SIEMENS

SINUMERIK 840Di sl/840D sl
SINUMERIK 810D/840D

补充操作界面

调试手册

11/2006
6FC5397-0DP10-0RA0

安全技术提示

为了您的人身安全以及避免财产损失，必须注意本手册中的提示。人身安全的提示用一个警告三角表示，仅与财产损失有关的提示不带警告三角。警告提示根据危险等级由高到低如下表示。

危险

表示如果不采取相应的小心措施，**将会**导致死亡或者严重的人身伤害。

警告

表示如果不采取相应的小心措施，**可能**导致死亡或者严重的人身伤害。

小心

带有警告三角，表示如果不采取相应的小心措施，可能导致轻微的人身伤害。

小心

不带警告三角，表示如果不采取相应的小心措施，可能导致财产损失。

注意

表示如果不注意相应的提示，可能会出现不希望的结果或状态。

当出现多个危险等级的情况下，每次总是使用最高等级的警告提示。如果在某个警告提示中带有警告可能导致人身伤害的警告三角，则可能在该警告提示中另外还附带有可能导致财产损失的警告。

合格的专业人员

仅允许安装和驱动与本文件相关的附属设备或系统。设备或系统的调试和运行仅允许由**合格的专业人员**进行。本文件安全技术提示中的合格专业人员是指根据安全技术标准具有从事进行设备、系统和电路的运行，接地和标识资格的人员。

按规定使用

请注意下列说明：

警告

设备仅允许用在目录和技术说明中规定的使用情况下，并且仅允许使用西门子股份有限公司推荐的或指定的其他制造商生产的设备和
部件。设备的正常和安全运行必须依赖于恰当的运输，合适的存储、安放和安装以及小心的操作和维修。

商标

所有带有标记符号 ®
的都是西门子股份有限公司的注册商标。标签中的其他符号可能是一些其他商标，这是出于保护所有者权利的目地由第三方使用而特别标示的。

责任免除

我们已对印刷品中所述内容与硬件和软件的一致性作过检查。然而不排除存在偏差的可能性，因此我们不保证印刷品中所述内容与硬件和软件完全一致。印刷品中的数据都按规定经过检测，必要的修正值包含在下一版本中。

Siemens AG
Automation and Drives
Postfach 48 48
90437 NÜRNBERG

6FC5397-0DP10-0RA0
11/2006

目录

1 功能范畴

概述

“补充操作界面”通过编译器和包含操作界面说明的设计文件实现。 通过 ASCII 文件配置“补充操作界面”： 该设计文件包含了关于操作界面的说明。
创建文件所需的句法参见下列章节。

通过“补充操作界面”方法可以设计显示机床制造商专用/最终用户专用的扩展功能的操作界面，或者仅设计自定义的对话框。
也可以改善或者更换由西门子或者机床制造商设计的操作界面。

编译器在 HMI 内置 sl、 ShopMill/ShopTurn 以及 HMI 高级上的使用方法一致。

例如，可通过新建的操作界面编辑零件程序。 或直接在控制系统上创建对话框。

前提条件

创建图形/图还需要使用图形程序。 在 HMI 内置 sl 上需要使用应用程序磁盘和 Paint Shop Pro (www.jasc.com)。
附随提供的工具盒中包含了设计新建对话框的实例，可以在此实例的基础上创建自定义的对话框。

使用

使用“补充操作界面”可以实现以下功能：

1. 显示对话框并提供：
   - 软键
   - 变量，表格
   - 文本和帮助文本
   - 图形和帮助画面
2. 通过以下方法调用对话框：
   - 按下（登入）软键
   - 选择 PLC
3. 动态重组对话框
   - 修改、删除软键
   - 定义并设计变量栏
   - 显示、更换、删除显示文本（和语言相关或无关）
   - 显示、更换、删除图形

4. 在进行以下操作时触发动作：
   – 显示对话框
   – 输入数值（变量）
   – 按下软键
   – 关闭对话框
5. 对话框间的数据交换
6. 变量
   – 读取（NC 变量、PLC 变量、用户变量）
   – 写入（NC 变量、PLC 变量、用户变量）
   – 和数学、比较或者逻辑运算符相连
7. 执行下列功能：
   – 子程序
   – 文件功能
   – PI 服务
   – 外部功能（HMI 高级）
8. 根据用户组考虑保护等级

边界条件

必须符合以下边界条件：

- 只允许在 HMI 操作区内交换对话框。
- 在 HMI 高级上，请求后才初始化用户、设定数据和机床数据。
- 用户变量不允许与系统变量或者 PLC 变量有相同的名称。
- 由 PLC 激活的对话框在 HMI 高级上构成自定义的操作区（和测量循环图类似）。

注意

通过“补充操作界面”可以建立编程支持（参见同名章节）以及建立西门子循环的操作界面。因此机床制造商或者最终用户可以在必要时并在上述说明的范围内修改编程支持和操作界面。

另见

章节“设计环境”中说明的设计文件。

编程

2

2.1 第一步

2.1.1 设计基础

设计文件

新操作界面的说明存储在设计文件中。 这些文件自动编译并显示屏幕上的结果。
在供货时并不提供设计文件，因此必须首先创建此文件。

创建设计文件时使用一个 ASCII 编辑器（例如记事本或者 HMI 编辑器）。

操作树的原理

多个相连的对话框构成了一个操作树。
如果能从一个对话框切换入另一个对话框，则表示这两个对话框间存在联系。
通过此对话框内重新定义的水平或者垂直软键可以返回上级对话框或者进入任意一个对话框。

在每个登入软键下都可以生成一个操作树：

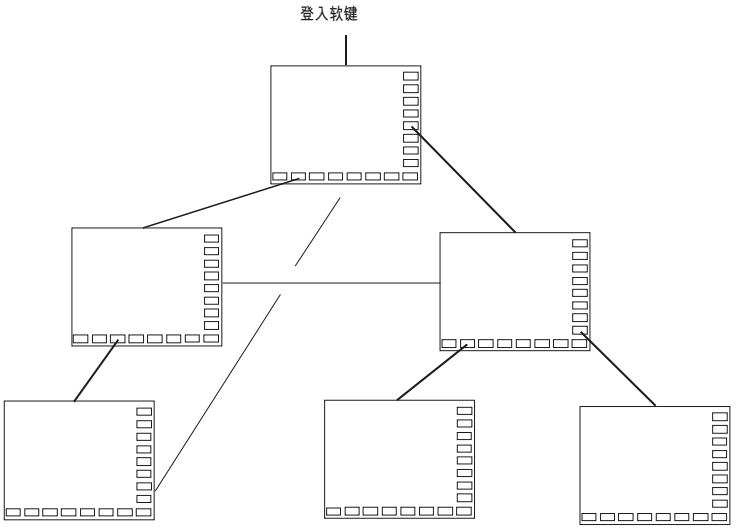

图 2-1 操作树

登入软键

在一个规定的设计文件中定义作为自身操作过程出发点的一个或者多个软键（登入软键）。

软键的定义决定执行下一步动作的自定义对话框或者其他软键栏。

按下登入软键则载入所属的对话框。 同时，属于对话框的相应软键激活。
如果没有设计特定位置，则在标准位置上给出变量。

返回到标准应用程序

可以退出新定义的操作界面并返回到标准应用程序。

通过<RECALL>（回调）键可以退出新定义的操作界面，如果这个按键还未被设计用于其它用途。

PLC 调用自定义对话框

除了软键，也可以由 PLC 选择对话框。 PLC 和 HMI 间存在用于信号交换的接口（在 DB19 中）。

参见

接口结构 (页 155)

设计登入软键 (页 181)

2.1.2 设计文件和登入文件

概述

每个应用程序都拥有固定（NCU 上的 HMI 内置 sl、ShopMill 和 ShopTurn)或者预设的登入软键，在该软键下新建的对话框形成树形图。 在 HMI 高级上可以设计其他登入软键。

其他文件：

调用设计文件中的“载入屏幕窗口” (LM) 或者“载入软键栏” (LS)可以重新命名已调用对象所在的文件。
采用这种方式可以划分设计，例如：一个自定义的设计文件同一操作级的所有功能。

创建设计文件为 ASCII 文件

对话框可以包含下列单元：

- 带下列文本的输入/输出栏（变量）
  - 短文本
  - 图形文本
  - 单位文本
- 图形
- 表
- 软键栏

设计文本的查找顺序

- **NCU 上的 HMI 内置 sl**

  在 NCU 上的 HMI 内置 sl 以及 ShopMill/ShopTurn 上，存取时会在 CF 卡相应的目录中查找设计文件。

  在 HMI 内置 sl 中的标准循环目录/用户循环目录下的文件 COMMON.COM 中可以设定，是否在每次存取时都重新查找设计文件（仅当在控制系统上直接创建对话框时比较重要），或者是否再次使用已经找到的或者保存在中间存储器中的设计文件（和标准操作情况一致）。

- **HMI 高级**

  在 HMI 高级 中，存取时首先在用户循环目录，然后在制造商循环目录，最后在标准循环目录中查找设计文件。

参见

查找方案的原理 (页 186)

2.1.3 设计文件的结构

概述

设计文件由以下单元组成：

- 登入软键说明
- 对话框定义
- 变量定义
- 块说明
- 软键栏定义

举例：

```
//S (START)                              ; 登入软键定义（可选）
....
//END
//M (.....)                              ; 对话框定义
DEF .....                                ; 变量定义
LOAD                                     ; 块说明
...
END_LOAD
UNLOAD
...
END_UNLOAD
ACTIVATE
...
END_ACTIVATE
...
//END
//S (...)                                ; 软键栏定义
//END
```

2.1.4 错误处理（日志）

概述

日志是记录解释句法时出现的错误消息的文件(Error.com)。
文件必须由操作员自己在注释目录中设立（HMI 高级）。

举例：

```
DEF VAR1 = (R)
DEF VAR2 = (R)
LOAD
VAR1 = VAR2 + 1                  ;  日志中的错误消息，因为 VAR2 没有数值
```

句法

当定义登入软键并且设计了带开始和结束标识的对话框和定义行后，才开始解释句法。

```
//S(Start)
HS6=(“第 1 屏幕窗口”)
PRESS(HS6)
  LM(“屏幕窗口 1”)
END_PRESS
//END

//M(屏幕窗口 1)
  DEF Var1=(R)
//END
```

ERROR.COM 的内容

如果“补充操作界面”解释定义文件时出现错误，则该错误保存在 ASCII 文件 ERROR.COM 中。

文件包含以下信息：

- 在执行何种动作时出现错误
- 第一个错误字符的行号和列号
- 设计文件中所有的错误行

如果借助 PC 测试环境创建对话框，则该文件位于环境变量 RAMDISK 参考的目录中 (HMI 内置 sl)。

仅当解释设计文件时确实出现错误，才创建文件 ERROR.COM。

文件 ERROR.COM 的保存路径：

- HMI 内置 sl： CF 卡上的目录 /tmp/hmiemb
- HMI 高级： 目录 \DH\COM.DIR\ 中

每次重新启动 HMI 内置 sl 和 HMI 高级时删除该文件。

显示文件 ERROR.COM

HMI 高级：

- 在操作区“通讯”或者“开机调试”中调用编辑器

HMI 内置 sl：

- 进入操作区“开机调试”→ “HMI” → “编辑器” → “临时驱动器”（垂直软键栏的第 4 个软键）。虽然使用制造商密码不能显示该软键，但软键仍作出反应（使用系统密码可显示软键）。
- 选中 ERROR.COM。
- 按下 <INPUT> 键。
- 按下软键“文件功能”可以通过设置的 Windows 网络驱动器将文件复制到 PC 上。

2.2 对话框的结构和组成单元

2.2.1 定义对话框

定义

对话框是操作界面上的一个组成部分，操作界面包含标题行、对话框单元和/或图形、显示消息的输出行以及 8 个水平软键和 8 个垂直软键。

对话框单元包含：

- 变量
  - 极限值
  - 变量预设值
- 帮助画面
- 文本
- 属性
- 系统或者用户变量
- 短文本的位置
- 输入/输出栏的位置
- 颜色
- 帮助(仅 HMI 高级)

对话框的属性：

- 标题
- 图形
- 尺寸
- 系统或者用户变量
- 图形的位置
- 属性

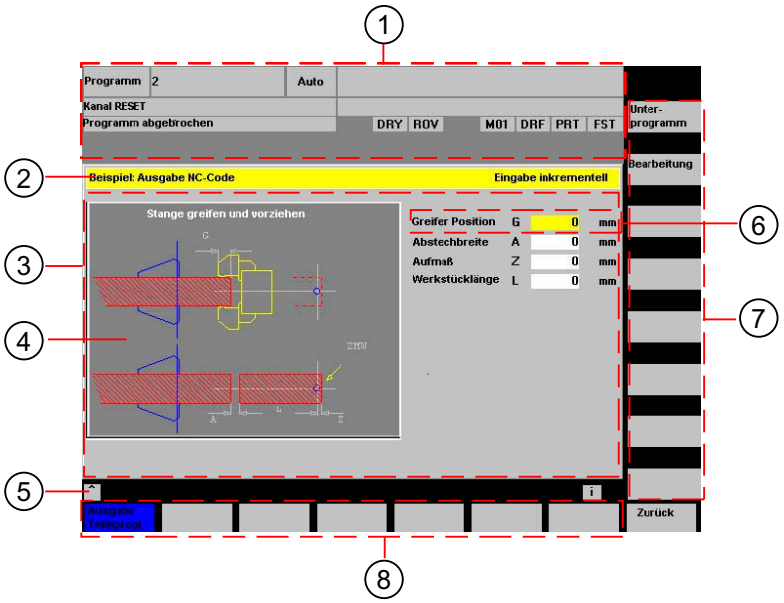

图 2-2　　对话框结构

1　机床状态显示（“标题”）
2　对话框的标题栏，包含标题和长文本
3　对话框
4　图形
5　显示消息的输出栏
6　对话框单元
7　8 个垂直软键
8　8 个水平软键

概述

基本上对话框说明（说明块）的结构如下：

说明块	注释	参考章节
//M...	; 对话框的开始标识	
DEF Var1=... ...	; 变量	参见章节“变量”
HS1=(...) ...	; 软键	参见章节“软键栏”
PRESS(HS1) LM... END_PRESS	; 方法的开始标识 ; 动作 ; 方法的结束标识	参见章节“方法”
//END	; 对话框的结束标识	

在对话框的说明块中，首先定义在对话框的对话框单元中显示的不同变量，然后定义水平和垂直软键。 然后在方法中设计不同的动作。

2.2.2 举例： 调用对话框

编程

进入操作区“参数”，按下登入软键“举例”可以调用一个新的对话框。

```
//S(Start)
HS7=(“举例”, ac7, se1)

PRESS(HS7)
  LM(“屏幕窗口 1”)
END_PRESS

//END
//M(屏幕窗口 1/“循环”)
HS1=("")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("")
VS8=("")
...                                          ;  方法
//END
```

结果

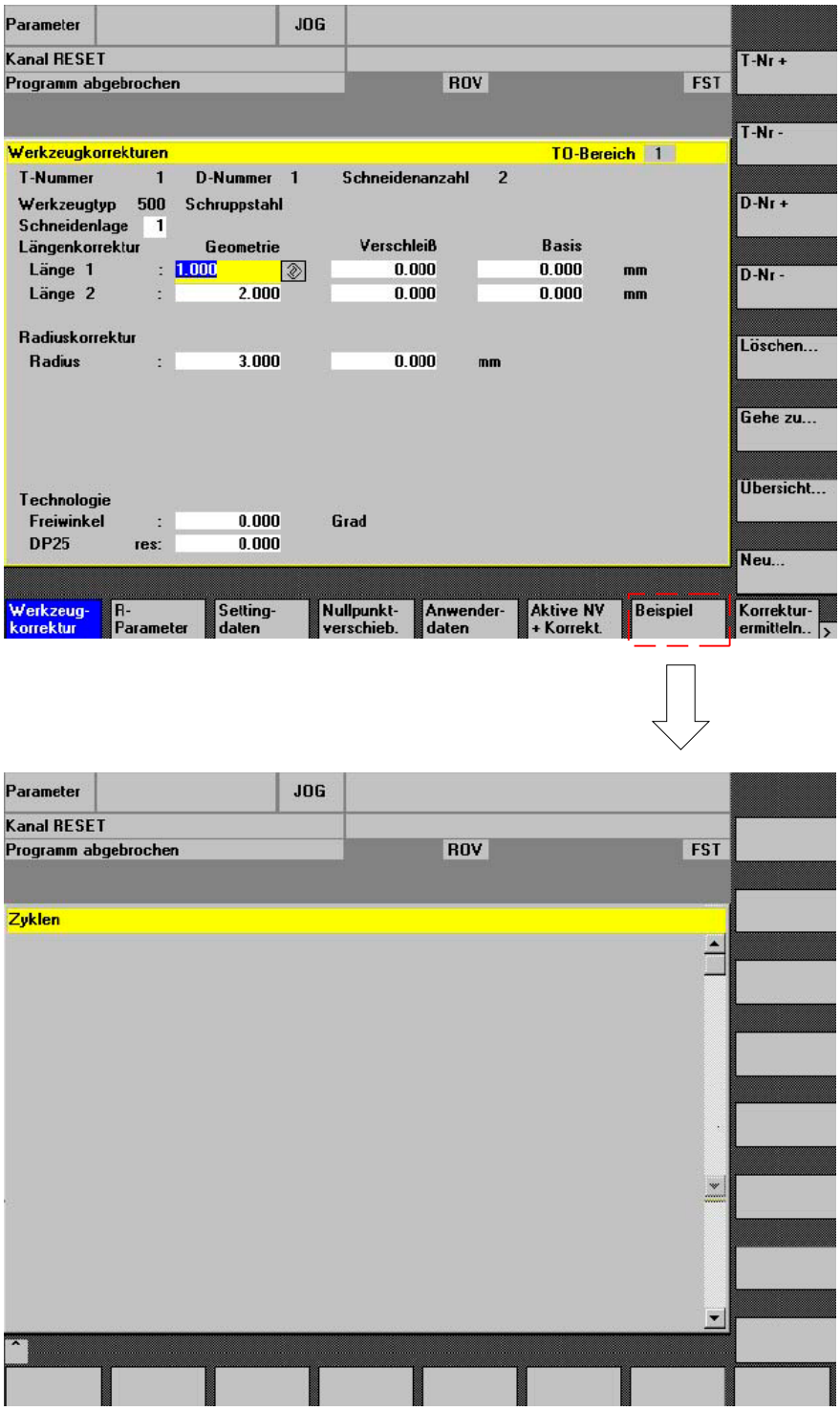

图 2-3　　举例： 按下登入软键“举例”调用对话框“循环”

