から後方に検索します	
パラメータ:	文字列	検索するテキスト

テキスト検索の規則

- 現在選択しているNCプログラムで、検索文字列とその値を構成要素とする検索キーの単位の前後にスペースを挿入してください。
- システムはコメントテキストの中、またはその他の文字列の途中では検索キーを検索しません。
- 出力する値は数値式です。「X1=4+5」形式の式は認識されません。
- システムは、X1='HFFFF'形式の16進数の定数、X1='B10010'形式の2進数の定数、X1='-.5EX-4'形式の指数式を認識します。
- 文字列と値の間に下記が含まれる場合、文字列の値を出力することができます。
  - なし
  - ブランク
  - 等号記号

例

下記の表記ができます。

```
X100 Y200                ; 変数Abcに値200が割り当てられます。
Abc = SB("Y")
X100 Y 200               ; 変数Abcに値200が割り当てられます。
Abc = SB("Y")
X100 Y=200               ; 変数Abcに値200が割り当てられます。
Abc = SB("Y")
```

6.3.26 STRING機能

概要

下記の機能で文字列を処理することができます。

- 文字列の長さを特定
- 文字列内の文字の検索
- 左からの部分文字列の抽出
- 右からの部分文字の抽出
- 文字列の中からの部分文字列の抽出
- 部分文字列の置換

LEN機能: 文字列の長さ

構文:	**LEN***(文字列 | varname)*	
説明:	文字列内の文字数を特定します。	
パラメータ:	文字列	すべての有効な文字列式。 文字列がブランクの場合、NULLが出力されます。
	varname	任意の有効な宣言済みの変数名称
	2つのパラメータのうち、どちらかのパラメータだけでも使用できます。	

例

```
DEF VAR01
DEF VAR02

LOAD
  VAR01="HALLO"
  VAR02=LEN(VAR01)           ; 結果 = 5
END_LOAD
```

INSTR機能: 文字列内の文字の検索

構文:	**INSTR***(スタート, 文字列1, 文字列2 [,方向])*	
説明:	文字数を検索します。	
パラメータ:	スタート	文字列1を文字列2のなかで検索するときの開始位置 文字列2の先頭で検索を開始する場合は0を入力してください。
	文字列1	検索される文字
	文字列2	検索がおこなわれる一連の文字
	方向(任意選択)	検索の方向 0: 左から右へ(初期設定) 1: 右から左へ
	文字列1が文字列2に存在しない場合、0が返されます。	

例

```
DEF VAR01
DEF VAR02

LOAD
  VAR01="HELLO/WORLD"
  VAR02=INST(1,"/",VAR01)        ; 結果 = 6
END_LOAD
```

LEFT機能: 左からの文字列

構文:	**LEFT***(文字列, 長さ)*	
説明:	LEFTは、文字列の左側から始めて、指定された文字数を含む文字列を返します。	
パラメータ:	文字列	文字列、または処理される文字列を含む変数
	長さ	読み出す文字数

例

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="HELLO/WORLD"
  VAR02=LEFT(VAR01,5)           ; 結果 = 「HELLO」
END_LOAD
```

RIGHT機能: 右からの文字列

構文:	**RIGHT***(文字列, 長さ)*	
説明:	RIGHTは、文字列の右側から始めて、指定された文字数を含む文字列を返します。	
パラメータ:	文字列	文字列、または処理される文字列を含む変数
	長さ	読み出す文字数

例

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="HELLO/WORLD"
  VAR02=LEFT(VAR01,4)          ; 結果 = 「WORLD」
END_LOAD
```

MIDS機能: 文字列の中から文字列

構文:	**MIDS***(文字列, スタート[, 長さ])*	
説明:	MIDSは、文字列内の指定された位置から始まる、指定された文字数を含む文字列を返します。	
パラメータ:	文字列	文字列、または処理される文字列を含む変数
	スタート	文字列内で文字を読み出す開始位置
	長さ	読み出す文字数

例

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="HELLO/WORLD"
  VAR02=LEFT(VAR01,4,4)            ; 結果 = 「LO/W」
END_LOAD
```

REPLACE機能: 文字の置換

構文:	**REPLACE***(文字列, FindString, ReplaceString [, スタート [, カウント]])*	
説明:	REPLACE機能は、文字列内の文字/文字列を他の文字/文字列で置換します。	
パラメータ:	文字列	この文字列のなかのFindString(文字列)を検索して、それをReplaceString(文字列)で置換
	FindString	置換される文字列
	ReplaceString	置換する文字列(FindStringの代わりにこれを使用)
	スタート	検索と置換をおこなう開始位置
	カウント	FindString (文字列)の開始位置からの検索文字数
戻り値:		
	文字列 = ブランク文字列	文字列のコピー
	FindString = ブランク文字列	文字列のコピー
	ReplaceString = ブランク文字列	見つかったFindString(文字列)をすべて削除した文字列のコピー
	スタート > Len(文字列)	ブランク文字列
	カウント = 0	文字列のコピー

下記も参照

文字列の用途 (ページ 83)

6.3.27 PIサービス

概要

PI_SERVICE機能を使用して、NC領域で、PLCからPIサービス(PI Services:Program Invocation Services(プログラム起動サービス))をスタートできます。

一般的なプログラミング

構文:	**PI_SERVICE** (*サービス, n個のパラメータ*)	
説明:	PIサービスを実行します。	
パラメータ:	サービス	PIサービスの識別子
	n個のパラメータ	PIサービスのn個のパラメータのリスト。 個々のパラメータはコンマで区切られます。

例

```
PRESS (HS2)
  PI_SERVICE("_N_CREATO",55)
END_PRESS
PRESS(VS4)
  PI_SERVICE("_N_CRCEDN",17,3)
END_PRESS
```

OEMサービスの開始

PI_START命令は、OEMマニュアルに基づいてPIサービスを実行します。

プログラミング

構文:	PI_START("*転送文字列*")	
説明:	PIサービスを実行します。	
パラメータ:	"転送文字列"	OEMマニュアルとは異なり、転送文字列はダブルクウォーテーションマークで囲んで入力します。

例

```
PI_START("/NC,001,_N_LOGOUT")
```

注記

チャネル関連のPIサービスは常に現在のチャネルに適用されます。

ツール機能(TOエリア)のPIサービスは常に現在のチャネルに割り当てられたTOエリアを参照します。

グラフィックおよびロジック項目 7

7.1 線と長方形

概要

線と長方形をLOADブロックに設定します。

- 最初に線が引かれ、続いて長方形、最後に、設定された制御フィールドまたはグラフィックが作成されます。
- システムの背景色に塗りつぶし色を設定することで、透き通った長方形が作成されます。

LINEの要素

プログラミング

構文:	LINE (x1,y1,x2,y2,f,s)	
説明:	線を定義します。	
パラメータ:	x1	開始点のx座標
	y1	開始点のy座標
	x2	終点のx座標
	y2	終点のy座標
	f	線の色
	s	線種: 1 = 実線 2 = 破線 3 = 点線 4 = 一点鎖線

RECTの要素

プログラミング

構文:	RECT (x,y,w,h,f1,f2,s)	
説明:	長方形を定義します。	
パラメータ:	x	左上のx座標
	y	左上のy座標
	w	幅
	h	高さ
	f1	境界線の色
	f2	塗りつぶしの色
	s	境界線種
		1 = 実線
		2 = 破線
		3 = 点線
		4 = 一点鎖線

下記を参照してください。

LOAD (ページ 102)

7.2 配列の定義

定義

配列を使用して、インデックスでデータにアクセスする方法で、メモリに保存された同一データタイプのデータを整理することができます。

概要

配列は1次元または2次元とすることができます。
1次元の配列は、1行または1列だけの2次元の配列として扱われます。

配列には開始識別子 //Aと終了識別子 //ENDがあります。
行と列の数は任意に選択できます。 配列の構成は下記の通りです。

プログラミング

構文:	*//A(識別子)* (a/b...) (c/d...) ... //END	
説明:	配列を定義します。	
パラメータ:	識別子	配列の名称
	a, b, c, d	配列の値 STRINGタイプの値はダブルクウォーテーションマークで囲んでください。

例