2.2.3 对话框属性

说明

使用对话框的开始标识可以同时定义对话框的属性。

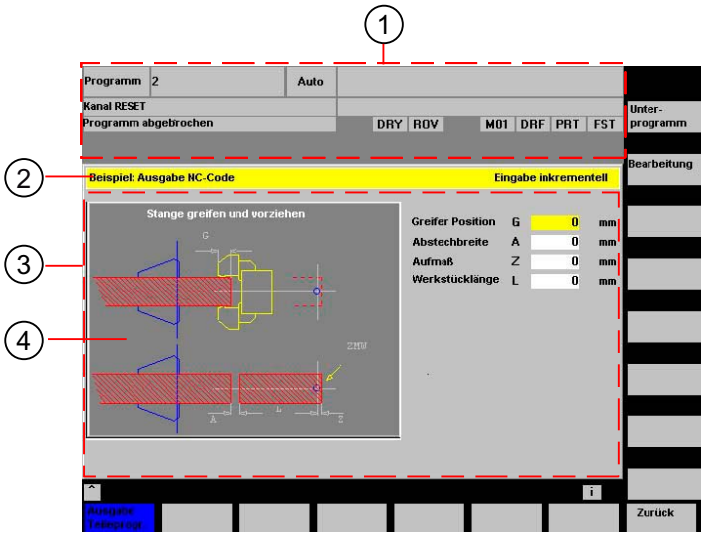

图 2-4 对话框属性

1 机床状态显示（“标题”）
2 对话框的标题栏，包含标题和长文本
3 对话框
4 图形

编程

句法： //M(名称/[标题]/[图形]/[尺寸]/[系统或用户变量]/[图形位置]/[属性])
说明： 定义对话框
参数：

名称	对话框的名称
标题	对话框的文本标题或者从和语言相关的文本文件中调用文本（例如：$85011）
图形	图形文件，路径在双引号内
尺寸	对话框的位置和大小，单位像素（和左/右边缘的间距、宽度、高度），以屏幕的左上角为基准。 数据值用逗号相隔。
系统或者用户变量	指定当前光标位置的系统和用户变量。 可以通过系统或者用户变量将光标位置传送给 NC 或 PLC。 第一个变量索引为 1。变量的顺序和变量设计顺序一致。

图形的位置		图形的位置，单位像素（和左/上边缘的间距），以对话框的左上角为基准。 和上边缘的最低间距为 18 像素。 数据值用逗号相隔。
属性		给定的属性用逗号相隔。
		允许的属性有：
CMx		Column Mode: 列对齐
	CM0	预设置： 每行单独分列。
	CM1	以包含最多列的行为标准分列。
CB		CHANGE 块: 打开对话框时的属性： 定义变量时规定的 cb 属性，优先于定义对话框时的总定义
	CB0	预设置： 在打开对话框时处理所有 CHANGE 部分。
	CB1	只有在附属的值改变后才处理 CHANGE 部分。
系统		在运行期间可读取“系统”的属性：
	0:	HMI_内置
	1:	HMI_高级

存取对话框属性

在方法（例如：PRESS 块）的范围内可以读取和写入对话框的以下属性：

- Hd = 标题
- Hlp = 帮助画面
- Var = 系统或者用户变量

举例

```
//S(Start)
HS7=(“举例”, se1, ac7)

PRESS(HS7)
  LM(“屏幕窗口 1”)
END_PRESS

//END
//M(屏幕窗口 1/“举例 2 : 显示图形”"MCP.BMP")
HS1=(“新%n标题”)
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("")
VS8=("")

PRESS(HS1)
  Hd= “新标题”
END_PRESS
...
//END
```

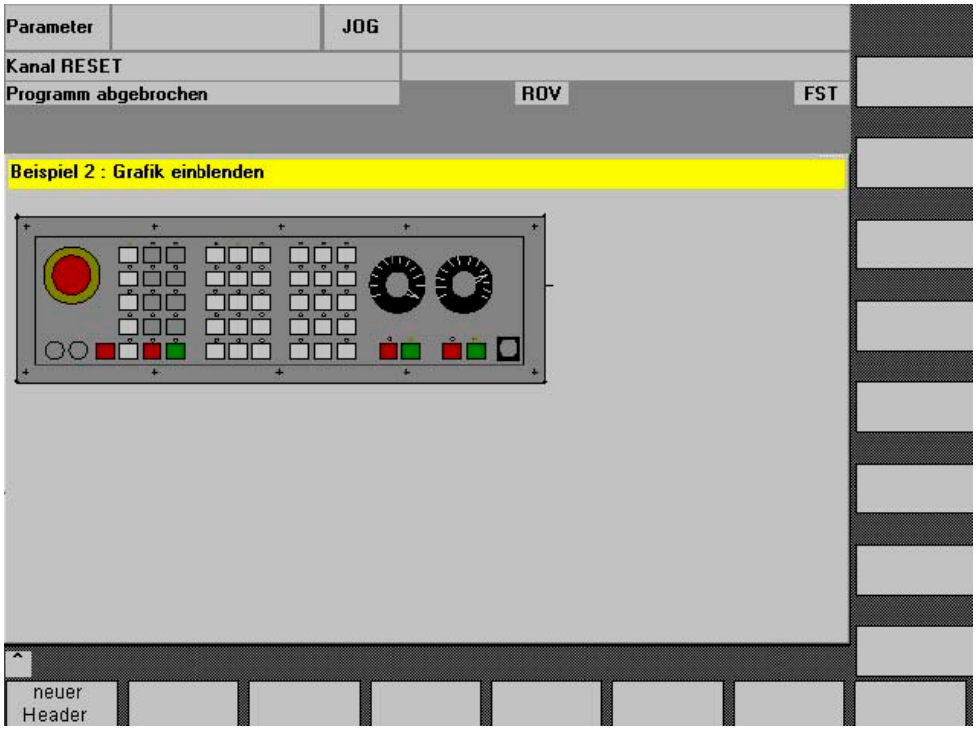

图 2-5 “举例 2： 显示图形”

参见

使用图/图形 (页 25)

和语言相关的文本 (页 183)

2.2.4 对话框单元

对话框单元

对话框单元是变量的可见部分，即短文本、图形文本、输入/输出栏和单位文本。
对话框单元位于对话框主体的行中。 每行可以定义一个或者多个对话框单元。

变量属性

所有变量仅在激活的对话框中有效。 通过定义变量指定其属性。 在方法（例如：PRESS 块）的范围内可以存取对话框属性值。

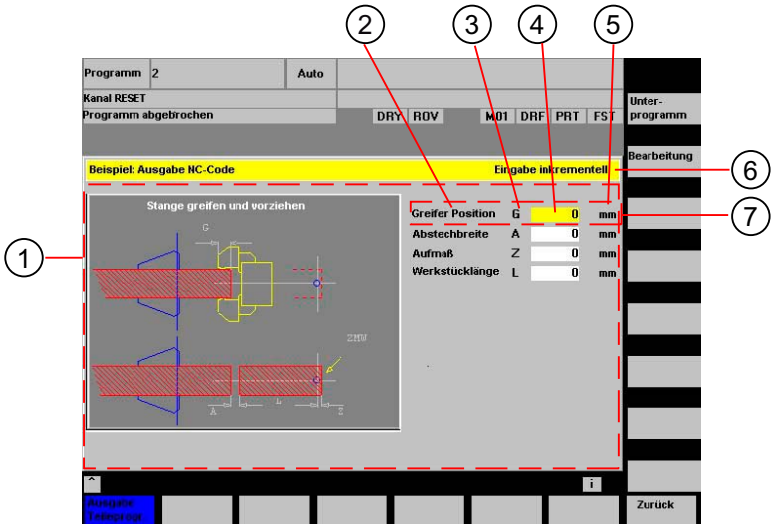

图 2-6　　对话框单元

1　　对话框主体
2　　短文本
3　　图形文本
4　　输入/输出栏
5　　单位文本
6　　对话框的标题栏，包含标题和长文本
7　　对话框单元

编程一览

在圆括号中通过逗号隔开各个参数。

DEF *Bezeichner* =	**Bezeichner = 变量名称**		
	变量类型	→	1
	/[极限值或者转换栏或者表格名称]	→	2
	/[预设值]	→	3
	/[文本（长文本，短文本|图，图形文本，单位文本）]	→	4
	/[属性]	→	5
	/[帮助画面]	→	6
	/[系统或者用户变量]	→	7
	/[短文本位置]	→	8
	/[输入/输出栏位置（左、上、宽度、高度）]	→	9
	/[颜色]	→	10
	/[帮助] （仅 HMI 高级）	→	11

参见

多列对话框 (页 23)

变量属性 (页 27)

2.2.5 多列对话框

概述

在对话框中，一行可以显示多个变量。
在这种情况下，设计文件中的所有变量都定义在一个定义行内。

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

为了可以更清除地显示设计文件中的各个变量，在每个变量定义和其紧随的逗号后可以换行。

关键字“DEF”总是表示新的一行的开始：

```
DEF Tnr1=(I//1/"","T ","/wr1///,,10/20,,50),
  TOP1=(I///,"Typ="/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)
...
```

在设计多列对话框时须注意所使用的硬件，例如： HMI 内置 sl 最多允许 10 列和 60 个 DEF 指令。

2.2.6 统一对话框显示图

边界条件

如果在操作面板同时安装了 HMI 高级和 ShopMill 或者 ShopTurn，则两个系统有不同字体类型。

HMI 高级为 Proporional Font（均衡字体），而 JobShop 产品以及 HMI 内置 sl 使用 Fixed Font（固定字体）。

如果在 HMI 内置 sl 和 HMI 高级上使用“补充操作界面”，则定义相同的对话框有不同的显示画面。

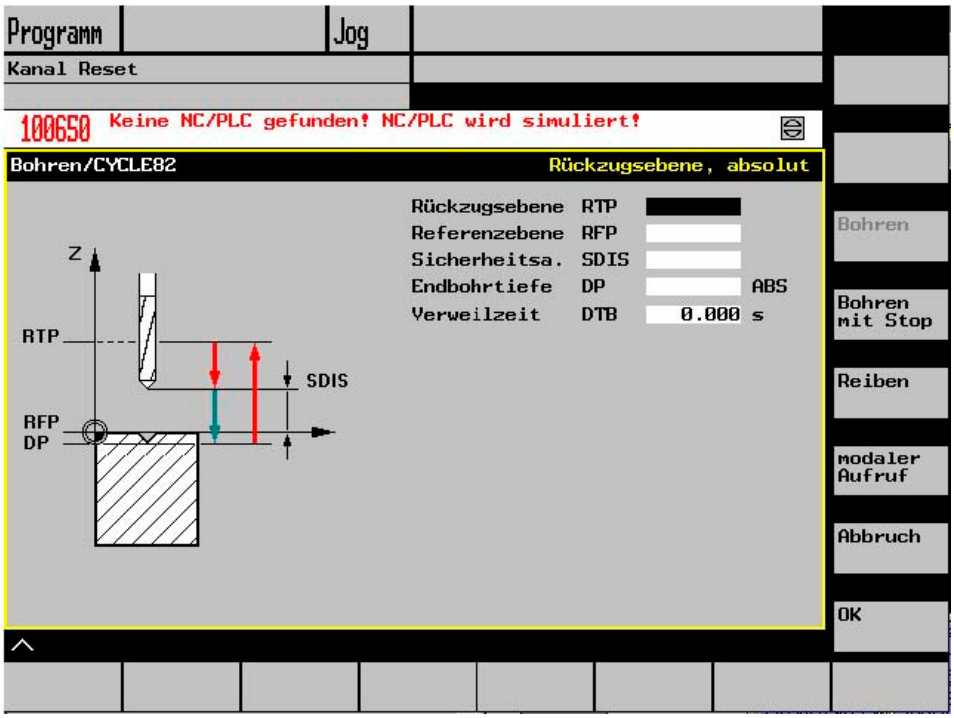

图 2-7 HMI 内置 sl 的显示画面

下图为图形内容相同的 HMI 高级下的显示画面。

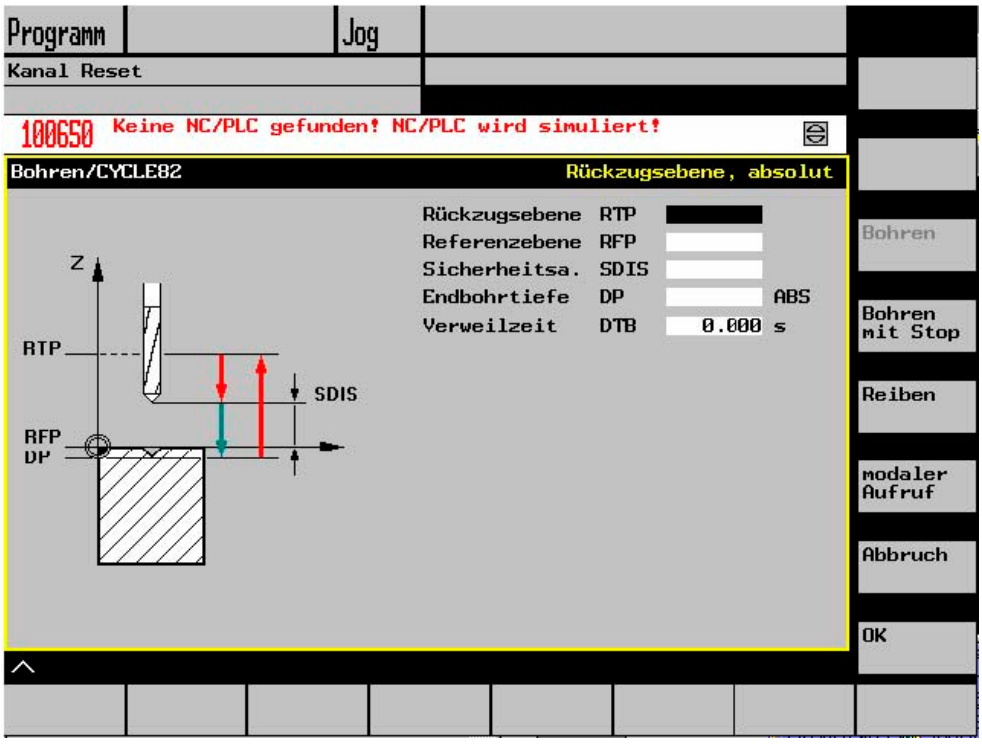

图 2-8 HMI 高级的显示画面

设置相同的显示画面

为了统一 HMI 高级和 HMI 内置 sl 的显示画面，可以在配置文件 WIZARD.INI 中将字体改为固定字体。

```
[FONT]
FixedFont=1
```

当前语言的文本为固定字体。 仅软键标签文本除外。 与 HMI 内置 sl 相反，HMI 高级的输入栏具有表示输入模式的象形图。 因此，该栏比 HMI 内置 sl 中的栏略小。

如上图所示，全部单元，如短文本、图形文本、输入栏和单位文本在相同的信息量情况下长度也大致相当。

然而如果必须定位某个栏，则可以在一定条件下通过待显示文本的长度得出不同 HMI 类型下不同的文本栏位置。

为了使两个系统使用同一个设计文件，则应设置对话框属性“属性”。
在对话框属性“系统”中查询运行环境。 从而，两个目标系统可以使用一个设计文件。

2.2.7 使用图/图形

使用图形

分为：

- 使用图/图形
- 帮助画面，例如：图示说明各个变量并突出显示在图形区中。
- 可以设计其它帮助画面替代可任意定位的短文本或者输入/输出栏。

最大尺寸	系统
560* 326 像素	HMI 高级/ HMI 内置 sl
688* 376 像素	PCU 50，带 OP012

HMI 内置 sl 的图形创建

创建图形，例如：用 MS Paint。

HMI 内置 sl 可以处理转换图形和 BMP 文件。 通常能够使用转换程序 BMP2BIN 生成的图形。 对于 BMP 文件，颜色在线转换。 为此，包含颜色表的文件(syscol.col, sysbw.col, ...), 必须添加段 [BMP]。 该文件总是分配一个 BMP 像素的颜色给 HMI 内置 sl 颜色。

存放位置

HMI 内置 sl 自动查找所连显示器的分辨率，并首先在所属的分辨率目录中查找所需文件。
如果在那里没有找到，就在下一个更小的分辨率目录中查找，直至目录 ico640 - 如果之前没有找到。

步骤

插入位图的步骤（用户图形）：

1. HMI 内置 sl 上的插入位图
2. 使用 Microsoft Paint 版本 4.0 或者更高版本创建 BMP 文件
3. 通过随附的 arj.exe（版本 2.41）将 BMP 文件压缩在文件夹 CUS.ARJ 中或者将每个单独的 BMP 文件压缩为一个独立的文件夹，使用 BMP 文件的文件名和扩展名为 ".BM_"

   例如：

   – 每个文件夹包含多个文件：

     arj a cus.arj my_file1.bmp my_file2.bmp my_file3.bmp my_file4.bmp

   – 每个文件夹包含一个文件：

     arj a my_file1.bm_ my_file1.bmp

限制

压缩是可选的。 然而，必须注意 CF 卡上的容量至少为 10 MB。

可以分配给位图任意一个软键，软键文本以 2 个反斜线符号开始。
反斜线符号后的文本则为包含位图的文件的名称。

例如在 ALUC.TXT 85000 0 0 "\\mybitmap.bmp"

注意

图形颜色在 HMI 内置 sl 上可能与画笔程序中的不同。

在 HMI 高级上创建图形

只要能够以此建立所述格式的一种，就可自由选择图像程序。 图像 / 图形和帮助画面可以有下列格式：

- 位图（BMP）
- Windows Meta 文件（WMF）
- 图标文件（ICO）

“图中图”

在背景（帮助画面）中可以显示其它的图形，但也取决于变量值。
例如对于软键，可以分配一个显示区给一个图形文件。

举例：

- 显示带有图形的短文本栏：

  DEF VAR1=(S///,"\\图 1.bmp" ////160,40,50,50) ;类型 S 无意义

- 显示带有图像的转换栏，此时可以通过一个 PLC 标记字节选择图像。

  DEF VAR1=(IDB/*1="\\图 1.bmp",2="\\图 2.bmp"//,$85000/wr1//"MB[0]"//160,40,50,50)

作为第四个参数在此位置给定显示图形的高度（输入/输出栏）。
也可以在该栏（短文本，输入/输出栏）中给定位图。

参见

图形的查找方案 (页 189)

颜色表 (页 194)

2.3 变量

2.3.1 变量属性

变量值

变量的重要属性即变量值。

可以通过如下方式分配值：

- 定义变量时预设
- 分配系统或者用户变量
- 采取方法

编程

句法：	名称**val** = 变量值	
	名称 = 变量值	
说明：	变量值 val (value)	
参数：	名称：	变量名称
	变量值	变量值
举例：	`VAR3 = VAR4 + SIN(VAR5)`	
	`VAR3.VAL = VAR4 + SIN(VAR5)`	

变量状态

通过变量状态的属性可以在运行时查询，变量是否包含有一个有效值。 该属性可通过值 FALSE = 0 读写。

编程

句法：	名称**vld**	
说明：	变量状态 vld (validation)	
参数：	名称：	变量名称
		查询结果可能是：
	FALSE =	无效值
	TRUE =	有效值

举例：

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

变量： 更改属性

在符号*属性的名称 = 值* 中重新分配变量值。
分析等号右边的表达式并分配变量或者变量属性。

举例：

名称**ac** = 存取级	(ac: access level)
名称**al** = 文本对齐	(al: alignment)
名称**bc** = 背景颜色	(bc: back color)
名称.**fc** = 前景颜色	(fc: front color)
名称**al** = 字体大小	(fs: font size)
名称**gt** = 图形文本	(gt: graphic text)
名称**hlp** = 帮助画面	(hlp: help)
名称**htx** = 帮助文本	(htx: help text)
名称**li** = 极限值	(li: limit)
名称**lt** = 长文本	(lt: long text)
名称**max** = 最大极限值	(max: maximum)
名称**min** = 最小极限值	(min: minimum)
名称**st** = 短文本	(st: short text)
名称**typ** = 变量类型	(typ: type)
名称**ut** = 单位文本	(ut: unit text)
名称**val** = 变量值	(val: value)
名称**var** = 系统或者用户变量	(var: variable)
名称**vld** = 变量状态	(vld: validation)
名称**wr** = 输入模式	(wr: write)

2.3.2 编程变量

编程

在以下概述中简要说明了变量参数。 详细的说明请参见后续章节。

参数	说明	
1 变量类型	必须规定变量类型。	
	R[x]:	REAL（小数点位数为正数 +）
	I:	INTEGER
	S[x]:	字符串（字符串长度为正数 +）
	C:	字符（单字符）
	B：	BOOL
	V:	VARIANT
2 极限值	最小极限值，最大极限值 预设置： 空 极限值通过逗号隔开。 类型 I、C 和 R 的极限值可以使用十进制的格式或者字符 "A"、"F"定义。	
转换栏	输入/输出栏中带有预设输入项的列表： 列表通过 * 开始，各输入项用逗号隔开。 可以赋值输入项。 极限值的输入项视为转换栏的列表。 如果只输入一个 *，则建立一个可变的转换栏。 预设置： 无	
表格名称	NCK/PLC 相同类型值的表格名称，它可以通过通道模块编译地址。 表格名称通过一个前置的极限值或者转换栏的 % 符号进行区分。 表格名称可以通过逗号分隔，还可以跟随一个文件名，该文件名指定定义表格描述的文件。	
3 预设值	如果没有定义任何预设值并且没有分配系统或者用户变量给变量，则分配转换栏的第一个单元。 如果没有定义转换栏，则不进行预设，即：变量处于状态“未计算”。 预设置： 未预设	
4 文本	顺序已预先规定。 也可以显示一个图形代替短文本。 预设置： 空	
	长文本：	显示行的文本
	短文本：	对话框单元的名称
	图形文本：	文本参考图形名称。
	单位文本：	对话框单元的单位
5 属性	属性影响下列特性： • 输入模式 • 存取等级 • 短文本的文本对齐 • 字体大小 • 极限值 • 打开对话框时的属性和 CHANGE 块有关 属性通过逗号隔开，顺序任意 属性不适用于转换栏。 每个组件都可以进行定义。	

参数	说明	
	输入模式	wr0: 输入/输出栏不可见，短文本可见， wr1: 读取（没有输入中心） wr2: 读取和写入（行以白色显示） wr3: wr1 带输入中心 wr4: 所有变量单元不可见，没有输入中心 wr5: 按下任何键立即保存输入的值（和 wr2 相反 - 该模式下，在退出栏或者按下返回键后才开始保存值）。 预设置： wr2
	存取等级	空： 总是可以写入 ac0...ac7: 保护等级 如果存取等级未达到，行显示为灰色，标准设置： ac7
	短文本的文本对齐	al0: 左对齐 al1: 右对齐 al2: 中间对齐 预设置： al0
	字体大小	fs1: 标准字体大小（8 Pt） fs2: 双倍字体大小 预设置： fs1 确定行间距。 标准字体大小为每对话框 16 行。 图形和单位文本只能为标准字体大小。
	极限值	通过极限值可以检查变量值是否在规定的最小极限值和最大极限值之内。 预设置： 取决于规定的极限值 li0: 没有检查 li1: 检查最小极限值 li2: 检查最大极限值 li3: 检查最大极限值和最小极限值
	打开时的属性	定义变量时规定的 cb 属性，优先于定义对话框时的总定义 多个属性通过逗号隔开。
	cb0: cb1:	在打开对话框时处理定义变量的 CHANGE 块（预设置）。 多个属性通过逗号隔开。 只有在变量值改变时，才处理定义变量的 CHANGE 块。
6 帮助画面	帮助画面文件：	pdf 文件名称。 预设置： 空
	帮助画面文件的名称在双引号中。 如果光标移至该变量，则自动显示画面（替代目前的图形）。	
7 系统或者用户变量	可向该变量分配 NC/PLC 上的系统或者用户变量。 系统或者用户变量用双引号括起。 参考文献：列表手册，“系统变量列表” /PGA1/	
8 短文本位置	短文本位置（和左边缘/上边缘的间距、宽度） 位置数据的单位为像素，并且以对话框主体的左上角为基准。 数据总是用逗号隔开。	
9 输入/输出栏位置	输入/输出栏位置（和左边缘/上边缘的间距、宽度、高度） 位置数据的单位为像素，并且以对话框主体的左上角为基准。 数据总是用逗号隔开。 如果改变此位置，则短文本、图形文本和单位文本的位置也同时改变。	

参数	说明	
10 颜色	前景颜色、背景颜色： 颜色通过逗号隔开。 颜色的设置仅适用于输入/输出栏；对于其他的文本，无需定义颜色。 值范围： 1...10 预设置： 前景颜色： 黑色，背景颜色： 白色 输入/输出栏的标准颜色取决于写入模式：	
	wr0:	前景和背景颜色： 窗口背景颜色
	wr1:	文本颜色： 黑色，窗口背景颜色
	wr2:	文本颜色： 黑色，背景颜色： 白色
	wr3:	同 wr0
	wr4:	同 wr1
	wr5:	同 wr2
11 帮助 **(仅 HMI 高级)**	帮助文件：	pdf 文件路径
	索引：	包含帮助文本的文件的索引
	帮助文本：	帮助文本文件中显示的帮助文本
	数据总是通过逗号隔开，顺序固定 用双引号括起帮助文件和帮助文本。 PDF 文件必须位于目录 CUS.DIR\hlp.dir 或者 CST.DIR\hlp.dir 下。 其它相应的 PDF 和文本文件的名称必须一致。 在文本文件中 PDF 文件名必须大写。 每个对话框单元都可以设计多个链式帮助参考（帮助链），即依次调用帮助并在最后的帮助后重新显示第一个帮助。 如果调用第二个或其他帮助时仍是同一个文件/索引/帮助文本，则表示缺少数据。 当光标位于相应栏中并且按下信息键时，帮助显示。	

2.3.3 变量类型 [1] 的详细说明

INTEGER 变量类型

" INTEGER"型变量可以使用下列扩展符号显示在输入/输出栏中并保存在存储器中。

- 扩展数据类型中的**第 2 个字符**

描述格式	
B	二进制
D	带有正负号的十进制
H	十六进制
没有数据	带有正负号的十进制

- 扩展数据类型中的**第 3 个和第 4 个字符**

存储器保存	
B	字节
W	字
D	双字
BU	不带正负号的字节
WU	不带正负号的字
DU	不带正负号的双字

INTEGER 型数据的字符顺序

1. “I” 作为 INTEGER 的基本标识符
2. 描述格式
3. 存储器保存
4. “U”，没有正负号

确定有效的 INTEGER 型：	
IB	整数变量 32 位二进制描述
IBD	整数变量 32 位二进制描述
IBW	整数变量 16 位二进制描述
IBB	整数变量 8 位二进制描述
I	整数变量 32 位 十进制描述，带正负号
IDD	整数变量 32 位 十进制描述，带正负号
IDW	整数变量 16 位 十进制描述，带正负号
IDB	整数变量 8 位 十进制描述，带正负号
IDDU	整数变量 32 位 十进制描述，不带正负号
IDWU	整数变量 16 位 十进制描述，不带正负号
IDBU	整数变量 8 位 十进制描述，不带正负号
IH	整数变量 32 位十六进制描述
IHDU	整数变量 32 位十六进制描述
IHWU	整数变量 16 位十六进制描述
IHBU	整数变量 8 位十六进制描述

VARIANT 型变量

变量类型 VARIANT 通过最后赋值的数据类型确定。 它可以通过功能 ISNUM 或者 ISSTR 查询。 VARIANT 类型主要适用于向 NC 码写入变量名或者数值。

编程

可以检查变量的数据类型：

句法： **ISNUM** *(VAR)*

参数：

VAR		待检查其数据类型的变量名称。
		查询结果可能是：
	FALSE =	没有数值（数据类型 = STRING）
	TRUE =	数值（数据类型 = REAL）

句法： **ISSTR** *(VAR)*

参数：

VAR		待检查其数据类型的变量名称。
		查询结果可能是：
	FALSE =	数值（数据类型 = REAL）
	TRUE =	没有数值（数据类型 = STRING）

举例：

```
IF ISNUM(VAR1) == TRUE
IF ISSTR(REG[4]+2) == TRUE
```

可以改变变量的显示模式：

- 对于 INTEGER 型变量可以改变显示方式。

B	二进制
D	带有正负号的十进制
H	十六进制

不带正负号

U 总是用于未标记

- 对于 REAL 型变量，只能更改小数点的位数。

  不允许更改基本类型，若更改，则导致文件 ERROR.COM 出现错误信息。

  举例：

```
Var1.typ = "IBW"
Var2.typ = "R3"
```

数值显示

计算可以以二进制、十进制、十六进制或者指数方式描述。
二进制、十六进制和指数描述方式的计算值都用单引号括起。

二进制	'B01110110'
十进制	123.45
十六进制	'HF1A9'
指数	'-1.23EX-3'

举例：

```
VAR1 = 'HF1A9'
REG[0]= 'B01110110'
DEF VAR7 = (R//'-1.23EX-3')
```

注意

在通过功能 GC
生成代码时计算值仅考虑十进制或者指数描述方式，而**不**使用二进制和十六进制描述方式。

2.3.4 转换栏 [2] 的详细说明

说明

通过转换栏扩展可以显示与 NC/PLC 变量有关的文本（在转换栏中的输入项）。
只能读取使用转换栏扩展的变量。

编程

句法：

DEF 名称 =(变量类型 /+ $文本号码 | * 值="\\图"[,值="\\图 2.bmp"][, ...]
/[预设值]
/[文本（长文本、短文本、图形文本、单位文本）]
/[属性]
/[帮助画面]
/[系统或者用户变量]
/[短文本位置]
/[输入/输出栏位置（左、上、宽度、高度）]
/[颜色]
/[帮助])

说明：

显示对话框时，在输入/输出栏中显示文本号码 $85015 的内容。
在系统变量 DB90.DBB5 输入预设值 15。 如果系统变量 DB90.DBB5 中的值改变，在每次改变时显示的文本号码重新生成 $（85000 + <DB90.DBB5>）。

参数：

变量类型	系统或者用户变量中指定的变量类型
文本号码	与语言相关的文本号码（基本），该号码作为基本号码使用
系统或者用户变量	通过最终的文本号码（基本 + 补偿）形成的系统或者用户变量（补偿）。

举例：

```
DEF VAR1=(IB/+ $85000/15///"DB90.DBB5")
```

可变转换栏

可以将可变转换栏分配给对话框单元，即：按下转换键则可以向变量分配 CHANGE 方法中定义的值。

为了标记一个可变的转换栏，在定义变量时在特性极限值或者转换栏中输入一个单独的星号 * 。

举例： `DEF VAR1=(S/*)`

图形与转换栏有关

转换栏用替换的图像覆盖。 如果标记字节值为 1 ，则显示 “图 1.bmp”；如果标记字节值为 2 ，则显示 “图 2.bmp”

```
DEF VAR1=(IDB/*1="\\图 1.bmp",
                  2="\\图 2.bmp"//,$85000/wr1//"MB[0]"//160,40,50,50)
```

图形的大小和位置在“输入/输出栏位置（左侧、上部、宽度、高度）”中规定。

2.3.5 预设值 [3] 的详细说明

概述

根据变量栏（输入/输出栏或者转换栏）是否分配了一个预设值，一个系统或者用户变量或者两者都分配，得到不同的变量状态。 只有当变量分配了一个有效值时，转换才可行。

预设值生效

当...			则...
栏类型	预设值	系统或者用户变量	栏类型反应
输入/输出栏	是	是	在系统或者用户变量中写预设值
	否	是	使用系统或者用户变量作为预设值
	错误	是	未计算，系统或者用户变量未描述/未使用
	是	否	预设值
	否	否	未计算
	错误	否	未计算
	是	错误	未计算
	否	错误	未计算
	错误	错误	未计算
Toggle	是	是	在系统或者用户变量中写预设值
	否	是	使用系统或者用户变量作为预设值
	错误	是	未计算， 系统或者用户变量未描述/未使用
	是	否	预设值
	否	否	预设值 =转换栏的第一个单元
	错误	否	未计算
	是	错误	未计算
	否	错误	未计算
	错误	错误	未计算

2.3.6 短文本 [8]、输入/输出栏 [9] 位置的详细说明

概述

短文本和图形文本以及输入/输出栏和单位文本总是形成一个单元。即短文本的位置数据也对图形文本和输入/输出栏数据以及单位文本上的数据有效。

编程

设计的位置数据覆盖标准值，即也仅能改变一个单独的值。
如果下列对话框单元没有设计位置数据，则采用上一对话框单元的数据。

如果对话框单元没有规定位置，则使用预设置。
短文本和输入/输出栏的栏宽度在标准情况下各行由栏数和最大栏宽度确定，即：栏宽度=最大行宽/列数。

图像和单位文本宽度是固定的，根据编程支持的请求优化。
如果已设计图像和单位文本宽度，则短文本或者输入/输出栏的宽度相应缩短。

短文本和输入/输出栏的顺序可以通过位置数据互换。

2.3.7 帮助 [11] 的详细说明（仅适用于 HMI 高级）

说明

可以在运行时扩展或者删除对话框单元的帮助环链。
通过多次调用功能可以任意扩展帮助环链。

编程

句法：	**ADDHTX**（名称、帮助文件、索引、帮助文本）	
说明：	扩展帮助环链	
参数：	名称	待扩展的帮助环链的变量名。
	帮助文件：	文件的路径数据（PDF 格式）
	索引：	包含帮助文本的文件的索引
	帮助文本：	帮助文本文件中显示的帮助文本
举例：	ADDHTX(VAR1,"C:\OEM\HLP\MYHLP.PDF",15,"机床数据")	

句法：	**CLRHTX**（名称）	
说明：	删除帮助环链	
参数：	名称	待删除的帮助环链的变量名称。

帮助文件： 文件的路径数据（PDF 格式）
索引： 包含帮助文本的文件的索引
帮助文本： 帮助文本文件中显示的帮助文本

举例：

```
CLRHTX(VAR1)
```

2.3.8 Anwendungsbeispiele（应用举例）

辅助变量

辅助变量是内部计算变量。
计算变量如同变量一样定义，但是另外除了变量值和状态之外没有属性，即辅助变量在对话框中不可见。 辅助变量是 VARIANT 类型。

编程

句法： DEF *名称*
说明： VARIANT 类型的内部计算变量
参数： 名称： 辅助变量名称

举例： DEF OTTO；定义一个辅助变量

句法： 名称 val = *辅助变量值*
名称 = *辅助变量值*
说明： 辅助变量值在方式中指定。
参数： 名称： 辅助变量名称
辅助变量值： 辅助变量内容

举例：

```
LOAD
  OTTO = "Test"                ;  分配值 "Test" 给辅助变量 Otto
END_LOAD
LOAD
  OTTO = REG[9].VAL            ;  分配寄存器的值给辅助变量
END_LOAD
```

通过变量计算

在每次退出输入/输出栏（通过 ENTER 或者转换键）后计算变量。 计算在 CHANGE 方式中设计并在每次更改值时运算。

通过变量状态可以查询，变量是否包含有效值，例如：

```
Var1 = Var5 + SIN(Var2)
Otto = PI * Var4
```

系统变量间接编译地址

系统变量也可以间接编译地址，即和另一个变量相关。

```
PRESS(HS1)
  轴= 轴 +1
  WEG.VAR="$AA_DTBW["<<ACHSE<<"]"    ;编程变量轴地址
END_PRESS
```

更改软键标签

举例：

```
HS3.st = “新文本”    ;改变软键标签
```

2.3.9 举例 1： 分配变量类型、文本、帮助、颜色

举例 1

分配变量类型、文本、帮助、颜色的属性

DEF Var1 = (R///,"实际值",,"mm"//"Var1.bmp"////8,2)

变量类型：	REAL
极限值或者转换栏输入项：	无
预设值：	无
文本	
长文本：	无
短文本：	实际值
图形文本：	无
单位文本：	mm
属性：	无
帮助画面：	Var1.bmp
系统或者用户变量：	无
短文本位置：	没有规定，则为标准设置
输入/输出栏位置：	没有规定，则为标准设置
颜色：	
前景颜色：	8
背景颜色：	2
帮助：	无

2.3.10 举例 2：定义变量类型，极限值，属性，短文本位置

举例 2

定义变量类型、极限值，属性，短文本位置的属性

DEF Var2 = (I/0,10///wr1,al1/// , ,300)

变量类型：	INTEGER
极限值或者转换栏输入项：	MIN： 0
	MAX: 10
预设值：	无
文本	无
属性：	
输入模式	只读
短文本的文本对齐	右对齐
帮助画面：	无
系统或者用户变量：	无
短文本位置：	
到左边缘的间距	无
到上边缘的间距	无，则为到左边缘、上边缘的标准间距
宽度：	300
输入/输出栏位置：	没有规定，则为标准设置
颜色：	没有规定，则为预设置
帮助：	无

2.3.11 举例 3： 定义变量类型、预设、系统或者用户变量、输入/输出栏位置

举例 3

定义变量类型属性、预设、系统或者用户变量、输入/输出栏位置

```
DEF Var3 =(S//10////"$R[1]"//300,10,200//"Help.pdf",1,"帮助
1")
```

变量类型：	字符串
极限值或者转换栏输入项：	无
预设值：	10
文本	无
属性：	无
帮助画面：	无
系统或者用户变量：	$R[1]（R 参数 1）
短文本位置：	相对于输入/输出栏的标准位置
输入/输出栏位置：	
到左边缘的间距	300
到上边缘的间距	10
宽度：	200
颜色：	没有规定，则为预设置
帮助：	如果按下<i> 键，则用帮助文本“帮助 1”从文件 Help.pdf 中调用包含索引 1 的文件页。