```
//A(Thread)                          ; サイズ/リード/ねじの谷径
(0.3 / 0.075 / 0.202)
(0.4 / 0.1   / 0.270)
(0.5 / 0.125 / 0.338)
(0.6 / 0.15  / 0.406)
(0.8 / 0.2   / 0.540)
(1.0 / 0.25  / 0.676)
(1.2 / 0.25  / 0.676)
(1.4 / 0.3   / 1.010)
(1.7 / 0.35  / 1.246)
//END
```

7.2.1 配列要素の値へのアクセス

概要

配列へのアクセス操作はプロパティ値(識別子.val)で渡すことができます。

行インデックス(配列の行番号)と列インデックス(配列の列番号)はともに0から始まります。行インデックスまたは列インデックスが配列の外側になる場合、値0またはブラン

ク文字列が出力され、ERR変数がTRUEに設定されます。
検索キーが見つからなかった場合も、変数ERRはTRUEに設定されます。

プログラミング

構文:	識別子 [Z,[M[,C]]].valまたは 識別子 [Z,[M[,C]]]		
説明:	1列の1次元配列へアクセスします。		
構文:	識別子 [S,[M[,C]]].val]または 識別子 [S,[M[,C]]]、あるいは		
説明:	1行の1次元配列へアクセスします。		
構文:	識別子 [Z,S,[M[,C]]].valまたは 識別子 [Z,S,[M[,C]]]		
説明:	2次元配列へアクセスします。		
パラメータ:	識別子:	配列の名称	
	Z:	行の値(行インデックスまたは検索キー)	
	S:	列の値(列インデックスまたは検索キー)	
	M:	アクセスモード	
		0	直接
		1	行、列の直接検索
		2	列、行の直接検索
		3	検索
		4	行インデックスの検索
		5	列インデックスの検索
	C:	比較モード	
		0	検索キーは行または列の値の範囲内になければなりません。
		1	検索キーは正確な位置になければなりません。

例:	`VAR1 = MET_G[REG[3],1,0].VAL`	`; Var1に配列MET_Gの値を割り当てます。`

アクセスモード

- **「直接」アクセスモード**

  「直接」アクセスモード(M = 0)では、行インデックスZと列インデックスSで配列にアクセスします。比較モードCは評価されません。

- **「検索」アクセスモード**

  アクセスモードM = 1、2、3の場合、検索は常に行0または列0から開始されます。

モードM	行の値Z	列の値S	出力値
0	行インデックス	列インデックス	行Z、列Sの値
1	検索キー 列0を検索	値が読み出される列の列インデックス	見つかった行と列Sからの値
2	戻り値が読み出される行の行インデックス	検索キー 行0を検索	行Zと見つかった列からの値
3	検索キー 列0を検索	検索キー 行0を検索	見つかった行と列からの値
4	検索キー 列Sを検索	検索列の列インデックス	行インデックス
5	検索行の行インデックス	検索キー 行Zを検索	列インデックス

比較モード

比較モードC = 0を使用する場合、検索行または検索列の内容は昇順に並べられていなければなりません。
検索キーが最初の要素より小さい、または最後の要素より大きい場合、値0またはブランク文字列が出力され、エラー変数ERRはTRUEに設定されます。

比較モードC = 1を使用する場合、検索キーが検索行または検索列に存在していなくてはなりません。
検索キーが見つからなかった場合、値0または空の文字列が出力され、エラー変数ERRはTRUEに設定されます。

7.2.2 配列要素へのアクセス例

前提条件

以下に2つの配列が定義されています。 これを基に下記の例を説明します。

```
//A(Thread)
                    (0.3 / 0.075 / 0.202)
                    (0.4 / 0.1   / 0.270)
                    (0.5 / 0.125 / 0.338)
                    (0.6 / 0.15  / 0.406)
                    (0.8 / 0.2   / 0.540)
                    (1.0 / 0.25  / 0.676)
                    (1.2 / 0.25  / 0.676)
                    (1.4 / 0.3   / 1.010)
                    (1.7 / 0.35  / 1.246)
//END


//A(Array2)
                    ("DES" /        "PTCH" /      "CDM" )
                    (0.3 /          0.075 /       0.202 )
                    (0.4 /          0.1 /         0.270 )
                    (0.5 /          0.125 /       0.338 )
                    (0.6 /          0.15 /        0.406 )
                    (0.8 /          0.2 /         0.540 )
                    (1.0 /          0.25 /        0.676 )
                    (1.2 /          0.25 /        0.676 )
                    (1.4 /          0.3 /         1.010 )
                    (1.7 /          0.35 /        1.246 )
//END
```

例

- **アクセスモード例1**

  検索キーはZにあります。このキーは常に列0で検索されます。検索キーが見つかった行インデックスの列Sの値が出力されます。

  ```
  VAR1 = Thread[0.5,1,1]   ; VAR1の値は0.125
  ```

  意味

  「Thread」配列の列0で値0.5を検索し、同じ行の列1で見つかった値を出力します。

- **アクセスモード例2**

  検索キーはSにあります。この検索キーは常に行0で検索されます。検索キーが見つかった列インデックスの行Zの値が出力されます。

  ```
  VAR1 = ARRAY2[3,"PTCH",2]   ; VAR1の値は0.125
  ```

  意味

  配列「Array2」の行0で「PTCH」を含む列を検索します。
  見つかった列の、インデックス3の行にある値を出力します。

- **アクセスモード例3**

  検索キーはそれぞれZとSにあります。列0に検索キーZをもつ行インデックスが検索され、行0にSの検索キーをもつ列インデックスが検索されます。見つかった行インデックスと列インデックスにある値が出力されます。

  ```
  VAR1 = ARRAY2[0.6,"PTCH",3]   ; VAR1の値は0.15
  ```

  意味

  配列「Array2」の列0で内容0.6をもつ行が検索され、Array2の行0で内容「STG」をもつ列が検索されます。 見つかった行と列にある値がVAR1に転送されます。

- **アクセスモード例4**

  検索キーはZにあります。Sは検索キーが検索される列の列インデックスを示します。 検索キーが見つかった行インデックスが出力されます。

  ```
  VAR1 = Thread[0.125,1,4]   ; VAR1の値は2
  ```

  意味

  「Thread」配列の列1で値0.125を検索し、値が見つかった行インデックスをVAR1に割り当てます。

- **アクセスモード例5**

  Zは検索キーが検索される行の行インデックスを示します。
  検索キーはSにあります。検索キーが見つかった列インデックスが出力されます。

  ```
  VAR1 = Thread[4,0.2,5,1]  ; VAR1の値は1
  ```

  意味

  「Thread」配列の行4で値0.2を検索し、値が見つかった列インデックスをVAR1に割り当てます。
  行4の値が昇順に並べられていないため、比較モード1が選択されました。

7.2.3 配列要素の状態のスキャン

概要

状態プロパティを使用して、配列アクセス操作が有効な値を与えているかどうかをスキャンできます。

プログラミング

構文:	*識別子 [Z, S, [M[,C]]]*.vld
説明:	このプロパティは読み出し専用です。
パラメータ:	識別子　　配列の名称
戻り値:	FALSE =無効な値
	TRUE =有効な値

例