2.3.12 转换栏、帮助调用和画面显示的举例

举例 4

在转换栏中不同的输入项：

极限值或者转换栏输入项：

```
DEF Var1 = (I/* 0,1,2,3)
DEF Var2 = (S/* "Ein", "Aus")
DEF Var3 = (B/* 1="Ein",          ；1 和 0
0="Aus")                          是数值，显示“接通（EIN）”和“关闭（AUS）”
DEF Var4 = (R/* ARR1)             ；ARR1 是数组名称。
```

举例 5 (仅 HMI 高级)

每个对话框包含多个帮助调用：

DEF Var5 = (R/////////"Help1.pdf",1,"帮助 1",,2,"帮助 2","Help3.pdf",3,)

变量类型：	REAL	
极限值或者转换栏输入项：	无	
预设值：	无	
文本	无	
属性：	无	
帮助画面：	无	
系统或者用户变量：	无	
短文本位置：	无	
输入/输出栏位置：	无	
颜色：	没有规定，则为预设置	
帮助：	1. 帮助项	
	帮助文件：	HELP1.PDF
	索引：	1
	帮助文本：	帮助 1
	2. 帮助项	
	帮助文件：	HELP2.PDF
	索引：	2
	帮助文本：	帮助 2
	3. 帮助项	
	帮助文件：	HELP3.PDF
	索引：	3
	帮助文本：	帮助 3

举例 6

显示画面，而不是短文本：
图的大小和位置在“输入/输出栏位置（左侧、上部、宽度、高度）”中规定。

DEF VAR6= (V///,"\\图 1.bmp" ////160,40,50,50)

变量类型：	VARIANT
极限值或者转换栏输入项：	无
预设值：	无
文本	无
属性：	无
帮助画面：	无
系统或者用户变量：	无
短文本位置：	图 1.bmp
输入/输出栏的位置	
左侧距离：	160
上部距离：	40
宽度：	50
高度：	50
颜色：	没有规定，则为预设置
帮助：	无

2.3.13 使用字符串

字符串链

设计时也可以使用字符串，以动态配置文本显示或者合并代码生成的不同文本。

规则

使用字符串变量时注意以下规定：

- 链接由左向右处理。
- 层叠的表达式由内向外运算。
- 忽略大小写。

字符串可以通过一个简单的空字符串指令删除。

字符串在等号右边以运算符 "<<"开始。 字符串中双引号 (") 通过两个连续的双引号标记。
字符串可以在 IF 指令中检查相等性。

举例

下列举例预设：

```
VAR1.VAL =“这是一个”
VAR8.VAL = 4
VAR14.VAL = 15
VAR2.VAL = “错误”
$85001 = “这是一个”
$85002 = “报警文本”
```

编辑字符串：

- 合并字符串：

  ```
  VAR12.VAL = VAR1 << “错误”    ;结果： “这是一个错误”
  ```

- 删除一个变量：

  ```
  VAR10.VAL = ""    ;结果： 空字符串
  ```

- 设定一个带有文本变量的变量：

  ```
  VAR11.VAL = VAR1.VAL    ;结果： “这是一个”
  ```

- 数据类型匹配：

  ```
  VAR13.VAL ="这是 " << (VAR14 - VAR8) << ". 错误"
                ;结果： “这是第 11 个错误”
  ```

- 处理数字值：

```
VAR13.VAL = “错误” << VAR14.VAL << ": " << $T80001 << $T80002
            ;结果： “错误 15： “这是一个报警文本”
IF VAR15 == “错误”  ; IF 指令中的字符串
  VAR16 = 18.1234
            ;结果： VAR16 等于 18.1234，
 ；当 VAR15 等于“错误”时
ENDIF
```

- 字符串中的双引号：

```
VAR2="Hallo dies ist ein "" Test"
            ;结果： 你好，这是一个“测试
```

- 和变量内容有关的系统或者用户变量的字符串：

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

2.3.14 变量 CURPOS

说明

通过变量 CURPOS 可以在当前对话框的激活的输入栏中调出或者操纵光标位置。
变量显示光标前有多少个字符。 如果光标位于输入栏开始处，则 CURPOS 接受值为 0。
如果更改 CURPOS 值，则光标停留在输入栏中相应的位置上。

为了可以在变量值更改情况下反应，可以借助于一个 CHANGE（改变）块监控改变情况。
如果 CURPOS 值改变，则跳转 CHANGE（改变）块并执行包含的指令。

2.3.15 变量 CURVER

说明

属性 CURVER（当前版本）允许匹配编程用于处理不同的版本。 变量 CURVER 仅可读。

注意

在代码生成时自动以最新的版本生成，即使之前已用老的版本反编译。 命令 "GC" 总是生成最新的版本。 在生成代码中，在使用注释当版本 > 0 时插入一个生成的代码的附加标记。

规则

总是显示带有所有变量的最新的对话框。

- 以前的变量不允许改变。
- 新的变量以任意顺序插入在以前（循环）的编程中。
- 不允许从对话框由一个版本到下一个版本去除变量。
- 对话框必须包含所有版本的变量。

举例

```
(IF CURVER==1 ...)                    ; CURVER 在反编译时自动通过反编译代码的版本设置。
```

2.3.16 变量 ENTRY

说明

通过变量 ENTRY 可以检查如何调用对话框。

编程

句法： **ENTRY**
说明： 变量 ENTRY 仅可读。
返回值： 查询结果可能是：

0 = 没有编程支持
1 = 编程支持（由编程支持调用对话框。）
2 = 编程支持 + 上一对话框的预设置（子对话框）
3 = 编程支持 + 反编译
4 = 编程支持 + 反编译，带有生成的注释，带有 # 符号
5 = 编程支持 + 反编译，带有生成的注释，不带 # 符号

举例

```
IF ENTRY == 0
 DLGL(“在编程下不调用对话框”)
ELSE
 DLGL(“在编程下调用对话框”)
ENDIF
```

2.3.17 变量 ERR

说明

通过变量 ERR 可以检查是否已正确执行先前行。

编程

句法： **ERR**

说明： 变量 ERR 仅可读。

返回值：

	查询结果可能是：
FALSE =	已正确执行先前行
TRUE =	未正确执行先前行

举例

```
VAR4 = 螺纹[VAR1,"KDM",3]          ;  作为数组给出值
IF ERR == TRUE                     ;  查询是否在数组中找到值
VAR5 = “数组存取出错”
                                   ;  如果未在数组中找到值，则分配值“数组存取出错”给变量。
ELSE
VAR5 = “一切正常”                  ;  如果在数组中找到值，则分配值“一切正常”给变量。
ENDIF
```

2.3.18 变量 FILE_ERR

说明

用变量 FILE_ERR 可以检查前面的 GC 或者 CP 指令是否正确执行。

编程

句法： **FILE_ERR**

说明： 变量 FILE_ERR 仅可读。

返回值： 可能的结果是：

0 = 排列中的操作
1 = 驱动器/路径不存在
2 = 路径/文件存取故障
3 = 驱动器未就绪
4 = 错误的文件名
5 = 文件已经打开
6 = 存取失败
7 = 目标路径不存在或不允许
8 = 复制源符合目标
10 = 内部错误： FILE_ERR = 10
时，与一个未分类到其他类别故障有关。

举例

```
CP("D:\source.mpf","E:\target.mpf")
                                    ;  从 source.mpf 向 E:\target.mpf 复制
IF FILE_ERR > 0                     ;  询问是否出现故障
IF FILE_ERR == 1                    ;  查询特定的错误号并输出所属的故障文本
VAR5 = “驱动器/路径不存在”
ELSE
IF FILE_ERR == 2
VAR5 = “路径 - /文件存取故障”
ELSE
IF FILE_ERR == 3
VAR5 = “错误文件名”
ENDIF
ENDIF
ENDIF
ELSE
VAR5 = “一切正常”                   ;  如果 CP (或者 GC)中没有出现错误，则输出“一切正常”
ENDIF
```

2.3.19 变量 FOC

说明

使用变量 FOC 可以控制对话框内的输入中心（当前有效的输入/输出栏）。
已预先固定定义光标左右键、前后键以及 PGUP、PGDN 的反应。

注意

FOC 不允许通过一个导航事件触发。 光标位置只允许在软键 PRESS 块，CHANGE 块，...中改变。

带有输入模式 wr = 0 和 wr = 4 的变量以及辅助变量无法实现聚焦。

编程

句法：	FOC	
说明：	可以读取和写入该变量。	
返回值：	读取	聚焦变量的名称作为结果输出。
	写入	可以写入一个字符串或者数值。 字符串视为变量名称，数值视为变量索引。

举例

```
IF FOC == "Var1"                ;  读取输入中心
  REG[1] = Var1
ELSE
  REG[1] = Var2
ENDIF

FOC = "Var1"                    ;  分配变量 1 给输入中心。
FOC = 3                         ;  分配第 3 对话框单元， WR ≥ 2 给输入中心。
```

2.3.20 变量 S_CHAN

说明

通过变量 S_CHAN 可以确定用于显示或者某个评估的当前通道号码。

2.4 综合对话框单元

2.4.1 数组

定义

通过数组可以排列或者保存同一数据类型的数据，从而可以通过索引存取数据。

说明

数组可以是一维或者二维。 一个一维数组可视为带有一行或者一列的一个二维数组。
数组通过标记 //A 定义并通过 //END 结束。 行和列数目任意。 一个数组有下列结构：

编程

句法：	***//A**(名称)*	
	(a/b...)	
	(c/d...)	
	...	
	//END	
说明：	定义数组	
参数：	名称	数组名称
	a, b, c, d	数组值
		STRING 类型的值必须用双引号括起。

举例

```
//A(螺纹)                              ;   高度/螺距/底直径
(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
```

2.4.2 存取数组单元的值

说明

通过属性值（名称.val）可以继续传送一个数组存取值。

行索引（数组的行编号）和列索引（数组的列编号）各自从 0 开始。如果显示数组外的行索引或者列索引，则输出值 0 或者空字符串并且变量 ERR 的值为 TRUE。 当未找到查找关键字时，变量 ERR 同样为 TRUE。

编程

句法： 名称 [Z,[M[,C]]].val 或者
名称 [Z,[M[,C]]]
说明： 存取仅带有一列的一维数组：
句法： 名称 [S,[M[,C]]].val] 或者
名称 [S,[M[,C]]] 或者
说明： 存取仅带有一行的单维数组
句法： 名称 [Z,S,[M[,C]]].val 或者
名称 [Z,S,[M[,C]]]
说明： 存取二维数组
参数： 名称： 数组名称
Z: 行值（行索引或者查找关键字）
S: 列值（列索引或者查找关键字）
M: 存取模式
0 直接
1 按行查找，列直接
2 按列查找，行直接
3 查找
4 查找行索引
5 查找列索引
C: 比较模式
0 查找关键字必须位于行或者列的值范围内。
1 查找关键字必须准确找到

举例：

```
VAR1 = MET_G[REG[3],1,0].VAL  ;分配 Var1 数组中的一个值
                              MET_G
```

存取模式

- **存取模式“直接”**

  在存取模式“直接”（M = 0）情况下，数组上的存取通过行索引（以 Z 表示）和列索引（以 S 表示）实现。不评估比较模式 C。

- **存取模式“查找”**

在存取模式 M = 1、2 或者 3 情况下，查找总是在行 0 或者列 0 中实现。

模式 M	行值 Z	列值 S	输出值
0	行索引	栏索引	行 Z 和列 S 中的值
1	查找关键字： 在列 0 中查找	列索引，从该列中读取值	查找的行和列 S 中的值
2	行索引，从该行中读取返回值	查找关键字： 在行 0 中查找	行 Z 和查找的列中的值
3	查找关键字： 在列 0 中查找	查找关键字： 在行 0 中查找	查找的行和查找的列中的值
4	查找关键字： 在列 S 中查找	行索引，在该行中查找	行索引
5	行索引，在该行中查找	查找关键字： 在行 Z 中查找	栏索引

比较模式

在使用比较模式 C = 0 时，查找行或者查找列的内容以升序分类。
如果查找关键字小于第一个元素或者大于最后一个元素，则给出值 0
或者一个空字符并且错误变量 ERR 为真。

在使用比较模式 C = 1 时，查找关键字必须可在查找行或者查找列中找到。
如果没有找到查找关键字，则给出值 0 或者一个空字符并且错误变量 ERR 为真。

2.4.3 举例： 存取数组单元

前提条件

下列定义两个数组是下面例子的前提条件。

```
//A(螺纹
                (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)
                ("BEZ" /     "STG" /     "KDM" )
                (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
```

举例

- **存取模式 1 举例：**

  在 Z 中有查找关键字。 该关键字总是在列 0 中查找。 通过查找的关键字的行索引给出列 S 中的值：

  ```
  VAR1 = 螺纹[0.5,1,1] ；VAR1 值为 0.125
  ```

  说明:

  数组“螺纹”的列 0 中查找值 0.5 并给出列 1 中查找的相同行的值。

- **存取模式 2 举例：**

  在 S 中有查找关键字。 该关键字总是在行 0 中查找。 通过查找的关键字的列索引给出行 Z 中的值：

  ```
  VVAR1 = ARRAY2[3,"STG",2] ；VAR1 值为 0.125
  ```

  说明:

  在数组“Array2”的行 0 中查找带有内容“STG”的列。 得到查找的列中的值和带有索引 3 的行。

- **存取模式 3 举例：**

  在 Z 和 S 中总是有一个查找关键字。 通过 Z 中的关键字在列 0 中找到行索引和通过 S 中的关键字在行 0 中找到列索引。 通过查找的行索引和列索引输出数组中的值：

  ```
  VAR1 = ARRAY2[0.6,"STG",3]    ;VAR1 值为 0.15
  ```

  说明:

  在数组“Array2”的列 0 中查找带有内容 0.6 的行，在数组“Array2”的行 0 中查找带有内容“STG”的列。 根据 VAR1 给出查找的行和列中的值。

- **存取模式 4 举例：**

  在 Z 中有查找关键字。 在 S 中有要查找的列索引。 给出查找的关键字的行索引：

  VAR1 = 螺纹[0.125,1,4] VAR1 值为 2

  说明:

  在数组“ 螺纹的列 1 中查找值 0.125 并根据 VAR1 给出查找的值的行索引。

- **存取模式 5 举例：**

  在 Z 中有要查找行的行索引。 查找关键字在 S 中。给出查找的关键字的列索引：

  VAR1 = 螺纹[4,0.2,5,1] VAR1 值为 1

  说明:

  在数组“ 螺纹”的行 4 中查找值 0.2 并根据 VAR1 给出查找的值的列索引。 已选择比较模式 1，因为行 4 的值不是按升序分类的。

2.4.4 查询数组单元的状态

说明

通过属性状态可以查询数组存取是否提供一个有效值。

编程

句法：	*名称 [Z, S, [M[,C]]]*.vld	
说明：	该属性仅可读。	
参数：	名称	数组名称
返回值：	FALSE =	无效值
	TRUE =	有效值

举例

```
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
```

2.4.5 表格栅格 (Grid)

定义

和数组相反，表格栅格 (Grid) 的值持续更新。
它与系统变量值的表格式描述有关，系统变量可以通过一个通道中的一个模块编译地址。

分配

通过表格名称将变量定义分配给表格单元定义：

- 变量定义确定显示的值，表格单元定义确定屏幕的外观和布置。
  表格栅格从变量定义行接收输入 / 输出栏的属性。
- 栅格的可见区域通过输入 / 输出栏宽度和高度确定。
  如果比可见区域位置存在更多的行和列，则可以通过水平和垂直滚动条进行查看。

说明

表格描述参考内容收录在变量描述中：

DEF *Bezeichner* =	**Bezeichner = 变量名称**		
	变量类型	→	1
	/[极限值或者转换栏或者表格名称]	→	2
	/[预设值]	→	3
	/[文本（长文本，短文本|图，图形文本，单位文本）]	→	4
	/[属性]	→	5
	/[帮助画面]	→	6
	/[系统或者用户变量]	→	7
	/[短文本位置]	→	8
	/[输入/输出栏位置（左、上、宽度、高度）]	→	9
	/[颜色]	→	10
	/[帮助] （仅 HMI 高级）	→	11

表格名称 [2]

NCK/PLC 相同类型值的表格名称，它可以通过通道模块编译地址。
表格名称通过一个前置的极限值或者转换栏的 % 符号进行区分。
表格命名符可以通过逗号分隔，还可以跟随一个文件名，该文件名指定定义表格描述的文件。

系统或者用户变量 [7]

参数留空用于表格栅格，因为详细信息中要显示的变量规定在列定义行中。
表格描述可以动态提供。

2.4.6 定义表格栅格

说明

表格块由以下部分组成：

- 标题描述
- 1 至 n 列描述

编程

句法：	*IIG (表格名称/表格类型/行数 / [固定行属性] , [固定列属性])*	
说明：	定义表格栅格	
参数：	表格名称	这里表格名称不使用前缀 % 符号。 表格名称只能在一个对话框中使用一次。
	表格类型	0（预设置） PLC 或者用户数据（NCK 和通道专用的数据）表 1 备用
	行数	行数包括标题行 固定行或者固定列无法滚动显示。 列数由设计的列的数目给定。
	固定行属性	1: 激活 0: 未激活
	固定列属性	1: 激活 0: 未激活

2.4.7 定义列

说明

表格栅格中可以通过索引使用变量。 索引号码对于带有一个或者多个索引的 PLC 或者 NC 变量比较重要。

表格栅格中显示的值可以由最终用户在由属性确定的权限范围和可能定义的极限值范围中直接编辑。

编程

句法： *（类型/极限值/空/长文本,列标题/属性/帮助画面/系统或用户变量/列宽/偏移 1、偏移 2、偏移 3）*

说明： 定义列

参数： 和变量类似

类型	数据类型
极限值	最小极限值，最大极限值
长文本，列标题	
属性	
帮助画面	
系统或者用户变量	作为变量在双引号内给出 PLC 或者 NC 变量。
列宽	参数，单位像素
偏移	在分配的偏移参数内规定步宽（在该步宽中各个索引应指数运算），以填写该列。 • 偏移 1： 第 1 个索引的步宽 • 偏移 2： 第 2 个索引的步宽 • 偏移 3： 第 3 个索引的步宽

STRING 类型变量

如果变量类型是 STRING，必须规定该类型的变量长度，例如： `DEF CHAN STRING [16] TEXT[41]`

变量 CHAN 的列定义开始，例如 ：(S16/...)

文本文件的列标题

列标题可以规定为文本或者文本号码（$8xxxx），并且同样无法滚动显示。

改变列属性

可动态改变的（可写）的列属性称为：

- 极限值（最大、最小）、
- 列标题(st)、
- 属性(wr, ac 和 li)
- 帮助画面(hlp)和
- BTSS 变量(var)。

通过定义行中的变量命名符和列索引（以 1 开始）改变列属性。

举例: `VAR1[1].st="列 1"`

无法在 LOAD 块中读取列属性。

对于列定义，可以规定属性 wr、ac 和 li。

参见

对话框单元 (页 21)
可用的系统变量列表 (页 196)

2.4.8 表格栅格中的聚焦控制

说明

通过列和行属性可以在表格中设置和确定聚焦：

- 名称**Row**
- 名称**Col**

编程

表格的每行都具有属性 Val 和 Vld。

对于行属性写入和读出，除了定义行中的变量命名符之外，还规定一个行索引和列索引。

句法：	名称[行索引，列索引].Vld 或者 名称[行索引，列索引]
说明：	Val 属性
句法：	名称[行索引，列索引].Vld
说明：	Vld 属性

举例

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

如果没有规定行索引和列索引，则适用于聚焦行的索引，即：

```
Var1.Row =2
Var1.Col=3
Var1.val=1.203
```

2.4.9 举例： 定义列

概述

下面三个举例指出了标准表格（表格类型= 0）中单格和 PLC 变量之间的分配。

举例 1：

在第一行中显示列标题：

```
//G(MB_TAB/0/4/,1)
(I///,"MB 1 至 MB 3"///"MB1"/100/1)
(I///,"MB 4 至 MB 6"///"MB4"/100/1)
```

结果：

MB 1 至 MB 3	MB 4 至 MB 6
值(MB1)	值(MB4)
值(MB2)	值(MB5)
值(MB3)	值(MB6)

举例 2：

如果列定义的偏移 > 1，则产生以下行和列的分配：

```
//G(MB_TAB/0/4/,1)
(I///," MB1, MB3, MB5"///"MB1"/100/2)
(I///," MB2, MB4, MB6"/// "MB2"/100/2)
```

结果：每行中都已提高了变量索引偏移（=2）。

MB1、MB3、 MB5	MB2、MB4、MB6
值(MB1)	值(MB2)
值(MB3)	值(MB4)
值(MB5)	值(MB6)

举例 3：

列定义中的偏移和索引号码：

- 在第一列中每行变量的第一个索引提高 1。 偏移 1 = 1
- 在第二列中每行变量的第二个索引提高 1。 偏移 2 = 1

```
//G(MB_TAB/0/4/,1)
(IB///,"M1.1, M2.1, M3.1"/// "M1.1"/100/1)
(IB///,"M1.1, M1.2, M1.3"/// "M1.1"/100/,1)
```

结果：

M1.1, M2.1, M3.1	M1.1、M1.2、M1.3
值 (M1.1)	值 (M1.1)
值 (M2.1)	值 (M1.2)
值 (M3.1)	值 (M1.3)

其它可能：

- 第一列可以用递升的数字填写：
  举例: (I///,“行”///"0"/60/1)
- 第一列可以用语言文件中的连续文本填写：
  举例: (S///,“行”///"$80000"/60/1)

2.4.10 举例： 载入不同的表格栅格

说明

在本举例中变量"VAR1"首先指定表格"dummygrid" 。 LOAD 块中，根据 R 参数 R[0] 的内容，选择载入表格 "grid1" 或者 "grid2" 。
在同一文件中和定义变量"VAR1"一样定义表格：

```
//M(MASKE1/"GRID")
DEF VAR1=(R/% dummygrid///////200,75,300,85)
HS1=("")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("EXIT",ac7,se1)
VS8=("")

LOAD
  REG[0] = RNP ("$R[0]")
    IF (REG[0] == 0)
LG ("grid1", "var1")
ELSE
LG ("grid2", "var1")
ENDIF
END_LOAD

PRESS(VS7)
EXIT
END_PRESS

//END

//G(grid1/0/5/1,1)                                    ; （名称/类型/行...）
(R///"长文本 1","R1 至 R4"/wr2//"$R[1]"/80/1)        ; 第 1 列，标题"R1 至 R4"，自 R1 起带偏移 1
```

```
(R///"长文本 2","R5 至 R8"/wr2//"$R[5]"/80/1)      ; 第 2 列自 R5 起
(R///"长文本 3","R9 至 R15"/wr2//"$R[9]"/80/2)     ; 第 3 列自 R9 带偏移 2, R9 11 13 15
//END

//G(grid2/0/5/1,1)                                 ; （名称/类型/行...）
(R///"长文本 1","R1 至 R4"/wr2//"$R[1]"/60/1)      ; 第 1 列，标题"R1 至 R4"，自 R1 起带偏移 1
(R///"长文本 2","R5 至 R8"/wr2//"$R[5]"/60/1)      ; 第 2 列自 R5 起
(R///"长文本 3","R9 至 R15"/wr2//"$R[9]"/60/2)     ; 第 3 列自 R9 带偏移 2, R9 11 13 15
(R///"长文本 4","R9 至 R15"/wr2//"$R[9]"/60/2)     ; 第 3 列自 R9 带偏移 2, R9 11 13 15
//END

//G(dummygrid/0/5/1,1)                             ; （名称/类型/行...）
(R///"长文本 1","R1 至 R4"/wr2//"$R[1]"/80/1)      ; 第 1 列，标题"R1 至 R4"，自 R1 起带偏移 1
(R///"长文本 2","R5 至 R8"/wr2//"$R[5]"/80/1)      ; 第 2 列自 R5 起
//END
```

2.5 软键栏

2.5.1 软键描述

概述

确定软键名称。 不必占用所有软键。

HSx x 1 - 8, , 水平软键 1 至 8

VSy y 1 - 8, 垂直软键 1 至 8

所有水平软键和所有垂直软键分别布置在一起称作软键栏。
另外针对已有的软键栏，还可以定义其它软键栏，可以部分或者完全覆盖已有的软键栏。

原则上软键栏描述（描述块）如下构建：

说明块	注释	参考章节
//S...	; 软键栏开始标记	
HSx=...	; 定义软键	
PRESS(HSx) LM... END_PRESS	; 方法的开始标识 ; 动作 ; 方法的结束标识	参见章节“方法”
//END	; 软键栏结束标记	

2.5.2 定义软键栏

说明

定义软键栏也同时分配软键属性。

编程

句法：	***//S(名称)***	;软键栏开始标记
	...	
	//END	;软键栏结束标记
说明：	定义软键栏	
参数：	名称	软键栏名称

句法：	**SK** = *(文本[, 存取等级][, 状态])*	
说明：	定义软键	
参数：	SK	软键，例如： HS1 到 HS8, VS1 到 VS8
	文本	给定文本
	图文件名称	"\\my_pic.bmp" 或者通过单独的文本文件 $85199，例如：和语言相关的文本文件的下列文本： 85100 0 0 "\\c:\pic\my_pic.bmp". 显示在软键上的图形大小： 最大 80 x 34 像素
	存取等级	ac0 到 ac7 (ac7: 预设置)
	状态	se1: 可见（预设置）
		se2: 不可操作（灰色标签）
		se3: 高亮显示（最后操作的软键）

注意

对于软键标签，用 %n 进行换行。

在 HMI 高级上，软键最多 2 行，每行最多 10 个字符；在 HMI 内置 sl 上，最多 2 行，每行最多 9 个字符。

分配保护等级

操作员只能访问符合其保护等级和各个低于其保护等级的信息。

不同的保护等级有下列不同的含义： ac0 是最高的保护等级，ac7 是最低的保护等级。

保护等级	禁用密码	区域
ac0	密码	西门子
ac1	密码	机床制造商
ac2	密码	维修
ac3	密码	用户
ac4	钥匙开关位置 3	编程员，调试员
ac5	钥匙开关位置 2	合格的操作员
ac6	钥匙开关位置 1	受过培训的操作员
ac7	钥匙开关位置 0	学过相关内容的操作员

举例

```
//S(软键栏 1)                    ;  软键栏开始标记
HS1=("新建",ac6,se2)             ;  定义软键 HS1，分配标签 "OK"，保护等级 6
                                    和状态“不可操作”。
HS3=("\\图 1.bmp")               ;  分配图形给软键
HS5=("Exit")
VS2=("子屏幕窗口")
VS3=($85011, ac7, se2)
VS7=("取消", ac1, se3)           ;  定义软键 VS7，分配标签“取消”，保护等级 1 和状态“高亮”。
VS8=("OK", ac6, se1)                定义软键 VS8，分配标签 "OK"，保护等级 6 和状态“可见”。

PRESS(HS1)                       ;  方法开始标记
  HS1.st=“计算”                  ;  分配标签文本给软键
...
END_PRESS                        ;  方法结束标记

PRESS(RECALL)                    ;  方法开始标记
  LM(“屏幕窗口 21”)              ;  载入对话框
END_PRESS                        ;  方法结束标记
//END                            ;  软键栏结束标记
```

按键<RECALL>（回调）

另外还有回调 <RECALL> 按键可用于操作。

该按键与软键相反，不必定义。 在运行期间内，可以向按键分配属性“状态”和“存取等级”。

如果该按键没有指定动作（功能、变量计算、属性改变），则您可以通过回调 <RECALL> 按键退出新配置的操作界面并返回标准应用程序。

举例

```
PRESS(RECALL)
  RECALL.ac = 1
  LM(“屏幕窗口 5”)
END_PRESS
```

2.5.3 运行时改变软键属性

说明

文本、存取等级和状态的属性可以在运行期间在方法中改变：

编程

句法：	SK.st = "文本""	；软键标签
	SK.ac = 存取等级	；软键存取等级
	SK.se = 状态	;软键状态
说明：	分配属性	
参数：	文本	引号中的标签文本
	存取等级	值范围： 0 ... 7
	状态	1: 可见并可操作
		2: 不可操作（灰色标签）
		3: 高亮显示（最后操作的软键）

举例

```
//S(Start)
HS7=(“举例”, ac7, se1)

PRESS(HS7)
  LM(“屏幕窗口 3”)
END_PRESS

//END

//M(屏幕窗口 3/"举例 3: 图形和软键"/"MST.BMP")
HS1=("")
```

```
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("")
VS8=("OK", AC7, SE1)
PRESS(VS8)
EXIT
END_PRESS
//END
```

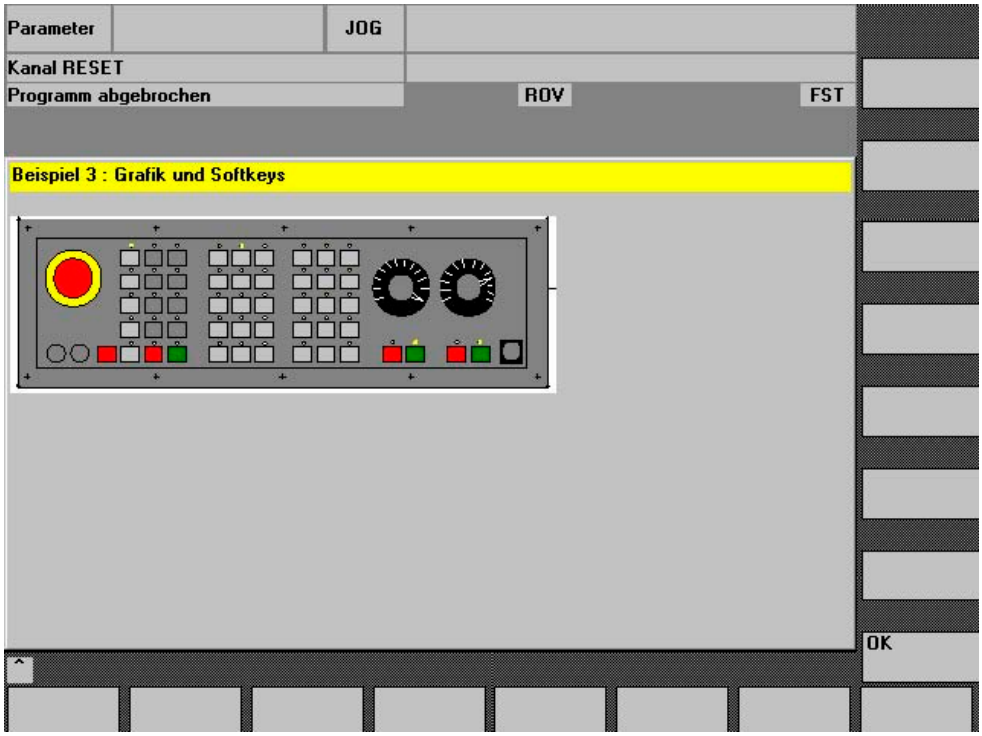

图 2-9　　举例 3： 图形和软键

2.5.4 定义登入软键

和对话框无关的软键

登入软键是和对话框无关的软键，它不由对话框调用，而是在第一个新对话框**之前**设计。为了可以到达登入对话框或者一个登入软键栏，必须对登入软键进行定义。

编程

登入软键的描述块如下构建：

```
//S(Start)                    ; 登入软键开始标记
HS1=(...)                     ; 定义登入软键： 水平软键 SK 1
PRESS(HS1)                    ; 方法
LM...                         ; 功能 LM 或者 LS
END_PRESS                     ; 方式结束
//END                         ; 登入软键结束标记
```

2.5.5 登入软键的功能

和对话框无关的软键的功能

通过登入软键只可以触发特定的功能。

允许下列功能：

- 通过**功能 LM**可以装载另一个对话框。 **LM***("名称"[,"文件"])*
- 通过**功能 LS**可以显示另一个软键栏。 **LS***("名称"[, "文件"][, 合并])*
- 通过**功能 "EXIT"** 可以离开新配置的操作界面并返回标准应用程序。
- 通过**功能 "EXITLS"**可以离开当前的操作界面并装载一个定义的软键栏。
- 通过**功能 "EXE"** 可以在 HMI 高级上调用一个程序，该程序通过 HMI 高级的 OEM 包建立作为应用程序，或者调用自由轮廓编程。 在 HMI 内置 sl 上只能通过 "EXE" 启动自由轮廓编程。

方法 PRESS

在描述块内定义软键并在方法 PRESS 内分配功能 "LM" 或者 "LS"。

如果登入软键定义标记为注释（在行开始处用分号 ; ）或者已删除设计文件，则登入软键无效。

```
//S(Start)                          ; 开始标记
HS6=(“第 1 屏幕窗口”)                ; 水平 SK 6，标签为“第 1 屏幕窗口”
PRESS(HS6)                          ; PRESS 方法用于水平软键 SK 6
  LM(“屏幕窗口 1”)                  ; 装载功能屏幕窗口
                                      1，此时必须在同一文件内定义屏幕窗口 1。
END_PRESS                           ; PRESS 方法结束
HS7=(“第 2 屏幕窗口”)                ; 水平 SK 7，标签为“第 2 屏幕窗口”
PRESS(HS7)                          ; PRESS 方法用于水平软键 SK 7
LM("屏幕窗口 2")                    ; 装载功能屏幕窗口
                                      2，此时必须在同一文件内定义屏幕窗口 2。
END_PRESS                           ; PRESS 方法结束
//END                               ; 登入块的结束标记
```

举例

```
HS1 = ("新的软键栏")
HS2 = ("没有功能")
PRESS(HS1)
  LS("软键栏 1")                    ; 载入新的软键栏
END_PRESS
PRESS (HS2)                         ; 空的 PRESS 方法
END_PRESS
```

参见

装载软键(LS) (页 99)
装载屏幕窗口 (LM) (页 98)

2.6 方法

概述

在对话框和与对话框相关的软键栏中（软键栏由新设计的对话框调用），可以通过不同的事件（退出输入栏，按下软键）触发某些特定的动作。 这些动作设计在方法中。

方法的基本编程按如下方式进行：

说明块	注释	参考章节
PRESS(HS1)	；方法的开始标识	
LM... LS...	；功能	参见章节“功能”
Var1.st = ...	；改变属性	参见章节“软键栏” 和章节“对话框单元”
Var2 = Var3 + Var4 ... EXIT	；通过变量计算	参见章节“定义变量”
END_PRESS	；方法的结束标识	

2.6.1 CHANGE

说明

当变量值已改变时运行 CHANGE（改变）方法。即在
CHANGE（改变）方法中设计变量改变时立即运行的变量计算。

单元特定的 CHANGE 方法和全局的 CHANGE 方法有所不同：

- 当特定变量值已改变时运行**单元特定的 CHANGE 方法**。
  如果系统或者用户变量已分配一个变量，则可以在 CHANGE 方法中循环更新变量值。
- 当改变任意一个变量值且没有设计单元特定的 CHANGE 方法时，运行**全局 CHANGE 方法**。

编程“单元特定”

句法：	CHANGE*(名称)* ... END_CHANGE
说明：	修改指定变量的值
参数：	名称　　变量名称

举例

```
DEF VAR1=(S//////"DB20.DBB1")          ;  Var1 分配一个系统变量
CHANGE(VAR1)
IF VAR1.Val <> 1
  VAR1.st="工具正确!"                  ;  如果系统变量的值 ≠ 1，则变量的短文本为：
                                          工具正确！
    otto=1
ELSE
    VAR1.st="注意 错误！"              ;  如果系统变量的值 = 1，则变量的短文本为：
                                          注意 错误！
    otto=2
ENDIF
  VAR2.Var=2
END_CHANGE
```

编程"全局特定"

句法：	CHANGE() ... END_CHANGE
说明：	改变任意变量值
参数：	- 无 -

举例

```
CHANGE()
EXIT                                   ;  如果任何一个变量值改变，则退出对话框。
END_CHANGE
```

2.6.2 FOCUS

说明

当对话框中聚焦（光标）定位在另一个栏上时，运行 FOCUS 方法。

方法 FOCUS 不允许通过一个导航事件触发。 光标位置只允许在软键 PRESS 块，CHANGE 块，...中改变。 光标移动的反应预先固定定义。

注意

在 FOCUS 块中不允许定位在另一个变量上并且也不允许装载新的对话框。

编程

句法：	FOCUS ... END_FOCUS
说明：	光标：定位
参数：	- 无 -

举例

```
FOCUS
DLGL("聚焦已设定在变量"<< FOC <<"上。")
END_FOCUS
```

2.6.3 LOAD GRID

说明

表格描述可以通过方法 LG 在 LOAD 块内动态提供。

为此可以通过方法 LG
指定一个表格，必须已定义变量作为栅格变量并参考有效且已存在的表格。

编程

句法：	LG *（栅格名称，变量名称 [,文件名称]）*	
说明：	载入表格	
参数：	栅格名称	表格（栅格）名称，用双引号括起
	变量名	应指定表格的变量名称，用双引号括起
	文件名	文件名称，在该文件中定义表格（栅格），用双引号括起 如果表格未在文件中定义，在该文件中也定义变量，则必须指定。

2.6.4 LOAD

说明

在已编译变量定义和软键定义（DEF Var1= ..., HS1= ...）后运行 LOAD 方法。
此时，对话框还未显示。

编程

句法：	LOAD
	...
	END_LOAD
说明：	装载
参数：	- 无 -

举例

```
LOAD                              ;  开始标记
 屏幕窗口 1.Hd =  $85111          ;  分配语言文件中的对话框标题
  VAR1.Min = 0                    ;  分配变量的最小极限值
  VAR1.Max = 1000                 ;  分配变量的最大极限值
END_LOAD                          ;  结束标记
```

2.6.5 UNLOAD

说明

在卸载对话框之前，运行 UNLOAD 方法。

编程

句法：	UNLOAD ... END_UNLOAD
说明：	卸载
参数：	- 无 -

举例

```
UNLOAD
  REG[1] = VAR1                    ;  保存寄存器的变量
END_UNLOAD
```

2.6.6 OUTPUT

说明

当调用功能 "GC" 时，运行 OUTPUT 方法。 在 OUTPUT 方法中变量和辅助变量作为 NC 代码设计。 代码行各个单元的链接用一个空格符实现。

注意

NC 代码可以用文件功能在一个额外的文件中生成并移向 NC。

编程

句法：	OUTPUT*(名称)* ... END_OUTPUT
说明：	输出 NC 程序中的变量
参数：	名称　　OUTPUT 方法的名称

程序段号码和隐藏标记

当要继续保留编程支持激活时在零件程序中直接设置的行号和反编译时的隐藏标记时，OUTPUT 块不允许包含行号和隐藏标记。

在零件程序中通过编辑器更改影响下列行为：

条件	行为
程序段数目保持不变。	程序段号码继续保留。
程序段数目变小。	删除最大的程序段号码。
程序段数目变大。	新的程序段没有程序段号码。

举例

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

2.6.7 PRESS

说明

当已按下相应的软键时，运行 PRESS 方法。

编程

句法：	PRESS*(软键)*		
	...		
	END_PRESS		
名称：	按下软键		
参数：	软键	软键名称： HS1 - HS8 和 VS1 - VS8	
	回调	按键<RECALL>（回调）	
	PU	Page Up	向前翻页
	PD	Page Down	向后翻页
	SL	Scroll Left	光标向左
	SR	Scroll Right	光标向右
	SU	Scroll Up	光标向上
	SD	Scroll Down	光标向下

举例

```
HS1 = ("其他软键栏")
HS2 = ("没有功能")
PRESS(HS1)
  LS("软键栏 1")                                    ;  载入其他软键栏
  Var2 = Var3 + Var1
END_PRESS
PRESS (HS2)
END_PRESS
PRESS(PU)
  INDEX = INDEX -7
CALL("UP1")
END_PRESS
```

2.6.8 举例： 管理带 OUTPUT 块的版本

概述

已有的对话框可以通过扩展 补充附加的变量。
附加的变量在定义中变量名后的圆括号内有一个版本识别号： （0 = 原始，未写入），1 = 版本 1，2 = 版本 2， ...