```
DEF MPIT = (R///"MPIT",,"MPIT",""/wr3)
DEF PIT  = (R///"PIT",,"PIT",""/wr3)
PRESS(VS1)
  MPIT = 0.6
  IF MET_G[MPIT,0,4,1].VLD == TRUE
    PIT    = MET_G[MPIT,1,0].VAL
    REG[4] = PIT
    REG[1] = "OK"
  ELSE
    REG[1] = "ERROR"
  ENDIF
END_PRESS
```

7.3 テーブルグリッド(表)

定義

配列とは異なり、テーブルグリッド(表)の値は常時更新されます。
これは、1つのチャネルの1つのブロックを使用してアドレス指定できるシステム変数値の表形式の表示をおこないます。

割り当て

変数の定義は表識別子で表の要素定義に割り当てられます。

- 変数の定義が表示される値を特定し、表要素の定義が画面ウインドウの表示と配置を特定します。
  テーブルグリッドは変数定義行からI/Oフィールドのプロパティを取り込みます。
- 表の表示エリアはI/Oフィールドの幅と高さによって決まります。
  見えない行または列は、水平または垂直にスクロールすれば表示できます。

表識別子

チャネルブロックでアドレス指定が可能な、同じタイプのNCK/PLC値を含んだ表の識別子。

表識別子は前に%記号を付けることで、制限または切り替えフィールドと区別します。表記述を含むファイルは、識別子の後ろにコンマを付け、ファイル名称を挿入することで指定することができます。

システム変数またはユーザー変数

列定義行が表示する変数についての詳細情報を含むので、このパラメータはテーブルグリッドに対しては空のままです。 表記述はダイナミックに与えることができます。

説明

変数定義には表記述への参照を含みます。

DEF *識別子* =	**識別子 = 変数名称**
	変数タイプ
	/[制限または切り替えフィールドまたは表識別子]
	/[初期設定]
	/[テキスト(ロングテキスト, ショートテキスト|イメージ, グラフィックテキスト, 単位テキスト)]
	/[属性]
	/[ヘルプ表示]
	/[システム変数またはユーザー変数]
	/[ショートテキストの位置]
	/[入力/出力フィールドの位置(左, 上, 幅, 高さ)]
	/[色]

下記を参照してください。

変数パラメータ (ページ 71)

7.3.1 テーブルグリッドの定義

概要

表のブロックは下記の構成です。

- ヘッダー
- 1 ～ nの列記述

プログラミング

構文:	***//G****(表識別子/表タイプ/行数/[固定行属性],[固定列属性])*		
説明:	テーブルグリッドを定義します。		
パラメータ:	表識別子	この表識別子は前に%記号を付けないで使用します。 これは対話画面では一回しか使用できません。	
	表タイプ	0 (初期設定)	PLCまたはユーザーデータ(NCK別データとチャネル別データ)用の表
		1	および、その他:予備
	行数	ヘッダーを含む行数	
		固定行または固定列はスクロールされません。 列数は設定済です。	
	固定行の属性	1: 0:	有効 無効
	固定列の属性	1: 0:	有効 無効

7.3.2 列の定義

概要

テーブルグリッドでは、インデックス付き変数を使用することを推奨します。
PLCまたはNC変数の場合、1つ以上のインデックスが存在する場合に、インデックス番号は有意味です。

グリッドに表示される値は、属性によって許可された権限の制約と定義された制限の中で、エンドユーザーが直接変更できます。

プログラミング

構文:	*(タイプ/制限/空/ロングテキスト,列ヘッダー/属性/ヘルプ表示/システム変数またはユーザー変数/列幅/オフセット1, オフセット2, オフセット3)*	
説明:	列を定義します。	
パラメータ:	変数に類似しています	
	タイプ	データタイプ
	制限	最小値、最大値
	ロングテキスト, 列ヘッダー	
	属性	
	ヘルプ表示	
	システム変数またはユーザー変数	変数として、PLCまたはNC変数は、ダブルクォーテーションマークで囲んで入力します。
	列幅	ピクセル単位で指定

	オフセット	列を満たすために各インデックスを増やすインクリメントサイズは、割り当てられたオフセットパラメータに指定します。 • オフセット1: 第1のインデックスのステップ幅 • オフセット2: 第2のインデックスのステップ幅 • オフセット3: 第3のインデックスのステップ幅

STRINGタイプの変数

変数がSTRINGタイプの場合、長さはタイプに指定します。たとえば、次のように指定します。

```
DEF CHAN STRING [16] TEXT[41]
```

その結果、CHAN変数の列定義はたとえば(S16/...)で始まります。

テキストファイルからの列ヘッダー

列ヘッダーはテキストまたはテキスト番号($8xxxx)で入力します。列ヘッダーはスクロールされません。

列のプロパティの変更

ダイナミックに変更(書き込み)できる列のプロパティは次の通りです。

- 制限値(最小値、最大値)
- 列ヘッダー(st)
- 属性(wr、ac、li)
- ヘルプ表示(hlp)
- OPI変数(var)

列のプロパティは、定義行の変数識別子と列インデックス(1から開始)を使用して変更します。

例: `VAR1[1].st="Column 1"`

列のプロパティはLOADブロックで読み出すことができません。

wr、ac、liの属性は列の定義用に指定することができます。

7.3.3 テーブルグリッドのフォーカス制御

概要

行と列のプロパティを使用して、表の中のフォーカスの設定と計算をおこなうことができます。

- 識別子.**Row**
- 識別子.**Col**

プログラミング

表の各セルにはValプロパティとVldプロパティがあります。

セルのプロパティの読み出しと書き込みのために、定義リストからの変数識別子に加えて、行と列のインデックスを指定してください。

構文:	識別子[行インデックス, 列インデックス].valまたは 識別子[行インデックス, 列インデックス]
説明:	Valプロパティです。
構文:	識別子[行インデックス, 列インデックス].vld
説明:	Vldプロパティです。

例

```
Var1[2,3].val=1.203
```

行と列のインデックスが指定されない場合、フォーカスされたセルのインデックスが適用されます。 これは下記に対応します。

```
Var1.Row =2

Var1.Col=3

Var1.val=1.203
```

7.4 カスタムウィジェット

7.4.1 カスタムウィジェットの定義

概要

ユーザー固有の表示項目は、対話画面でカスタムウィジェットを使用して設定されます。

ソフトウェアオプション

ダイアログボックスでカスタムウィジェットを使用するには、以下の追加ソフトウェアオプションが必要です。

「SINUMERIK Integrate Run MyHMI /3GL」(6FC5800-0AP60-0YB0)

プログラミング

定義:	**DEF***(名称)*	
構文:	**(W///"","***(ライブラリ名称)*.*(クラス名称)***"/////***a*,*b*,*c*,*d*);**	
説明:	W	カスタムウィジェットの定義
パラメータ:	名称	カスタムウィジェット名称、自由に選択可能
	ライブラリ名称	自由に選択可能、dll (Windows)またはライブラリファイル(Linux)の名称
	クラス名称	自由に選択可能、上記で名前が指定されたライブラリのクラス機能の名称
	a, b, c, d	設定の位置とサイズ

例

カスタムウィジェットは、設定対話画面で、次のように定義されます。

```
DEF Cus = (W///"","slestestcustomwidget.SlEsTestCustomWidget"/////20,20,250,100);
```

7.4.2 カスタムウィジェットライブラリの構造

概要

基本的に、カスタムウィジェットライブラリには定義されたクラスが含まれます。
設定対話画面では、ライブラリ名称に加えて、このクラスの名称を指定してください。
「Run MyScreens」では、ライブラリ名称から始めて、たとえば、次のような同じ名称を持つdllファイルにアクセスします。

```
slestestcustomwidget.dll
```

プログラミング

dllファイルのクラス定義は、次のようにおこなわれます。

```
#define SLESTESTCUSTOMWIDGET_EXPORT Q_DECL_EXPORT


class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget : public QWidget
{
 Q_OBJECT
....
....
}
```

7.4.3 カスタムウィジェットインタフェースの構造

概要

対話画面でカスタムウィジェットを表示するために、ライブラリはインタフェースによって補われます。
このインタフェースには、「Run MyScreens」がカスタムウィジェットを開始するマクロ定義が含まれています。 このインタフェースは、cppファイル形式で使用できます。
ファイル名称は、たとえば次のように自由に選択可能です。
`sleswidgetfactory.cpp`

プログラミング

このインタフェースは、以下のように定義されます。