举例：

```
DEF var100=(R//1)          ;  原始，相当于版本 0
DEF var101(1)=(S//"Hallo")  ;  补充，从版本 1 起
```

在写入 OUTPUT 块时可以参考某个版本状态，与定义的总体性有关。

举例：

```
OUTPUT(NC1)                ;  ;在 OUTPUT 块中仅提供原始变量
OUTPUT(NC1,1)              ;  在 OUTPUT 块中提供原始变量和带有版本标识符 1
                              的补充变量
```

原始的 OUTPUT 块不需要版本标识符，然而也可以记为 0。 OUTPUT(NC1) 相当于 OUTPUT(NC1,0)。 OUTPUT 块中的版本标识符 n 包括所有变量，从原始 0、1、2、... 直至 n。

编程版本标识

```
//M(XXX)                          ;  版本 0 (默认)
DEF var100=(R//1)
DEF var101=(S//"Hallo")
DEF TMP
VS8=("GC")
PRESS(VS8)
GC("NC1")
END_PRESS

OUTPUT(NC1)
var100",,"var101
END_OUTPUT

; ************ 版本 1 , 补充的定义 ***************
//M(XXX)
DEF var100=(R//1)
DEF var101=(S//"Hallo")
DEF var102(1)=(V//"HUGO")
DEF TMP
VS8=("GC")
PRESS(VS8)
GC("NC1")
END_PRESS
...

OUTPUT(NC1)                       ;  ; 原始和其它新的版本
var100","var101
END_OUTPUT
...

OUTPUT(NC1,1)                     ;  版本 1
var100","var101"," var102
END_OUTPUT
```

2.7 功能

概述

在对话框和与对话框相关的软键栏中提供不同的功能，这些功能通过事件（例如：退出输入栏，按下软键）触发并在方法中设计。

子程序

重复的或者其它的设计指令，这些指令总结为一个特定的过程，可以在子程序中设计。
子程序可以随时装载到主程序或者其它子程序中并随时进行编辑，即指令不必多次重复设计。
作为主程序适用于对话框描述块或者软键栏。

PI 服务

通过功能 PI_SERVICE 可以在由 PLC 在 NC 区中启动 PI 服务（程序实例服务）。

外部功能（仅 HMI 高级）

借助于外部功能可以引入其它一些用户特定的功能。 外部功能存放在一个 DLL 文件中并通过设计文件定义行中的条目识别。

参见

PI 服务列表 (页 206)

外部功能（仅 HMI 高级） (页 116)

2.7.1 主动程序（AP）

说明

功能 AP（主动程序）将一个文件从被动 HMI 文件系统传输到 NC 的主动文件系统中。
将文件装载到 NC 中并许可，然后在 HMI 文件系统中删除。 对于 HMI 内置 sl，该功能作用如同设置许可。

编程

句法：	**AP**("*文件*")	
说明：	将被动 HMI 文件系统的文件传输至 NC 的主动文件系统	
参数：	文件	要传输的 HMI 文件的完整路径数据

举例

```
//M(测试GC/"代码生成:")
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)                ;  将 OUTPUT 方法的代码写入文件
                                       \MPF.DIR\MESSEN.MPF 中
END_PRESS
PRESS(HS8)
  D_NAME = "\MPF.DIR\MESSEN.MPF"
  AP(D_NAME)                        ;  文件装载到 NC 中
END_PRESS
```

2.7.2 定义块(//B)

说明

子程序在程序文件中用块标记 //B 标记并通过 //END 结束。
每个块标记可以定义多个子程序。

注意

必须在调用子程序的对话框中定义子程序所使用的变量。

编程

一个块有下列结构：

句法：	**//B**(*块名称*)	
	SUB*(名称)*	
	END_SUB	
	[SUB(*名称*)	
	...	
	END_SUB]	
	...	
	//END	
说明：	定义子程序	
参数：	块名称	块标记名称
	名称	子程序名称

举例

```
//B(PROG1)                          ;  块开始
SUB(UP1)                            ;  子程序开始
  ...
REG[0] = 5                        ;  寄存器 0 赋值 5
  ...
END_SUB                           ;  子程序结束
SUB(UP2)                          ;  子程序开始
  IF VAR1.val=="Otto"
    VAR1.val="Hans"
RETURN
ENDIF
  VAR1.val="Otto"
END_SUB                           ;  子程序结束
//END                             ;  块结束
```

2.7.3 子程序调用(CALL)

说明

通过 CALL 功能可以从方法的任意一个位置调用一个装载的子程序。
允许叠加，即由一个子程序调用另一个子程序。

编程

句法：	CALL*("名称")*	
说明：	调用子程序	
参数：	名称	子程序名称

举例

```
//M(屏幕窗口 1)
VAR1 = ...
VAR2 = ...
LOAD
  ...
LB("PROG1")                          ;    装载块
  ...
END_LOAD
CHANGE()
  ...
CALL("UP1")                          ;    调用并编辑子程序
  ...
END_CHANGE
...
//END
```

2.7.4 检查变量（CVAR）

说明

借助于功能
CVAR（检查变量）可以查询对话框的所有变量或者仅特定变量或者辅助变量是否正确。

在用功能 GC 产生一个 NC 代码之前，查询变量是否包含一个有效值是必要的。

如果变量状态命名符.vld = 1，则变量正确，

编程

句法： **CVAR**(*VarN*)

说明： 检查变量的有效内容

参数： VarN 要检查的变量列表。

可以最多有 29 个变量，各自之间通过逗号隔开。
最大字符长度为 500。

查询结果可能是：

1 = TRUE（真）（所有变量都有有效内容）

0 = FALSE（假）（至少一个变量没有有效内容）

举例

```
IF CVAR == TRUE                ;  检查所有变量
  VS8.SE = 1                   ;  如果所有变量正确，则软键 VS8 可见
ELSE
  VS8.SE = 2                   ;  如果变量包含错误值，则软键 VS8 不可操作
ENDIF

IF CVAR("VAR1", "VAR2") ==
TRUE
                               ;  检查变量 VAR1 和 VAR2
  DLGL ("VAR1 和 VAR2 正确")
                               ;  如果 VAR1 和 VAR2 无错误，则显示对话框行“VAR1 和
                                  VAR2 正确”
ELSE
  DLGL ("VAR1 和 VAR2 不正确")
                               ;  如果 VAR1 和 VAR2 错误，则显示对话框行“VAR1 和 VAR2
                                  不正确”
ENDIF
```

2.7.5 复制程序（CP）

说明

功能 CP（复制程序）在 HMI 文件系统或者 NC 文件系统中复制文件。

注意

对于 NCU 上的 HMI 内置，只能在 NC 文件系统中复制。

编程

句法：	CP*("源文件", "目标文件")*	
说明：	文件：复制	
参数：	源文件	源文件完整的路径数据
	目标文件	目标文件完整的路径数据

举例

```
CP("\MPF.DIR\CFI.MPF ", "\spf.dir\cfi.nc")
```

2.7.6 对话框行(DLGL)

说明

在对话框的对话框行中可以根据确定的情况给出短文本（信息或者输入帮助）。

标准字体大小时允许的字符数量：

- HMI 内置 sl： 约 50 个字符
- HMI 高级： 约 100 个字符

编程

句法：	DLGL*("字符串")*	
说明：	显示对话框行的文本	
参数：	字符串	显示在对话框中的文本

举例

```
IF Var1 > Var2
  DLGL("值太大！")                    ; 如果变量 1>变量 2，则对话框行中显示文本“值过大！”。
ENDIF
```

2.7.7 删除程序（DP）

说明

功能 DP（删除程序）删除一个被动的 HMI 文件系统或者主动的 NC 文件系统的文件。

编程

句法：	DP*("文件"*)	
说明：	文件：删除	
参数：	文件	要删除文件的完整路径数据

举例

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

2.7.8 评估(EVAL)

说明

功能 EVAL 评估作出的输出结果，然后执行。 为此可以首先在运行期间建立表达式。
例如用于变量上的显示存取。

编程

句法：	EVAL*(exp)*	
说明：	评估表达式	
参数：	exp	逻辑表达式

举例

```
VAR1=(S)
VAR2=(S)
VAR3=(S)
VAR4=(S)
CHANGE()
  REG[7] = EVAL("VAR"<<REG[5])        ;  如果 REG[5] 的值为3， 则括号中的表达式为
                                         VAR3。 然后，分配 REG[7] VAR3 的值。
  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
```

2.7.9 执行(EXE)

说明

通过功能 EXE 可以在 HMI 高级上调用一个程序，该程序通过 HMI 高级的 OEM 包建立作为应用程序，或者调用自由轮廓编程。

在 HMI 内置 sl 上只能通过 EXE 启动自由轮廓编程。

注意

功能 EXE 仅提供用在零件程序编辑器中。 为了启动程序，需要在应用程序 INI 文件中的 [CHILDS] 下面输入程序的任务索引，如存放在 REGIE.INI 中。

编程

句法：	**EXE***(程序名称)*	; HMI 高级
	EXE*(GPROC)*	; HMI 内置 sl
说明：	执行程序	
参数：	程序名称	要执行的程序名称

举例

```
PRESS(VS3)
  EXE(GPROC)                ;    启动 GPROC.EXE（自由轮廓编程）
END_PRESS
```

2.7.10 存在程序（EP）

说明

功能 EP（存在程序）检查 NC 文件系统的特定 NC 程序或者 HMI 文件系统中在规定的路径下面是否存在某个文件。

编程

句法：	**EP***("文件")*	
说明：	检查 NC 程序的存在	
参数：	文件	NC 文件系统或者 HMI 文件系统的文件的完整的路径数据
返回值：		待分配查询结果的变量的名称。 查询结果可能是： • M = 文件位于 HMI 上 • N = 文件位于 NC 上 • 空字符串 = 文件既不在 HMI 上，也不在 NC 上

举例

```
EP("\MPF.DIR\CFI.MPF", VAR1)                    ; 检查在 HMI 文件系统内是否存在文件 CFI.MPF
                                                  i。
IF VAR1 == "M"
DLGL("文件位于 HMI 文件系统中")
ELSE
IF VAR1 == "N"
DLGL("文件位于 NC 文件目录")
ELSE
DLGL("文件即不在 HMI 文件目录中也不在 NC 文件目录中")
ENDIF
ENDIF
```

2.7.11 退出对话框(EXIT)

说明

通过功能 EXIT 可以退出对话框并返回主对话框。
如果不存在主对话框，则退出新配置的操作界面并返回标准应用程序。

编程（没有参数）

句法：	**EXIT**
说明：	退出对话框
参数：	- 无 -

举例

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

说明

如果调用带传输变量的当前对话框，则变量值改变并且返回初始对话框。

变量值分别分配给通过功能"LM"从初始对话框传输到后续对话框的变量。 可以最多传输 20 个变量值，各自之间通过逗号隔开。

注意

变量/变量值的顺序必须根据 LM 功能的传递变量顺序进行，以此达到分配明确。
如果一些变量值没有规定，则这些变量不改变。 改变的传输变量在功能 LM 后立即在初始对话框中生效。

使用传输变量编程

句法：	**EXIT***[(VARx)]*
说明：	退出对话框，传输一个或者多个变量
参数：	VARx 计算变量

举例

```
//M(屏幕窗口 1)
...
PRESS(HS1)
  LM("屏幕窗口 2","CFI.COM",1, POSX, POSY, 直径)
                                    ; 中断屏幕窗口 1 并显示屏幕窗口 2。 传输变量
                                      POSX、POSY 和直径。
DLGL(“退出屏幕窗口 2”)               ; 从屏幕窗口 2 返回后在屏幕窗口 1
                                      的对话框行中显示文本： 退出屏幕窗口 2。
END_PRESS
...
//END

//M(屏幕窗口 2)
...
PRESS(HS1)
  EXIT(5, , 计算_直径)
                                    ; 退出屏幕窗口 2 并返回到屏幕窗口 1 的 LM 后的行中。
                                      此时，分配值 5 给变量 POSX
                                      并分配变量“计算_直径”的值给变量“直径”。 变量 POSY
                                      包含当前值。
END_PRESS
...
//END
```

2.7.12 退出装载软键(EXITLS)

说明

通过功能 EXITLS 可以离开当前的操作界面并装载一个定义的软键栏。

编程

句法：	EXITLS(*"软键栏"[, "路径名"*])	
说明：	退出时装载软键栏	
参数：	软键栏	待装载的软键栏名称
	路径名	要装载的软键栏目录路径

举例

```
PRESS(HS1)
    EXITLS( "软键栏 1", "AEDITOR.COM" )
END_PRESS
```

2.7.13 生成代码(GC)

说明

功能 GC（生成代码） 由 OUTPUT（输出）方法生成 NC 代码。

编程

句法：	GC("*名称"[,"目标文件"][,Opt],[Append])*	
说明：	生成 NC 代码	
参数：	名称	OUTPUT（输出）块名称用作代码生成的基础
	目标文件	HMI 或者 NC 文件系统的目标文件路径数据。 如果未规定目标文件（只能在编程支持中），则在该位置上写入代码，在该位置处光标停留在当前打开的文件中。
	可选	可选注释生成 0: （预设）建立用于反编译的带有注释的预置代码。 1: 在生成的代码中不生成注释。 **说明：** 该代码无法反编译。

Append　　只有当规定目标文件情况下，该参数才有意义。
0: （预设）当文件已存在情况下，删除旧的内容。
1: 当文件已存在情况下，在文件开始处写入新的代码。
2: 在文件已存在的情况下，在结束处加入新的代码。

举例

```
//M(测试GC/"代码生成:")
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)                    ; 将 OUTPUT 方法的 NC 代码写入文件
                                          C:\MPF.DIR\MESSEN.MPF 中
                                          Cycle123(123, -6)
                                          M30

END_PRESS
```

反编译

- **目标文件没有数据：**

  功能 GC 只能在编程支持中使用并待当前编辑器中打开的文件内写入 NC 代码。 NC 代码可以反编译。 如果功能 GC
  在没有目标文件数据情况下在“补充操作界面”下设计，则此时发出故障信息。

- **目标文件数据：**

  从 OUTPUT（输出）块生成的代码输入在目标文件中。 如果不存在目标文件，则在 NC 文件系统中设立。 如果目标文件在 HMI 文件系统中，则该文件存放在硬盘上（仅 HMI 高级）。 不设立使用注释行（反编译需要的信息），即无法进行反编译。

目标文件的数据特点

基本上规定一个目标文件有三个不同的类型：

- **NC 格式：** /_N_MPF_DIR/_N_MY_FILE_MPF

  只适用于 HMI 内置 sl。

  该文件存放在 NC 上的目录 MPF 下。

- **DH 格式：** /MPF.DIR\MY_FILE.MPF

  适用于 HMI 高级 和 HMI 内置 sl。

  对于 HMI 内置 sl，目标文件的数据以 NC 格式转换并且文件存放在 NCU 上。

  对于 HMI 高级或者对于HMI 内置 WIN32，文件存放在数据管理路径下。

- **DOS 格式：** d:\abc\my_file.txt 或者 \\RemoteRechner\files\my_file.txt

  适用于 HMI 高级 和 HMI 内置 sl。

  文件写入在硬盘上规定的目录中或者指定的计算机上，此时硬盘上的目录或者远程计算机必须已存在。

  对于 HMI 内置 sl，可以以该格式仅写入在 RAM
  驱动器上或者网络中的一台计算机上，前提是已配置好网络连接。

注意

无效的变量在生成的 NC
代码中产生一个空字符串，如果读取该代码，在日志中出现一条出错信息。

反编译时的特殊情况

在子对话框中无法调用 GC
功能，因为在子对话框中可以使用源自主对话框的变量，然而在直接调用时并不存在。

在通过编辑器手动介入生成的代码时不允许改变由代码生成产生的值的符号数目。
这些改变会妨碍到反编译。

解决方法：

1. 反编译
2. 通过设计的对话框输入修改（例如：99 → 101）
3. GC

参见

反编译 (页 107)

2.7.14 装载数组(LA)

说明

通过功能 LA（装载数组）可以从另一个文件装载一个数组。

编程

句法：	**LA***(名称 [, 文件])*	
说明：	从文件装载数组	
参数：	名称	待装载数组的名称
	文件	在该文件中定义数组

注意

如果在当前的设计文件中，一个数组要由另一个设计文件中的数组替代，则数组名称必须相同。

举例

```
                                    ;  文件 maske.com 部分摘录
DEF VAR2 = (S/*ARR5/“关闭”，“转换栏”)
PRESS(HS5)
  LA("ARR5","arrayext.com")         ;  从文件 arrayext.com 中装载数组 ARR5
  VAR2 = ARR5[0]                    ;  代替 "Aus"/"Ein" ( “关”/“开” ) 显示在 VAR2 的转换栏
                                       "上"/"下"/"右"/"左"
END_PRESS
//A(ARR5)
("关"/"开")
//END
                                    ;  文件 arrayext.com 部分摘录
//A(ARR5)
("上"/"下"/"右"/"左")
//END
```

注意

请注意，在通过 LA 功能分配给变量转换栏另一个数组后，必须赋予一个有效值给变量。

2.7.15 装载块（LB）

说明

通过功能 LB（装载块）可以在运行期间内装载带有子程序的块。 首先 LB 要在 LOAD（装载）方法中设计，由此可以随时调用装载的子程序。

注意

子程序也可以直接在对话框中定义，如果那样则不必装载。

编程

句法：	LB*("块名称"[,"文件"])*	
说明：	运行时装载子程序	
参数：	块名称	块标记名称
	文件	设计文件的路径数据
		预设 = 当前的设计文件

举例

```
LOAD
LB("PROG1")                    ;  在当前的设计文件中查找块“PROG1”，接着装载块。
  LB("PROG2","XY.COM")         ;  在设计文件 XY.COM 中查找块“PROG2”，接着装载块。
END_LOAD
```

2.7.16 装载屏幕窗口 (LM)

说明

通过功能 LM 可以载入一个新的对话框。

主对话框/子对话框

能够调用其他对话框并且不能自行结束的对话框称为主对话框。
由主对话框调用的对话框是子对话框。

编程

句法： **LM***("名称"[,"文件"] [,MSx [, VARx]])*

说明： 载入对话框

参数：

名称	待装载对话框的名称
文件	设计文件的路径数据（HMI 文件系统或者 NC 文件系统）；标准设置： 当前的设计文件
MSx	对话框切换模式
0:	（预设）退出当前对话框，装载并显示新的对话框。 在 EXIT 时（参见“退出”功能）返回到标准应用程序。 通过参数 MSx 可以确定在对话框切换时是否退出当前对话框。 保留当前的对话框，可以将变量传递到新的对话框中。 使用参数 MSx 的优点在于，在切换时无需总是重新初始化对话框，而是保留当前对话框的数据和设计，简化数据传输
1:	自功能 LM 起中断当前对话框，装载并显示新的对话框。 退出时关闭子对话框并且返回到主对话框的中断位置。 中断时，不处理主对话框中的 UNLOAD（卸载）块。
VARx	前提条件： MS1 列出可以从主对话框传送到子对话框的变量。 可以最多传输 20 个变量，各自之间通过逗号隔开。

注意

参数 VARx 总是只传递变量值，也就是说变量可以在子对话框中读写，但是不可见。
由子对话框向主对话框返回传递变量可以通过功能 EXIT（退出）进行。

举例

```
PRESS(HS1)
  LM("屏幕窗口 2","CFI.COM",1, POSX, POSY, 直径)
                                ; 中断屏幕窗口 1 并显示屏幕窗口 2：传输变量
                                  POSX、POSY 和直径。
DLGL(“退出屏幕窗口 2”)          ; 从屏幕窗口 2 返回后在屏幕窗口 1 的对话框行中显示文本：
                                  退出屏幕窗口 2。
END_PRESS
```

2.7.17 装载软键(LS)

说明

通过功能 LS 可以显示另一个软键栏。

编程

句法：	**LS***("名称"[, "文件"][, 合并])*	
说明：	显示软键栏	
参数：	名称	软键栏名称
	文件	设计文件的路径数据（HMI 文件系统或者 NC 文件系统）
		预设置：当前的设计文件
	Merge	
		0: 删除所有存在的软键，输入重新设计的软键。
		1: （预设）仅重新设计的软键覆盖存在的软键，而其他软键（= HMI 或者 ShopMill/ShopTurn 的标准软键）的功能和文本保持不变。

举例

```
PRESS(HS4)
  LS("软键栏 2",,0)             ; 软键栏 2 覆写存在的软键栏，删除所有存在的 软键
END_PRESS
```

注意事项

只要编译器还没有显示出对话框，即还没有处理 LM
功能，则在登入软键描述块和软键栏描述块的 PRESS（按压）方法中总是仅设计一个 LS
命令或者 LM 命令，并且没有其它动作。

功能 LS 和 LM 仅允许在软键
PRESS（按压）块中调用，然而不作为导航键（PU、PD、SL、SR、SU、SD）上的反应。

参见

登入软键的功能 (页 71)

2.7.18 被动程序（PP）

说明

功能 PP（被动程序）将一个文件从 NC 的主动文件系统传输到 HMI 高级的被动文件系统中。
文件在执行功能 PP 后不再存在于 NC 的主动文件系统中。 对于 HMI 内置
sl，该功能作用起效，如删除使能。

编程

句法：	**PP***("文件*")	
说明：	将文件从 NC 的主动文件系统中传送到 HMI 高级的被动文件系统中。	
参数：	文件	要传输的 NC 文件的完整路径数据

举例

```
PP("\MPF.DIR\MESSEN.MPF")
```

2.7.19 读取 NC PLC (RNP), 写入 NC PLC (WNP)

说明

通过指令 RNP (读取 NC PLC) 可以读取 NC 或者 PLC 变量或者机床数据。

编程

句法：	**RNP** (*"系统或者用户变量",值*)	
说明：	读取 PLC 或者 NC 变量或者机床数据	
参数：	系统或者用户变量	NC 或者 PLC 变量名称
	值	要写入系统或者用户变量中的值。 如果值类型为字符串，则必须以双引号括起。

举例

```
VAR2=RNP("$AA_IN[2]")                    ;     读取 NC 变量
```

说明

通过指令 WNP (写入 NC PLC) 可以写入 NC 或者 PLC 变量或者机床数据。

在每次处理功能 WNP 都重新存取 NC 变量或 PLC 变量，即：总是在 CHANGE 方法下存取 NC 变量或 PLC 变量。 这在系统或者用户变量经常改变值情况下有意义。 如果只进行一次 NC/PLC 存取，则必须在 LOAD（装载）或者 UNLOAD（卸载）方法下设计。

编程

句法：	**WNP***("系统或者用户变量", 值)*	
说明：	写入 PLC 或者 NC 变量或者机床数据	
参数：	系统或者用户变量	NC 或者 PLC 变量名称
	值	要写入系统或者用户变量中的值。 如果值类型为字符串，则必须以双引号括起。

举例

```
WNP("DB20.DBB1",1)                       ;     写入 PLC 变量
```

2.7.20 多次读取 NC PLC (MRNP)

说明

用命令 MRNP 可以通过在寄存器中一次存取输入多个系统或者 BTSS 变量。
这种存取比通过单个存取速度明显加快。 系统或者 BTSS 变量必须在由自身区域的 MRNP 命令中。

系统或者 BTSS 变量区域分为如下几种：

- 一般 NC 数据 ($MN..., $SN.., /nck/...)
- 通道专用的 NC 数据 ($MC..., $SC.., /channel/...)
- PLC 数据 (DB..., MB.., /plc/...)
- 相同轴专用的 NC 数据 ($MA..., $SA..)

编程

句法：	**MRNP***(变量名称 1*变量名称 2[*...], 寄存器编号)*
说明：	读取多个变量
参数：	变量名称 用"*"作为分隔符。 按照命令中变量名称的顺序，采用寄存器 REG[寄存器编号] 中的值和下列值。 关系到： 第一个变量的值在 REG[寄存器编号] 中。 第二个变量的值在 REG[寄存器编号 + 1]或

注意事项

必须注意，变量列表最多为 500 行，并且寄存器的数量受限。

举例

```
MRNP("$R[0]*$R[1]*$R[2]*$R[3]",1)     ;     REG[1] 至 REG[4] 以变量值 $R[0] 至 $R[3] 描述。
```

读取显示机床数据：

显示机床数据通过 RNP（$MM...）在 LOAD（装载）块中读取。

显示机床数据上一般的读写访问不通过"补充操作界面"进行。

注意

用户变量不允许与系统变量或者 PLC 变量有相同的名称。

NC 变量

提供所有的机床和设置数据以及 R 参数，但是仅提供某些系统变量（参见附录列表）。 在 HMI 高级中，在操作范围参数/系统变量/编辑视图/插入变量中找到可获取的系统变量。

可访问所有全局的和通道专用的用户变量（GUD）。 本地和程序全局的用户变量无法编辑。

机床数据	
全局机床数据	$MN_...
轴专用的机床数据	$MA_...
通道专用的机床数据	$MC_...

设定数据	
全局设定数据	$SN_...
轴专用的设定数据	$SA_...
通道专用的设定数据	$SC_...

系统变量	
R 参数 1	$R[1]

PLC 变量

提供所有 PLC 数据。

PLC 数据	
数据模块 x 的字节 y 位 z	DBx.DBXy.z
数据模块 x 的字节 y	DBx.DBBy
数据模块 x 的字 y	DBx.DBWy
数据模块 x 的双字 y	DBx.DBDy
数据模块 x 的实数 y	DBx.DBRy
标志器字节 x 位 y	Mx.y
标志器字节 x	MBx
标志器字 x	MWx
标志器双字 x	MDx
输入字节 x 位 y	Ix.y 或者 Ex.y
输入字节 x	IBx 或者 EBx
输入字 x	IWx 或者 EWx

PLC 数据	
输入双字 x	IDx 或者 EDx
输出字节 x 位 y	Qx.y 或者 Ax.y
输出字节 x	QBx 或者 ABx
输出字 x	QWx 或者 AWx
输出双字 x	QDx 或者 ADx
数据模块 x 中带有长度 z 的字符串 y	DBx.DBSy.z

2.7.21 刷新

说明

在所有块内可以调用功能“刷新”。 该功能没有参数。

生效方式：

- 所有显示区中激活的变量内容（输入/输出栏）给定新的背景和前景。
- 所有显示区中可见的激活的短文本，图像文本和单位文本重新给定，之前可不删除该文本的背景。

编程

句法：	**刷新**
说明：	刷新输入//输出栏和输入文本
参数：	- 无 -

2.7.22 寄存器(REG)

寄存器说明

寄存器用于在两个不同的对话框之间切换数据。
寄存器分配给每个对话框并且在载入第一个对话框时生成，以 0 或空字符串预占。

注意

寄存器不允许直接用在 OUTPUT（输出）块中用于 NC 代码生成。

编程

句法：	REG*[x]*	
说明：	定义寄存器	
参数：	x	寄存器编号，以 x = 0...19 类型： REAL 或者 STRING = VARIANT x ≥ 20的寄存器由西门子使用。

寄存器值说明

寄存器值的分配在方法中设计。

注意

如果由一个对话框通过功能 LM
生成另一个对话框，则寄存器的内容自动接受入新的对话框中并在第二个对话框中用于其它计算。

编程

句法：	*名称*.val = *寄存器值*	
	或者	
	名称 = *寄存器值*	
说明：		
参数：	名称	寄存器名称
	寄存器值	寄存器的值

举例

```
UNLOAD
  REG[0] = VAR1                    ;    寄存器 0 赋值为变量 1 的值
END_UNLOAD

UNLOAD
REG[9].VAL = 84                    ;    寄存器 9 赋值为变量 84 的值
END_UNLOAD
                                   ;    在下列对话框中该寄存器可以在方法中重新分配本地变量。
LOAD
  VAR2 = REG[0]
END_LOAD
```

寄存器状态说明

通过状态特性可以在设计中查询寄存器是否包含一个有效值。

此外，当一个对话框作为主对话框使用时，可以使用寄存器的状态查询向一个寄存器中仅写入一个值。

编程

句法：	*名称*.vld	
说明：	该属性仅可读。	
参数：	名称	寄存器名称
返回值：		查询结果可能是：
	FALSE =	无效值
	TRUE =	有效值

举例

```
IF REG[15].VLD == FALSE                ;  查询寄存器值的有效性
REG[15] = 84
ENDIF
VAR1 = REG[9].VLD                      ;  向 Var1 分配 REG[9] 状态查询的值。
```

2.7.23 RETURN

说明

通过 RETURN 功能可以提前取消当前的子程序处理并返回到最后一次 CALL 命令的跳转位置。

如果子程序中没有设计 RETURN，则执行子程序，直至结束然后返回到跳转位置。

编程

句法：	**RETURN**
说明：	返回跳转位置
参数：	- 无 -

举例

```
//B(PROG1)                    ;  块开始
SUB(UP2)                      ;  子程序开始
  IF VAR1.val=="Otto"
    VAR1.val="Hans"
RETURN                        ;  如果变量值 = Otto，则分配值 "Hans" 给变量，
                                 子程序在此位置结束。
ENDIF
  VAR1.val="Otto"             ;  如果变量值 ≠ Otto，则分配值 "Otto" 给变量。
END_SUB                       ;  子程序结束
//END                         ;  块结束
```

2.7.24 反编译

说明

在编程支持中可以**反编译**功能 GC 生成的 NC 代码并在所属输入对话框中的输入/输出栏中再次显示变量值。

编程

来自 NC 代码的变量接受至对话框中。 此时对来自 NC 代码的变量值与来自设计文件的计算的变量值加以比较。
如果存在不一致，则在日志文件中给出错误信息，因为在生成的 NC 代码中数值已改变。

如果一个变量在 NC 代码中多次存在，则在反编译时总是分析该变量最后一次出现的值。
另外在日志文件中给出警告。

代码生成时，不在 NC 代码中使用的变量作为使用注释存储。
通过使用注释标记所有反编译时需要的信息。 使用注释不得更改。

注意

NC 代码块和使用注释如果在一行的启始处开始，则只能反编译。

举例：

在程序中有下列 NC 代码：

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

在零件程序中存放下列代码：

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

在反编译时读取编辑器：

```
X101 Y200
X222 Y0                    ; 在零件程序中更改 X 的值 (X101 → X222)
```

在输入对话框中给出下列 VAR1 值： VAR1 = 222

参见

生成代码(GC) (页 93)

2.7.25 向前/后查找(SF, SB)

说明

通过功能 **向前/后查找（SF, SB）**可以在编辑器当前的 NC 程序中从当前光标位置开始查找某个字符串并给出其值。

编程

句法：	**SF***("字符串")*
名称：	**S**earch **F**orward: 从当前光标位置向前查找
句法：	**SB***("字符串")*
名称：	**S**earch **B**ackward: 从当前光标位置向后查找
参数：	字符串　　要查找的文本

查找规则：

- 要查找的字符串和数值的单元在 NC 程序中前后必须有空格。
- 查找对象无法在注释中和字符串内查找。
- 给出的值必须是一个数字表达式，表达式形式“X1=4+5”无法识别。
- 可识别十六进制常数的形式 X1=‘HFFFF'、二进制常数的形式 X1=‘B10010' 和指数常数的形式 X1=‘-.5EX-4’。
- 在字符串和数值之间有下列符号，可以给出字符串值：
  - 无
  - 空格键
  - 等号

举例

允许以下的写入方式：

```
X100 Y200                    ;   变量 Abc 包含数值 200
Abc = SB("Y")
X100 Y 200                   ;   变量 Abc 包含数值 200
Abc = SB("Y")
X100 Y=200                   ;   变量 Abc 包含数值 200
Abc = SB("Y")
```

2.7.26 选择程序（SP）

说明

功能 SP（选择程序）选择一个主动的 NC
文件系统文件，以对其进行处理。即该文件之前必须就已装载在 NC 中。

编程

句法：	SP*("文件")*	
名称：	**S**elect **P**rogram	
参数：	“文件”	NC 文件的完整路径数据

举例

```
//M(测试GC/"代码生成:")
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)                    ;  将 OUTPUT 方法的代码写入文件
                                            \MPF.DIR\MESSEN.MPF 中
END_PRESS
PRESS(HS8)
 AP("\MPF.DIR\MESSEN.MPF")               ;  文件装载到 NC 中
 SP("\MPF.DIR\MESSEN.MPF")               ;  文件选择
END_PRESS
```

2.7.27 字符串功能

概述

以下的功能允许进行字符串处理：

- 确定字符串长度
- 查找字符串中的一个字符
- 由左提取部分字符串
- 由右提取部分字符串
- 由字符串中间提取部分字符串
- 替换部分字符串

功能 LEN： 字符串长度

句法：	**LEN***(字符串 | 变量名称)*	
说明：	确定一个字符串的字符数目。	
参数：	字符串	每个有效的字符串表达式。 对于一个空字符串，返回为零。
	变量名称	每个有效的和表示的变量名称
	仅允许两个可行参数中的一个。	

举例

```
DEF VAR01
DEF VAR02

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

功能 INSTR： 查找字符串中的字符

句法：	**INSTR***(开始, 字符串 1, 字符串 2 [,方向])*	
说明：	查找字符	
参数：	启动	从字符串 1 向字符串 2 查找的开始位置。 如果从字符串 2 的开头开始查找，则指定为 0。
	字符串 1	要查找的字符。
	字符串 2	在该字符串链中查找
	方向（可选）	查找的方向
		0: 从左向右（预设）
		1: 从右到左
	如果字符串 2 中没有包含 字符串 1，则返回数值 0。	

举例

```
DEF VAR01
DEF VAR02

LOAD
VAR01="HALLO/WELT"
VAR02=INST(1,"/",VAR01)                ;     结果 = 6
END_LOAD
```

功能 LEFT： 左边字符串

句法：	**LEFT***(字符串, 长度)*	
说明：	LEFT 返回一个字符串，该字符串从字符串左侧开始包含指定的字符数。	
参数：	字符串	字符串或者带有要处理的字符串的变量
	长度	要读取的字符数目

举例

```
DEF VAR01
DEF VAR02
LOAD
VAR01="HALLO/WELT"
VAR02=LEFT(VAR01,5)                    ; 结果 = "HALLO"
END_LOAD
```

功能 RIGHT： 右边字符串

句法： **RIGHT***(字符串, 长度)*
说明： RIGHT
返回一个字符串链，该字符串链从字符串右侧开始包含指定的字符数。
参数：

字符串	字符串或者带有要处理的字符串的变量
长度	要读取的字符数目

举例

```
DEF VAR01
DEF VAR02
LOAD
VAR01="HALLO/WELT"
  VAR02=LEFT(VAR01,4)                    ; 结果 = "WELT"
END_LOAD
```

功能 MIDS： 中间字符串

句法： **MIDS***(字符串, 开始 [, 长度])*
说明： MIDS
返回一个字符串，该字符串从字符串的指定位置开始包含指定的字符数。
参数：

字符串	字符串或者带有要处理的字符串的变量
开始	开始，从字符串链中该处开始读取
长度	要读取的字符数目

举例