```
#include "slestestcustomwidget.h"      ;
                                       該当カスタムウィジェットのヘッダーファイルが、ファイルの先頭に挿入されます。
....
//Makros                               ; マクロ定義は変更されません。
....
WIDGET_CLASS_EXPORT(SlEsTestCustomWi   ;
dget)                                  該当カスタムウィジェットは、ファイルの最後で宣言されます。
```

例

クラス名称がSlEsTestCustomWidgetのカスタムウィジェットファイルsleswidgetfactory.cppの内容

```
#include <Qt/qglobal.h>
#include "slestestcustomwidget.h"

////////////////////////////////////////////////////////////////////////
// MAKROS FOR PLUGIN DLL-EXPORT - DO NOT CHANGE
////////////////////////////////////////////////////////////////////////

#ifndef Q_EXTERN_C
#ifdef __cplusplus
#define Q_EXTERN_C extern "C"
#else
#define Q_EXTERN_C extern
#endif
#endif

#define SL_ES_FCT_NAME(PLUGIN) sl_es_create_ ##PLUGIN
#define SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( IMPLEMENTATION , PARAM) \
 { \
 IMPLEMENTATION *i = new PARAM; \
 return i; \
 }

#ifdef Q_WS_WIN
# ifdef Q_CC_BOR
# define EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(PLUGIN,PARAM) \
 Q_EXTERN_C __declspec(dllexport) void* \
 __stdcall SL_ES_FCT_NAME(PLUGIN) (QWidget* pParent) \
 SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( PLUGIN,PARAM )
# else
# define EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(PLUGIN,PARAM) \
 Q_EXTERN_C __declspec(dllexport) void* SL_ES_FCT_NAME(PLUGIN) \
 (QWidget* pParent) \
 SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( PLUGIN,PARAM )
# endif
#else
```

```
# define EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(PLUGIN,PARAM) \
 Q_EXTERN_C void* SL_ES_FCT_NAME(PLUGIN) (QWidget* pParent) \
 SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( PLUGIN,PARAM )
#endif

#define WIDGET_CLASS_EXPORT(CLASSNAME) \
 EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(CLASSNAME,CLASSNAME(pParent))

////////////////////////////////////////////////////////////////////////
// FOR OEM USER - please declare here your widget classes for export
////////////////////////////////////////////////////////////////////////

WIDGET_CLASS_EXPORT(SlEsTestCustomWidget)
```

7.4.4 カスタムウィジェットとダイアログの間の相互作用

概要

カスタムウィジェットはダイアログボックスと相互作用し、値の表示または操作をおこなうことができます。 この結果、以下の条件の場合に、データが交換されます。

条件	方向
対話画面の起動または再コンパイルをおこなうとき	対話画面 → カスタムウィジェット
サイクル呼び出しを生成するためにGC命令を実行するとき	カスタムウィジェット → 対話画面

プログラミング

相互作用をおこなうには、以下の定義が必要です。

設定対話画面の拡張

定義:	**DEF** *(変数)*	
構文:	**(***(タイプ)***//5/"","***(変数)***","/wr2/)**	
変数タイプ:	タイプ	任意のデータタイプ(Wなし)の標準入力フィールド(グリッドおよび切り替えなし)
パラメータ:	変数	データ交換用変数の任意の名称
入力モード:	wr2	読み出しと書き込み

例

```
DEF CUSVAR1 = (R//5/"","CUSVAR1",""/wr2/)
```

クラス定義の拡張

カスタムウィジェットのクラス定義では、たとえば次に示すように、名称が設定対話画面の選択した変数と同一であるQPropertyを作成してください。

```
Q_PROPERTY(double CUSVAR1 READ cusVar1 WRITE setCusVar1);
```

例

dllファイルのクラス定義は、次のようにおこなわれます。

```
#define SLESTESTCUSTOMWIDGET_EXPORT Q_DECL_EXPORT


class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget : public QWidget
{
 Q_OBJECT
 Q_PROPERTY(double CUSVAR1 READ cusVar1 WRITE setCusVar1);
....
....
}
```

「Custom」操作エリア 8

8.1 「Custom」操作エリアの有効化の方法

「Custom」操作エリアの有効化

「Custom」操作エリアは出荷時には有効になっていません。

1. まず、slamconfig.iniファイルを/siemens/sinumerik/hmi/templates/cfgから/oem/sinumerik/hmi/cfg
   ディレクトリに、またはこれに応じて/addon/sinumerik/hmi/cfgまたは/user/sinumerik/hmi/cfgディレクトリにコピーします。
2. 「Custom」操作エリアを有効にするには、下記を入力します。

```
[Custom]

Visible=True
```

結果

有効化が完了すると、「Custom」操作エリアのソフトキーが、メインメニュー(F10)のメニュー連続バーのHSK4に表示されます(= 初期設定)。

「Custom」操作エリアは操作エリア全体をあらわす空のウインドウを設定可能なヘッダー付きで表示します。
すべての水平ソフトキーと垂直ソフトキーは設定することができます。

8.2 「Custom」ソフトキーの設定方法

「Custom」操作エリアのソフトキーの設定

「Custom」操作エリアのソフトキーの名称と位置はslamconfig.iniファイルに設定します。

スタートソフトキーを設定するために、下記の選択ができます。

1. ソフトキーラベルを**言語テキスト**で置き換えるには、[Custom]の箇所に下記のように入力します。

```
TextId=MY_TEXT_ID

TextFile=mytextfile

TextContext=mycontext
```

この例では、ソフトキーは「MyContext」の下のmytextfile_xxx.qm
テキストファイル
(xxxは言語コードを表します)に「MY_TEXT_IDのテキストIDで設定された言語テキストを示します。

2. ソフトキーラベルを**言語中立テキスト**で置き換えるには、[Custom]の区間に下記のように入力します。

```
TextId=HELLO

TextFile=<empty>

TextContext=<empty>
```

この例では、「Custom」操作エリアのソフトキーはすべての言語に対して「HELLO」テキストを表示します。

3. ソフトキーにはテキストだけでなく**アイコン**も表示できます。

   このためには、[Custom]の箇所に下記のように入力します。

   ```
   Picture=mypicture.png
   ```

   ソフトキーはmypicture.pngファイルからのアイコンを表示します。グラフィックとビットマップは下記のパスに保存されています: /oem/sinumerik/hmi/ico/ico<Resolution>。 表示解像度に対応したディレクトリを使用してください。

4. ソフトキーの**位置**も設定することができます。
   [Custom]の箇所に下記を入力することで、この設定ができます。

   ```
   SoftkeyPosition=12
   ```

   初期設定は位置12です。これは、操作エリアメニューのメニュー連続バーのHSK4に対応します。 位置1 ～ 8はメニューバーのHSK1 ～ HSK8に対応し、位置9 ～ 16はメニュー連続バーのHSK1 ～ HSK8に対応します。

8.3 「Custom」操作エリアの設定方法

「Custom」操作エリアのソフトキーの設定

操作エリアを設定するにはeasyscreen.iniファイルとcustom.iniファイルが必要です。この2つのファイルのテンプレートは/siemens/sinumerik/hmi/templates/cfgディレクトリにあります。

1. まず、ファイルを/oem/sinumerik/hmi/cfgディレクトリにコピーして、変更を加えます。
2. easyscreen.iniファイルにはすでに「Custom」操作エリア用の定義行があります。

```
;StartFile02 = area := Custom, dialog := SlEsCustomDialog, startfile := custom.com
```

行の先頭の「;」はコメント文字を示します。
これは、この行がコメントであり、そのため有効ではないことを意味します。
これを変更するには、「;」を削除します。

この行の「startfile」属性を使用して、「Custom」操作エリアを選択したとき、この項目が、custom.comのプロジェクトファイルを示すように定義します。

3. /oem/sinumerik/hmi/projディレクトリに**custom.comのプロジェクトファイル**を作成します。
これは当該の設定を含みます。この設定は、「プログラム」操作エリアのaeditor.comファイルと同じ方法で作成されます。
設定したスタートソフトキーは「Custom」操作エリアに表示されます。
4. custom.iniファイルの対話画面のタイトルバー用に**言語中立テキスト**を設定します。

下記の項目はこの目的のために、テンプレートにあるものを使用できます。

```
[Header]Text=Custom
```

このテキストを、カスタマイズしたテキストで置き換えることができます。

5. テンプレートには「Custom」操作エリア用の**初期画面**を設定するための下記の項目が含まれています。

```
[Picture]Picture=logo.png
```

Logo.pngは「Custom」操作エリアの対話開始画面に表示される初期画面の名称です。 ここで、たとえば会社名やその他のイメージを表示することができます。
このファイルは次の解像度用ディレクトリに保存してください。
/oem/sinumerik/hmi/ico/ …

8.4 「Custom」エリアのプログラミング例

例

図 8-1 「Start example」ソフトキーの例

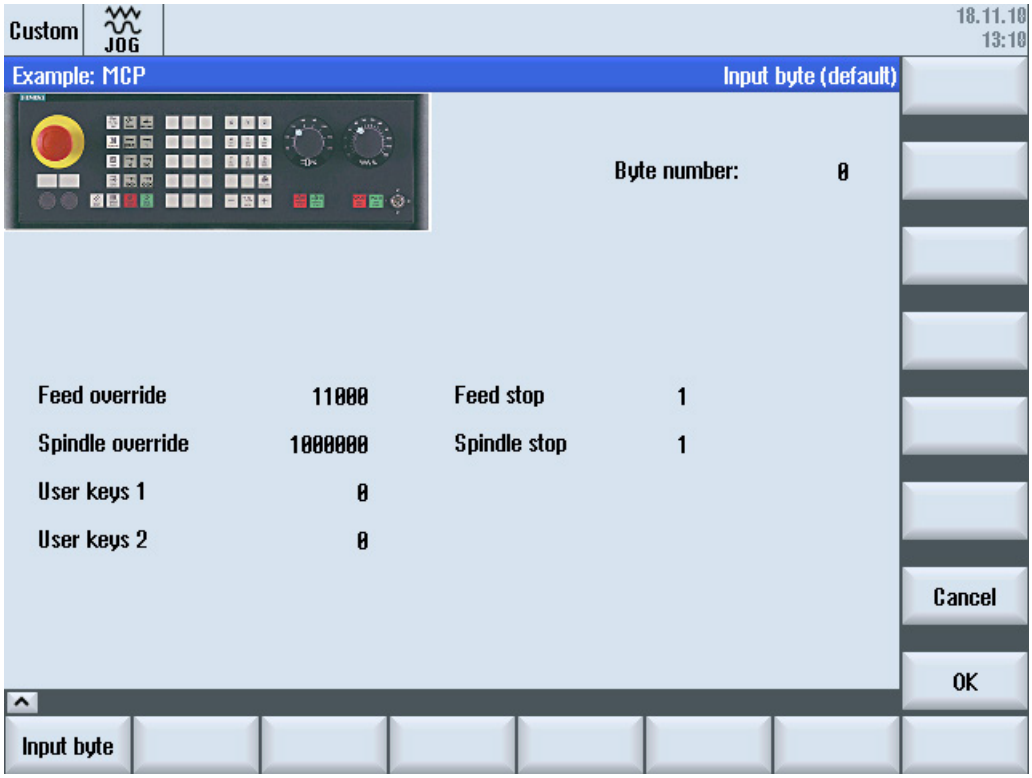


図 8-2　　ビットマップとテキストフィールドの例

ファイル一覧

以下のファイルが必要です。

- custom.com
- easyscreen.ini

プログラミング

custom.comファイルの内容

注記

例のなかの組み込まれたグラフィックファイルmcp.pngは、単なるサンプルファイルです。
このプログラミング例を自身の用途に使用する場合、このグラフィックを別に用意したもので置き替えてください。

```
//S(Start)
 HS7=("Start example", se1, ac7)
 PRESS(HS7)
 LM("Maske4")
 END_PRESS
//END
//M(Maske4/"Example: MCP"/"mcp.png")
 DEF byte=(I/0/0/"Input byte=0 (default)","Byte number:",""/wr1,li1///380,40,100/480,40,50)
 DEF Feed=(IBB//0/"","Feed override",""/wr1//"EB3"/20,180,100/130,180,100), Axistop=(B//0/"","Feed
stop",""/wr1//"E2.2"/280,180,100/380,180,50/100)
 DEF Spin=(IBB//0/"","Spindle override",""/wr1//"EB0"/20,210,100/130,210,100),
spinstop=(B//0/"","Spindle stop",""/wr1//"E2.4"/280,210,100/380,210,50/100)
 DEF custom1=(IBB//0/"," User keys 1",""/wr1//"EB7.7"/20,240,100/130,240,100)
 DEF custom2=(IBB//0/"","User keys 2",""/wr1//"EB7.5"/20,270,100/130,270,100)
 DEF By1
 DEF By2
 DEF By3
 DEF By6
 DEF By7

 HS1=("Input byte", SE1, AC4)
 HS2=("")
 HS3=("")
 HS4=("")
 HS5=("")
 HS6=("")
 HS7=("")
 HS8=("")
 VS1=("")
 VS2=("")
 VS3=("")
 VS4=("")
 VS5=("")
 VS6=("")
 VS7=("Cancel", SE1, AC7)
 VS8=("OK", SE1, AC7)

 PRESS(VS7)
 EXIT
 END_PRESS
```

```
PRESS(VS8)
  EXIT
END_PRESS

LOAD
  By1=1
  By2=2
  By3=3
  By6=6
  By7=7
END_LOAD

PRESS(HS1)
  Byte.wr=2
END_PRESS

CHANGE(Byte)
  By1=byte+1
  By2=byte+2
  By3=byte+3
  By6=byte+6
  By7=byte+7
  Feed.VAR="EB"<<By3
  Spin.VAR="EB"<<Byte
  Custom1.VAR="EB"<<By6
  Custom2.VAR="EB"<<By7
  Axisstop.VAR="E"<<By2<<".2"
  Spinstop.VAR="E"<<By2<<".4"
  Byte.wr=1
END_CHANGE

CHANGE(Axis stop)
  IF Axistop==0
    Axistop.BC=9
  ELSE
    Axistop.BC=11
  ENDIF
END_CHANGE
```

```
 CHANGE(Spin stop)
  IF Spinstop==0
     Spinstop.BC=9
   ELSE
     Spinstop.BC=11
   ENDIF
 END_CHANGE
//END
```

対話画面選択

9

9.1 PLCソフトキーを使った対話画面選択

設定

手順の説明

- systemconfiguration.iniには[keyConfiguration]セクションがあります。
  この項目は特別なPLCソフトキーの動作を指定します。
- 番号は動作を表すものとして与えられます。
  番号が100以上の場合、「Run MyScreens」呼び出しが含まれます。
- 実行動作を定義するセクションをeasyscreen.iniファイルに作成します。
  セクションの名称は、操作エリア名称と対話画面名称に基づいて決めます([keyconfiguration]のArea:=...,
  Dialog:=...の項目を参照してください）。この設定は [<Area>_<Dialog>]
  に適用し、 たとえば、[AreaParameter_SIPaDialog]と設定します。
- 動作番号(これはsystemconfiguration.iniに設定されています→
  Action:=...を参照してください)がこのセクションに定義されます。
  ここには関連する2つの命令あります。

  1. LS("Softkey menu1","param.com") ... ソフトキーメニューの読み込み
  2. LM("Screen form1","param.com") ... 画面の読み込み

PLCソフトキーによるソフトキーメニューの選択

「Run MyScreens」ではPLCソフトキーで、「Run MyScreens」ソフトキーメニューと「Run MyScreens」対話画面を選択することができます。
当該のPLCソフトキーを設定する際に指定する「動作」属性の値が100以上の場合のみ、これが可能です。

PLCソフトキーは`systemconfiguration.ini`ファイルの[keyconfiguration]セクションで設定します。

```
[keyconfiguration]

KEY75.1 = Area:=area, Dialog:=dialog, Screen:=screen, Action:= 100,
```

当該のPLCソフトキーを押したときに実行されるLMとLS命令はeasyscreen.iniファイルに設定されます。 設定用に使用するセクション名称の構成は次の通りです。

[areaname_dialogname]	名称の最初の部分「areaname」は操作エリアを表し、2番目の部分「dialogname」はこのセクションで設定した命令が適用される対話画面を表します。
[AreaParameter_SlPaDialog] 100.screen1 = LS("Softkey1","param.com") 101.screen3 = LM("Screen form1","param.com")	操作エリアと対話画面用にsystemconfiguration.iniファイルに設定した名称を使用します。 対話画面は指定しなくても構いません。 これは特に、操作エリアが単一の対話画面で実装されるときだけ当てはまります。 左の例を参照してください。 SlPaDialog対話画面によって実装された操作エリアAreaParameterに「screen1」が表示される場合、値が100の「動作」が発生すると、「LS("Softkey1","param.com")」命令を実行します。
action.screen=Command	「action」と「screen」の属性はともに、指定された命令がいつ実行されるかを明確に示します。 「screen」の情報は任意に選択できます。 下記の命令を使用できます。 LM (LoadMask) LS (LoadSoftkeys)

9.2 PLCハードキーを使った対話画面選択

用途

PLCによって操作ソフトウェアで以下の機能を開始できます。

- 操作エリアの選択
- 操作エリア内における特定の場合の選択
- ソフトキーで設定した機能の実行

ハードキー

以下では、すべてのキー(PLCキーも)をハードキーと呼びます。
最大254個のハードキーを定義できます。 以下の割り当てが適用されます。

キー番号	用途
キー1 - キー9	操作パネルのキー
キー10 - キー49	予約済
キー50 - キー254 キー50 – キー81	PLCキー: OEM用に予約済み
キー255	制御情報でプリセット済み。

ハードキー1 - 9は、以下のようにプリセットされます。

キー名称		動作/働き
HK1	位置	[運転]操作エリア、最後の対話画面の選択
HK2	プログラム	[プログラム]操作エリア、最後の対話画面または最後のプログラムの選択
HK3	オフセット	[パラメータ]操作エリア、最後の対話画面の選択
HK4	プログラムマネージャ	機能なし
HK5	アラーム	アラーム診断操作エリア、[Alarm list]対話画面の選択

設定

この設定は、systemconfiguration.ini設定ファイルの[keyconfiguration]セクションでおこないます。 各行は、ハードキーイベントと呼ばれるものを定義します。
ハードキーイベントは、特定のハードキーのn番目の操作です。
たとえば、あるハードキーの2番目と3番目の操作で、異なる動作がおこなわれます。

systemconfiguration.ini設定ファイルのエントリは、ユーザー独自の設定で上書きできます。 \user\sinumerik\hmi\cfg and
\oem\sinumerik\hmi\cfgディレクトリは、この目的のために用意されています。

ハードキーイベントの設定のための行は、以下のような構造になっています。

```
KEYx.n = Area:=area、Dialog:=dialog、Screen:=screen、Forms:=form、Menus:=menu、
Action:=menu.action、Cmdline:=cmdline
KEYx.n = Area:=area、Dialog:=dialog、Cmdline:=cmdline、Action:= action
```

x: ハードキー番号、データ範囲:1 – 254

n: イベント番号 – ハードキーのn番目の操作に対応、数値の範囲:0 – 9

必要条件

PLCユーザープログラムは、以下の要求を満たさなければなりません。

1つのハードキーのみを処理すること。
その結果、操作ソフトウェアが前の要求を確認した場合のみ新規の要求が設定されます。
PLCユーザープログラムがMCPキーからハードキーを取得する場合、すばやいキーストロークも逃さないように、これにはキーの十分なバッファロケーションを用意してください。

PLCインタフェース

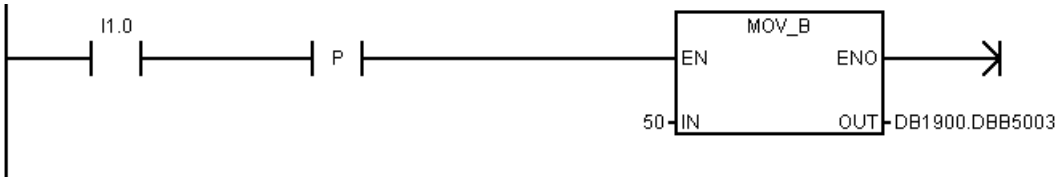
ハードキー選択のための操作エリアは、PLCインタフェースで提供されます。
この操作エリアは、DB1900.DBB5003にあります。
ここでは、PLCが50～254のキー値を直接指定することができます。

操作ソフトウェアによる確認は、2段階でおこなわれます。
この処理は、同一のキーコードが2回連続して入力された場合に、操作ソフトウェアが2つの別のイベントを特定するために必要です。
第1段階では、制御情報255がバイトDB1900.DBB5003に書き込まれます。
この定義された仮想キー起動によって、HMIがそれぞれのPLCキー処理を一義的に特定することができます。
制御情報は、PLCユーザープログラムには意味がありません。これを変更しないでください。
第2段階では、DB1900.DBB5003をクリアすることでPLCに対して実際の確認がおこなわれます。
この時点で、PLCユーザープログラムが新規のハードキーを指定できます。
これと同時に、実際のハードキー要求が操作ソフトウェアで処理されます。

例

[運転]操作エリアの選択:

1. PLCユーザープログラム



2. 設定ファイル

```
\oem\sinumerik\hmi\cfg\systemconfiguration.ini
```

```
; PLCハードキー(KEY50-KEY254)
[keyconfiguration]
KEY50.0 = area:=AreaMachine, dialog:=SlMachine
KEY51.0 = area:=AreaProgramEdit
KEY52.0 = area:=AreaParameter
KEY53.0 = area:=AreaProgramManager
KEY54.0 = area:=AreaDiagnosis, dialog:=SlDgDialog,
          cmdline:="-slGfwHmiScreen SlDgAeAlarmsScreen"
KEY55.0 = area:=Custom
KEY56.0 = area:=AreaStartup
KEY57.0 = area:=AreaStartup, dialog:=SlSuMDDialog
```

9.3 NCによる対話画面の選択

HMI OperateのMMC命令

以下の説明に従って、MMC命令を使用できます。

1. **MMC命令の定義**

   以下が、標準の`systemconfiguration.ini`ファイルにあります。 組合わせ:

   address:=MCYCLES --> command:=LM

   address:=CYCLES --> command:=PICTURE_ON

   計測サイクルとそれ以外のサイクルに分けるため、この区別が必要です。つまり、以下のようになります。

   – LMは常に計測サイクルに適用され、PICTURE_ONは常に非計測サイクルに適用されます。
   – 新たに定義されたMMC命令に、PICTURE_ONおよびLMという名称をつけることは許容されません。

2. **Run MyScreens - ライセンス**

   計測サイクルを除き、Run MyScreenによって開かれたすべての対話画面は、Run MyScreensライセンスが適用されます。つまり、これはライセンスがなければ、使用できるのは一定の対話画面のみであることを意味します。

   test.com (Run MyScreen)の呼び出し例:

```
g0 f50
MMC("CYCLES,PICTURE_ON,test.com,maske1","A")
m0
MMC("CYCLES,PICTURE_OFF","N")
M30
```

参照リスト

A

A.1 スタートソフトキー一覧

A.1.1 旋削用スタートソフトキー一覧

旋削用プログラム操作エリア

HSK1	HSK2	HSK3	HSK4	HSK5	HSK6	HSK7	HSK8
編集	**穴あけ**	**ターニング**	**輪郭ターニング**	**ミリング**	**その他**	**シミュレーション**	**NC選択**
HSK9	HSK10	HSK11	HSK12	HSK13	HSK14	HSK15	HSK16
--	直線 円 (ShopTurnのみ)	--	--	ワーク計測	工具計測	**OEM**	--

ターニング

以下の表に、旋削用途で可能なスタートソフトキーをリスト表示します。
個々のスタートソフトキーの割り当ては、個々のシステムに応じて異なる場合があります。 指定されたOEMソフトキーは、「Run MyScreens」で許可されます。

programGUIDE (Gコード)スタートソフトキー:

	穴あけ	ターニング	輪郭ターニング		ミリング		その他	
	HSK2	HSK3	HSK4		HSK5		HSK6	
VSK1	センタリング	切削	輪郭	--	正面削り	輪郭	設定	高速設定
VSK2	穴あけリーマ仕上	溝	切削	--	ポケット	輪郭切削	旋回面	平行軸
VSK3	深穴ドリル	アンダーカット	切削削り残し	--	スピゴット多角形	下穴あけ	旋回工具	--
VSK4	ボーリング	ねじ切り	溝切り	--	溝	ポケット	--	--
VSK5	ねじ切り	突っ切り	溝切り削り残し	--	ねじ切り	ポケット削り残し	--	--
VSK6	OEM	--	プランジ加工	--	彫刻	スピゴット	サブプログラム	--
VSK7	位置	OEM	プランジ加工削り残し	OEM	OEM	スピゴット削り残し	--	OEM
VSK8	繰返し位置	--	>>	<<	輪郭切削	<<	>>	<<

ShopTurnスタートソフトキー

	穴あけ	ターニング	輪郭ターニング		ミリング		その他		
	HSK2	HSK3	HSK4		HSK5		HSK6		HSK10
VSK1	センタ穴あけ	切削	新しい輪郭	--	正面削り	新しい輪郭	設定	高速設定	工具
VSK2	センタリング	溝	切削	--	ポケット	輪郭切削	旋回面	平行軸	直線
VSK3	穴あけリーマ仕上	アンダーカット	切削削り残し	--	スピゴット多角形	下穴あけ	旋回工具	プログラム繰り返し	円の中心
VSK4	深穴ドリル	ねじ切り	溝切り	--	溝	ポケット	対向主軸	--	円弧半径
VSK5	ねじ切り	突っ切り	溝切り削り残し	--	ねじ切り	ポケット削り残し	座標変換	--	極座標
VSK6	OEM	--	プランジ加工	--	彫刻	スピゴット	サブプログラム	--	逃げ/アプローチ
VSK7	位置	OEM	プランジ加工削り残し	OEM	OEM	スピゴット削り残し	--	OEM	--
VSK8	繰返し位置	--	>>	<<	輪郭切削	<<	>>	<<	--

下記も参照

スタートソフトキーの定義 (ページ 27)

A.1.2 フライス削り用スタートソフトキー一覧

フライス削り時のプログラム操作エリア

HSK1	HSK2	HSK3	HSK4	HSK5	HSK6	HSK7	HSK8
編集	**穴あけ**	**ミリング**	**輪郭切削**	**ターニング** (Gコードのみ)	**その他**	シミュレーション	**NC選択**
HSK9	HSK10	HSK11	HSK12	HSK13	HSK14	HSK15	HSK16
--	直線 円 (ShopMillのみ)	--	--	ワーク計測	工具計測	**OEM**	--

ミリング

以下の表に、フライス削り用途で可能なスタートソフトキーをリスト表示します。
個々のスタートソフトキーの割り当ては、個々のシステムに応じて異なる場合があります。 指定されたOEMソフトキーは、「Run MyScreens」で許可されます。

programGUIDE (Gコード)スタートソフトキー:

	穴あけ	ミリング	輪郭切削		ターニング		その他	
	HSK2	HSK3	HSK4		HSK5		HSK6	
VSK1	センタリング	正面削り	輪郭	--	切削	輪郭	設定	--
VSK2	穴あけリーマ仕上	ポケット	輪郭切削	--	溝	切削	旋回面	平行軸
VSK3	深穴ドリル	スピゴット多角形	下穴あけ	--	アンダーカット	切削削り残し	旋回工具	--
VSK4	ボーリング	溝	ポケット	--	ねじ切り	溝切り	高速設定	--
VSK5	ねじ切り	ねじ切り	ポケット削り残し	--	突っ切り	溝切り削り残し	--	--
VSK6	OEM	彫刻	スピゴット	--	--	プランジ加工	サブプログラム	--
VSK7	位置	OEM	スピゴット削り残し	OEM	OEM	プランジ加工削り残し	--	OEM
VSK8	繰返し位置	--	>>	<<	輪郭ターニング	<<	>>	<<

ShopMillスタートソフトキー:

	穴あけ	ミリング	輪郭切削		その他		直線円
	HSK2	HSK3	HSK4		HSK6		HSK10
VSK1	センタリング	正面削り	新しい輪郭	--	設定	--	工具
VSK2	穴あけリーマ仕上	ポケット	輪郭切削	--	旋回面	平行軸	直線
VSK3	深穴ドリル	スピゴット多角形	下穴あけ	--	旋回工具	プログラム繰り返し	円の中心
VSK4	ボーリング	溝	ポケット	--	高速設定	--	円弧半径
VSK5	ねじ切り	ねじ切り	ポケット削り残し	--	座標変換	--	ヘリカル
VSK6	OEM	彫刻	スピゴット	--	サブプログラム	--	極座標
VSK7	位置	OEM	スピゴット削り残し	OEM	--	OEM	--
VSK8	繰返し位置	--	>>	<<	>>	<<	--

A.2 色のリスト

システム色

対話画面設定用に同一のカラーテーブルを使用できます(各標準色のサブセットです)。
要素(テキスト、入力フィールド、背景等)の色は (0 ～
128のなかの)下記の選択肢から選択できます。

インデックス	ピクトグラム	色	色の説明
1		黒色	
2		オレンジ色	
3		暗緑色	
4		明灰色	
5		暗灰色	
6		青色	
7		赤色	
8		茶色	
9		黄色	
10		白色	
128		オレンジ色	システム色有効フィールド
129		明灰色	背景色
130		青色	ヘッダーの色(有効)
131		黒色	ヘッダーの前面色(有効)

A.3 ファイル名称に使用される言語コードのリスト

サポートされている言語

標準言語:

言語	ファイル名称の略語
中国語(簡体字)	chs
ドイツ語	deu
英語	eng
フランス語	fra
イタリア語	ita
スペイン語	esp

その他の言語

言語	ファイル名称の略語
中国語(繁体字)	cht
デンマーク語	dan
フィンランド語	fin
日本語	jpn
韓国語	kor
オランダ語	nld
ポーランド語	plk
ポルトガル語(ブラジル)	ptb
ルーマニア語	rom
ロシア語	rus
スウェーデン語	sve
スロバキア語	sky
チェコ語	csy
トルコ語	trk
ハンガリー語	hun

A.4 アクセス可能なシステム変数のリスト

参照先

リストマニュアルシステム変数/PGAsl/

下記も参照

複数のNC PLC読み出し(MRNP) (ページ 140)

用語集

PIサービス

NCで明確に定義された操作を実行する機能です。
PIサービスはPLCシステム、およびHMIシステムから呼び出すことができます。

PLCハードキー

ホットキー同様、HMIソフトウェアのPLCインタフェースで用意されています。
操作画面から起動することができる機能を設定できます。

これは、MCPキーまたはPLCユーザープログラムのPLC信号論理演算を使用した構成になります。 このため、「仮想のキー」と呼ばれます。

アクセスレベル

ユーザーの権限に応じて操作画面での機能のアクセスと利用の可能性を定義する、権限の階層化システムです。

イベント

→メソッドの実行を開始させる動作です: 文字の入力、ソフトキーの操作等です。

インタープリタ

→設定ファイルから定義されたコードを→
対話画面に自動的に変換し、その用途を制御します。

エディタ

文字をファイルに入力したり編集したりするためのASCIIエディタです。

グループ

→設定ファイルの再読み込み単位です。

シミュレーション

実際の機械軸を動かすことなく、→パートプログラム運転をシミュレーションします。

スタートソフトキー

新しく作成された最初の→対話画面開始用のソフトキーです。

ソフトキーメニュー

すべての水平ソフトキー、またはすべての垂直ソフトキーです。

ソフトキーラベル

ソフトキーに割り当てられる、画面上のテキスト/イメージです。

パートプログラム

軸の動作処理と各種の特殊動作を指定する、NC言語で作成されたプログラムです。

パラメータ

プログラム構文の変数要素です。これは→設定ファイルで他のワード/シンボルと置き換えられます。

プログラミングサポート

「上位」コンポーネントで→パートプログラムを書く際にプログラマを補助する→対話画面を提供します。

プロパティ

オブジェクトの特性(→変数の特性等)です。

ヘルプ変数

プロパティを割り当てることができない内部算術変数です。そのため対話画面に表示されません。

ホットキー

OP 010、OP
010Cの6個のキーとホットキーブロック付きのSINUMERIKキーボードです。
操作エリアを直接選択する場合にこれらのキーを押します。
オプションとして、2個の追加キーをホットキーとして設定できます。

メソッド

対応する→イベントが発生したときに実行される、プログラムで指令された操作処理です。

メニューツリー

相互にリンクされた→ 対話画面のグループです。

ユーザー変数

データブロック、または→パートプログラムに、ユーザーによって定義される変数です。

行インデックス

配列の行番号です。

再コンパイル

NCコード区間は、→プログラミングサポートシステムの→対話画面の入力フィールドから、→パートプログラムに作成することができます。
再コンパイルはこの逆の操作です。
NCコードの選択区間を作成するために使用される入力フィールドは、NCコードから検索されて、元の対話画面に表示されます。

切り替えフィールド

→入力/出力フィールド内の値のリスト; これを切り替えフィールドでチェックします。
フィールドに入力された値はリスト内の値の1つと同じでなければいけません。

設定ファイル

→対話画面の表示とその→機能を特定する定義と命令を含むファイルです。

属性

これでオブジェクト(→対話画面または→変数)に特定の→プロパティを割り当てます。

対話画面

→操作画面の表示です。

- **対話画面関連のソフトキーメニュー**

  新たに設定された対話画面から呼び出されるソフトキーメニューです。

- **対話画面に依存しないソフトキー**

  対話画面から呼び出されないソフトキーです。すなわち、ユーザーが新しい最初の対話画面の前に設定するスタートソフトキーとソフトキーメニューです。

定義行

→変数とソフトキーを定義するプログラム区間です。

入力/出力フィールド

またはI/Oフィールド: 変数値の入力または出力用です。

配列

配列を使用することによって、メモリに保存された標準データタイプのデータを、インデックスでデータにアクセスできるように構成することができます。

変数

→プロパティを割り当てることで→対話画面に表示可能な、そして入力データ、および算術演算の結果を入力可能なメモリのロケーションの名称です。

列インデックス

配列の列番号です。

索引

D

DLLファイル, 126

L

LINE (線の定義), 161

N

NCコードの生成, 129
NC変数
    書き込み, 139
    読み出し, 138

P

PIサービス, 109
PLCソフトキーの設定, 191
PLC変数
    書き込み, 139
    読み出し, 138

R

RECT(長方形の定義), 162

あ

アクセスレベル, 53
アラーム
    言語コード, 204

お

オンラインヘルプ, 59

か

カスタムウィジェット
    Library, 176
    インタフェース, 177
    相互作用, 179
    定義, 175

く

グラフィックテキスト, 72

さ

サブプログラム, 109
    キャンセル, 145
    ブロック識別子, 110
    呼び出し, 111
    変数, 110
サブ対話画面, 135

し

システム色, 203
システム変数, 64, 74
ショートテキスト, 72
ショートテキストの代わりのイメージの表示, 70

す

スタートソフトキー, 26, 27

そ

ソフトキー
プロパティ, 55
プロパティの割り当て, 51
ソフトキーメニューの定義, 51

て

テーブルグリッド
プログラミング, 171
定義, 169
列を定義します。, 172
テキスト, 72

ひ

ヒント欄, 72

ふ

ファイル
コピー, 114
移動, 119
削除, 115
フォーカス制御, 174

へ

ヘルプ表示, 73
ヘルプ変数, 63

ま

マスター対話画面, 135

め

メソッド
CHANGE, 99
LOAD, 102
LOAD GRID, 102
OUTPUT, 104
PRESS, 105
UNLOAD, 104
概要, 98
メニューツリー, 26

ゆ

ユーザー変数, 74

れ

レジスタ
データ交信, 143
状態, 144
値, 143

ろ

ロングテキスト, 72

漢字

位置
ショートテキスト, 74, 82
入力/出力フィールド, 74, 82
演算子
ビット, 96
算術, 93
加工ステップサポート, 148
機能
CALL(Subprogram
call(サブプログラム呼び出し)), 111
CP(Copy Program(プログラムのコピー)), 114
CVAR(Check Variable(変数のチェック)), 112
DLGL(Dialog line(対話画面行)), 121

DP(Delete Program(プログラムの削除)), 115
EP(Exist Program(プログラムの存在確認)), 117
EVAL(Evaluate(評価)), 122
EXIT, 123
EXITLS (EXIT Loading
Softkey(ソフトキー読み込みの終了)), 126
FCT, 126
GC(Generate code(コードの作成)), 129
INSTR(文字列), 154
LA(Load Array(配列の読み込み)), 133
LB(Load Block(ブロックの読み込み)), 134
LEFT(文字列), 155
LEN(文字列), 153
LM(対話画面の読み込み), 135
LS(Load Softkey(ソフトキーの読み込み)), 137
MIDS(文字列), 156
MP(Move Program(プログラムの移動)), 119
MRNP(Multiple Read NC PLC(複数のNC
PLC読み出し)), 140
NCコードの再コンパイル, 146
PI_SERVICE, 158
PI_START, 158
REPLACE(文字列), 157
RETURN(戻り), 145
RIGHT(文字列), 156
RNP(Read NC PLC Variable(NC
PLC変数の読み出し)), 138
SB (Search Backward(後方に検索)), 152
SF (Search Forward(前方に検索)), 152
SP(Select Program(プログラムの選択)), 120
WNP(Write NC PLC Variable(NC
PLC変数の書き込み)), 139
コメントなしの再コンパイル, 148
概要, 109
言語コード, 204
三角関数, 94
初期設定, 71
書き込みモード, 74
条件, 96
色, 74
数値フォーマット, 78
制限, 71
切り替えフィールド, 71, 79
設定ファイル, 24
前面色, 74
属性, 72
対話画面
プロパティ, 40
定義, 37
定義ブロック, 39
複数列, 48
対話画面変更モード, 135
対話画面要素, 45
単位のテキスト, 72
定数, 95
入力モード, 72
背景色, 74
配列
アクセスモード, 164
行インデックス, 164
状態, 168
定義, 162
比較モード, 164
要素, 163
列インデックス, 164
比較演算子, 95
表 → テーブルグリッド, 169
文字列, 83
変数
CURPOS, 85
CURVER, 85
ENTRY, 87

ERR, 88
FILE_ERR, 89
FOC, 90
S_CHAN, 91
チェック, 112
パラメータ, 71
プロパティの変更, 62
計算, 64
終了, 124
変数タイプ, 71
INTEGER, 75
VARIANT, 76
変数状態, 61
変数値, 61