```
DEF VAR01
DEF VAR02
LOAD
VAR01="HALLO/WELT"
  VAR02=LEFT(VAR01,4,4)                  ; 结果 = "LO/W"
END_LOAD
```

功能 REPLACE： 替换字符

句法： **REPLACE** *(字符串, 查找字符串, 替换字符串 [, 开始 [, 计数]])*

说明： 功能 REPLACE
用另一个字符/字符链替代字符串中的一个字符/字符链。

参数：

字符串	待通过替换字符串替换查找字符串的字符串。
查找字符串	需被替代的字符串
替代字符串	替代字符串（位于查找字符串位置）
开始	查找和替换的开始位置
计数器	从开始位置起要开始查找的查找字符串的字符数量。

返回值：

字符串 = 空字符串	复制字符串
查找字符串 = 空字符串	复制字符串
替换字符串 = 空字符串	复制字符串，在该字符串中删除所有出现的查找字符串
开始 > Len(长度)	空字符串
计数 = 0	复制字符串

2.7.28 PI 服务

说明

通过功能 PI_SERVICE 可以在由 PLC 在 NC 区中启动程序实例服务（PI 服务）。

一般编程

句法：	**PI_SERVICE** (*服务, n 参数*)	
说明：	执行 PI 服务	
参数：	服务	PI 服务的标识
	n 参数	n 个 PI 服务参数组成的参数列表。 参数用逗号隔开。

举例

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

启动 OEM 服务

命令 PI_START 根据 OEM 文献执行一个 PI 服务。

编程

句法：	PI_START("*传输字符串*")	
说明：	执行 PI 服务	
参数：	“传输字符串”	传输的字符串与 OEM 文献相反，应用双引号括起。

举例

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

注意

通道相关的 PI 服务总是与当前的通道有关。

刀具功能（TO 区）的 PI 服务总是以分配到当前通道的 TO 区为参考。

参见

PI 服务列表 (页 206)

2.7.29 外部功能（仅 HMI 高级）

说明

通过调用可以使用更多用户特定的功能。 外部功能存放在一个 DLL 文件中并通过设计文件定义行中的条目识别。

注意

一个外部功能必须至少有一个返回参数。

编程

句法：	FCT*功能名称 =* *("文件"/返回类型/固定调用参数的类型/可变调用参数的类型)*	
说明：	定义其他外部功能	
参数：	功能名称	外部功能的名称
	文件	DLL 文件的完整路径数据
	返回类型	功能返回值数据类型
	R, I, S, C, B.	**固定**调用参数的数据类型和返回值。 数据类型用逗号隔开。
	变量或者寄存器	**可变**调用参数的数据类型
句法：	FCT*功能名称（调用参数）*	
	调用参数	所有调用参数列表。 参数用逗号隔开。

举例

```
//M(屏幕窗口 1)
DEF VAR1 = (R)
DEF VAR2 = (I)
DEF RET = (I)
FCT InitConnection = ("c:\user\mydll.dll"/I/R,I,S/I,S)
                          ;  定义外部功能“InitConnection”。 定义的外部功能的数据类型。
                             返回值是整数，固定调用参数是实数、整数和字符串，可变调用参
                             数是整数和字符串。
LOAD
  RET = InitConnection(VAR1+SIN(VAR3),13,"Servus",VAR2,
  REG[2])
                          ;  将带调用参数 VAR1+SIN(VAR3), 13, Servus, VAR2 和 REG[2]
                             的外部功能 "InitConnection" 的值分配给 RET。
END_LOAD
```

DLL 文件中的摘录

```
void __export WINAPI InitConnection(ExtFctStructPtr FctRet, ExtFctStructPtr
FctPar, char cNrFctPar)
FctRet->value.i   功能的返回值
FctPar[0]->value.r   第 1 个参数 (VAR1+SIN(VAR3))
FctPar[1]->value.i   第 2 个参数(13)
FctPar[2]->value.s  第 3 个参数 ("Servus")
FctPar[4]->value.i   第 4 个参数(参考 VAR2)
FctPar[5]->value.s  第 5 个参数(参考 REG[2])
cNrFctPar   参数数量 (5)
```

2.7.30 编程举例

编程

```
//S(Start)
HS7=(“举例”, se1, ac7)
PRESS(HS7)
  LM(“屏幕窗口 4”)
END_PRESS
//END
//M(屏幕窗口 4/"举例 4: 机床操作面板"/"MST.BMP")
DEF byte=(I/0/0/"输入字节, 标准
=0","字节号.:","/wr1,li1///380,40,100/480,40,50)
DEF Feed=(IBB//0/"","进给倍率","/wr1//"EB3"/20,180,100/130,180,100),
Axistop=(B//0/"","进给停止","/wr1//"E2.2"/280,180,100/380,180,50/0,11)
DEF Spin=(IBB//0/"","主轴倍率","/wr1//"EB0"/20,210,100/130,210,100),
spinstop=(B//0/"","主轴停止","/wr1//"E2.4"/280,210,100/380,210,50/0,11
)
DEF custom1=(IBB//0/"","用户定义键
1","/wr1//"EB6"/20,240,100/130,240,100)
DEF custom2=(IBB//0/"","用户定义键
2","/wr1//"EB7"/20,270,100/130,270,100)
DEF By1
DEF By2
DEF By3
DEF By6
DEF By7

HS1=("输入字节", SE1, AC4)
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("")
VS8=("OK", SE1, AC7)
```

```
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
```

```
PRESS(VS8)
EXIT
END_PRESS
```

结果

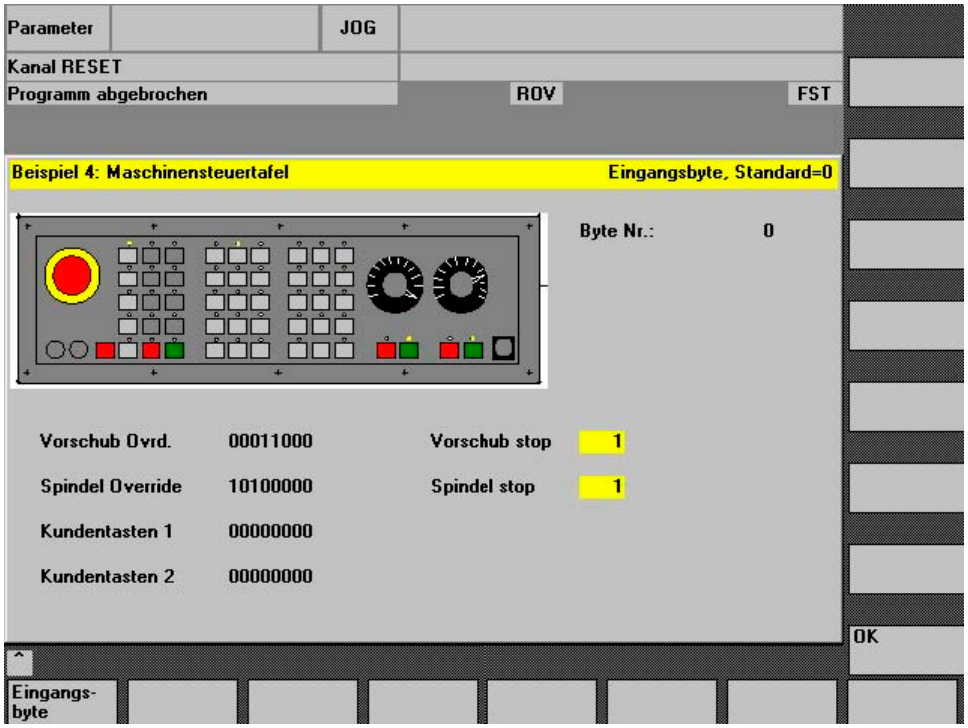

图 2-10 （机床控制面板）

注意

其它举例可在随附的工具箱中找到。

2.8 运算符

概述

在编程时可以使用以下几种运算符：

- 数学运算符
- 比较运算符
- 逻辑（布尔）运算符
- 位运算符
- 三角函数功能

2.8.1 数学运算符

概述

数学运算符	名称
+	加法
-	减法
*	乘法
/	除法
MOD	模数运算
( )	括号
AND	和运算符
OR	或运算符
NOT	非运算符
ROUND	带有小数位的数四舍五入

举例: `VAR1.VAL = 45 * (4 + 3)`

ROUND

在执行对话框设计时，用 ROUND 指令把数值四舍五入直至小数点后面的 12 位。
小数点后位置在变量栏不能显示。

使用

ROUND 通过用户的两个参数来控制:

```
VAR1 = 5,2328543
VAR2 = ROUND( VAR1, 4 )
```

结果： `VAR2 = 5,2339`

VAR1 包含在四舍五入的数中。 参数"4"给出了结果中的小数点后的位数，保存在 VAR2 中。

三角函数功能

三角函数功能	名称
SIN(x)	正弦 x
COS(x)	余弦 x
TAN(x)	正切 x
ATAN(x, y)	反正切 x/y
SQRT(x)	平方根 x
ABS(x)	绝对值 x
SDEG(x)	换算为度数
SRAD(x)	换算为弧度

注意

这些函数处理弧度。 为了换算，可以使用函数 SDEG() 和 SRAD()。

举例: `VAR1.VAL = SQRT(2)`

常量

常量	
PI	3.14159265358979323846
FALSE	0
TRUE	1

举例: `VAR1.VAL = PI`

比较运算符

比较运算符	
==	相等
<>	不等
>	大于
<	小于
>=	大于等于
<=	小于等于

举例：

```
IF VAR1.VAL == 1
  VAR2.VAL = TRUE
ENDIF
```

条件

层叠深度没有限制。

带有一个命令的条件：

```
IF
...
ENDIF
```

带有两个命令的条件：

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

2.8.2 位运算符

概述

位运算符	名称
BOR	位方式 OR
BXOR	位方式 XOR
BAND	位方式 AND
BNOT	位方式 NOT
SHL	位向左移动
SHR	位向右移动

运算符 SHL

通过运算符 SHL(SHIFT LEFT)向左移动位。
此时要移动的值或者移动步数可以直接规定或者作为变量规定。
如果达到数据格式极限，则已超出位，没有出错信息。

使用

句法：	变量 = *值* SHL*步数*	
说明：	向左移动	
参数：	值	要移动的值
	步数	移动步数

举例

```
PRESS(VS1)
VAR01 = 16 SHL 2                ;  结果 = 64
VAR02 = VAR02 SHL VAR04         ;  VAR02 内容转换为 32 位格式，无正负之分;并且位向左移动
                                   VAR04 内容。 接着 32 位值重新转换回变量 VAR02
                                   的格式。
END_PRESS
```

运算符 SHR

通过运算符 SHR (SHIFT RIGHT)向右移动位。
此时要移动的值或者移动步数可以直接规定或者作为变量规定。
如果达到数据格式极限，则已超出位，没有出错信息。

使用

句法：	变量 = *值* SHR*步数*	
说明：	向右移动	
参数：	值	要移动的值
	步数	移动步数

举例

```
PRESS(VS1)
VAR01 = 16 SHR 2                ;  结果 = 4
VAR02 = VAR02 SHR VAR04         ;  VAR02 内容转换为 32
                                   位格式，无正负之分并且位向右移动 VAR04 内容。 接着
                                   32 位值重新转换回变量 VAR02 的格式。
END_PRESS
```

编程支持

3

3.1 编程支持提供什么?

概述

在用 ASCII 编辑器建立 NC 程序时编程支持可以辅助编程人员开展工作。
在编程支持下也可以设计自身的操作界面。 创建操作界面通过 ASCII 编辑器和方法“补充操作界面”工具进行。

为此提供下列标准化工具：

- 循环辅助
- 自由轮廓编程
- 轮廓编程
- 反编译
- 模拟

注意

循环支持（//C...）出于兼容性的原因另外通过以前的描述语言支持，并且不用“补充操作界面”语句。

建立新对话框

原则上，要借助于“补充操作界面”工具设计新的操作界面。
在编程支持下现有的另外一些区别在本章节中有相关描述。

设计文件

编程支持的新对话框的描述在设计文件 AEDITOR.COM 中定义。

- 新建立的对话框可以在编辑器基本画面中通过 5 个登入软键（水平软键 2、3、4、5、6）显示。
- 水平软键 2 至 5 预设有标准化的“轮廓”、“钻削”、“铣削”和“车削”。
- 水平软键 14 和 15（扩展栏的软键 6 和 7）预设为“车削测量”和“铣削测量”。

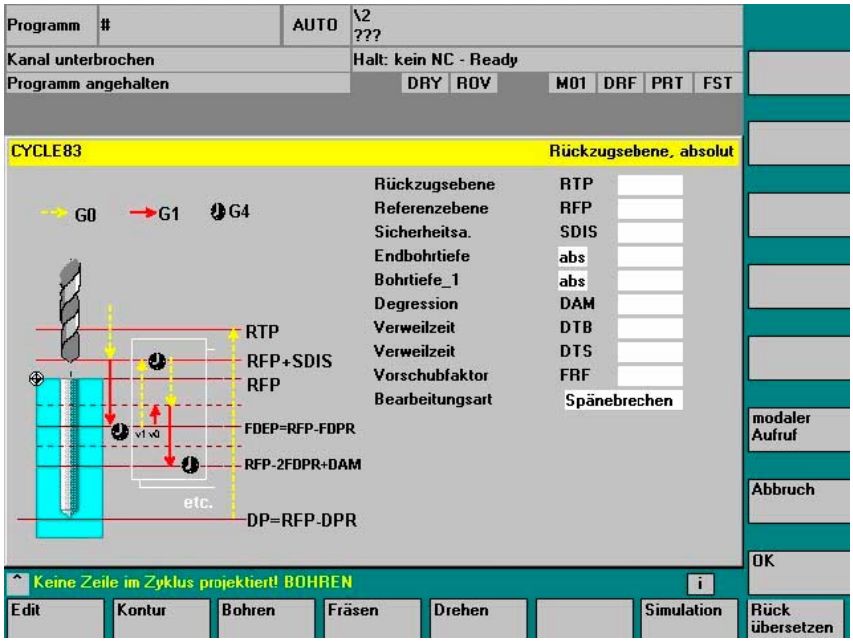

图 3-1　　预设置下的视图

3.2　循环辅助

使用

在使用循环支持时在零件程序中循环调用的前后另外建立注释行，这些注释行用于反编译。该行以;# ...作为开始。

对于直接用编辑器记录的循环调用（也称作“旧”循环），这些信息缺失。 在下列中描述，如;#-行包含的循环信息可以通过设计 INI 文件制备。
对于西门子循环，在供货范围中包含所需的设计文件。

为此可以在反编译后通过对话框支持用循环调用处理以下情况：

- 直接记录的循环调用
- 西门子循环调用
- 用户循环调用

前提条件

对于该循环，必须存在或/建立一个使用“补充操作界面”工具而创建的设计。

对于用于用户循环支持的、通过“补充操作界面”建立的对话框可以在用于此过程的对话框说明文件中设置这些信息。

确定是否要为了循环在零件程序中用 ;#-
行处理或者通过设计文件反编译，可以根据循环在设计时进行。

用这种方式可以在零件程序中以 ;#-
行选择性地建立用于直接记录的“旧”循环，用于对话框支持所需的扩展：

设计

文件 WIZARD.INI 可以保存在以下目录中。 按照此顺序查找目录：

..\user

..\oem

..\hmi_adv

..\mmc2

以下的条目参考循环软件包，例如：

```
INI_1=bohren.ini
INI_2=drehen.ini
...
INI_n=paket.ini
```

现在合并章节[MMC_CycleWizard]的信息。

在每个 paket.ini 文件中有以下形式的循环特定的条目：

```
[cycleName]     ; 循环名称作为自身段
Mname=          ; 必须设置    对话框名称
Dir=            ; 必须设置    包含对话框信息的文件的目录
Dname=          ; 必须设置    包含对话框信息的文件
Output=         ; 必须设置    OUTPUT 块的名称
Anzp=           ; 必须设置    参数（定义的变量）数目
Version=        ; 可选        循环版本，不带数据 0
Code_typ=       ; 可选        输出带有 = 0 或者不带 = 1 ;#-行
```

举例：

```
[CYCLE83]
Mname=CYC83
Dir=cst.dir
Output=bohren.ini
Anzp=17
Version=3
Code_typ=0
```

不同的循环版本

对不同版本相同名称的循环工具版本号进行区分。
程序调用前，主程序段/辅助程序段和程序段号码保留不变。

零件程序中循环调用行：

```
/1234 :44 CYCLE94(, , )
```

扩展后以字符串和 ;#-行：

```
;NCG#CYC94#\CST.DIR\DREHEN.COM#NC1#1#*NCG;*RO*;*HD*
;##########*NCG;*RO*;*HD*
/1234 :44 CYCLE94(,,"",)
;#END#*NCG;*RO*;*HD*
```

参见

反编译 (页 107)

向前/后查找(SF, SB) (页 109)

3.3 从 NC 程序激活对话框

引言

通过 HMI 高级和 HMI 内置 sl 可以显示由用户定义的对话框：
通过设计确定对话框的外观（在循环目录中修改 COM 文件）。

通过零件程序中的功能调用来调用并再次退出对话框。 HMI
软件（操作界面）此时保持不变。 不能同时在不同的通道中调用由用户定义的对话框。

命令通道

功能“从 NC 程序激活对话框”也指“命令通道”。

激活命令通道

由用户定义的对话框可以，例如用于下列场合，在进行零件程序过程之前用定义的值占用用户变量（GUD）。

- **最多 2 个通道：**

  按照标准为通道 1 和 2 激活“命令通道”。

- **多于 2 个通道：**

  对于 HMI 高级，必须激活“命令通道”（在尚未安装西门子测量循环时）。
  为此，在开机调试操作区的文件 F:\MMC2\ COMIC.NSK 中进行更改：

  在操作区“开机调试”中按下软键 “HMI” → “编辑器”选择文件 F:\MMC2\COMIC.NSK
  并插入以下文本（通道 1 和通道 2 后）：

```
REM CHANNEL
TOPIC(machineswitch) COMIC_START(COMIC001MachineSwitch"...)
[对于通道 1 和 2 对照文本]
```

  在随后的控制系统重新启动时（关闭/打开），为相应的通道激活命令通道。

HMI 高级时激活

文件 COMIC.NSK 的内容：

```
REM ----- TYPICAL COMIC START
REM CHANNEL 1
TOPIC(machineswitch) COMIC_START("COMIC001MachineSwitch
",/Channel/Configuration/mmcCmd[u1],
/Channel/Configuration/mmcCmdQuit[u1])
REM CHANNEL 2
TOPIC(machineswitch) COMIC_START("COMIC002MachineSwitch
",/Channel/Configuration/mmcCmd[u2],
/Channel/Configuration/mmcCmdQuit[u2])
```

3.3.1 指令 "MMC"的结构

编程

句法	MMC (“操作区、指令、COM 文件、对话画面名称、用户数据定义文件、图形文件、显示时间或确认变量、文本变量...”，“确认模式”)		
参数：	操作区	软键名称，通过该软键可以调用设计的用户对话框。 预设置： CYCLES，在软键 14 上显示为“循环”且可以通过按键 < ETC> 实现。	
	指令	PICTURE_ON	画面选择
		PICTURE_OFF	取消画面选择
	COM 文件	对话画面文件名（最多 8 个字符，在用户循环、制造商循环或标准循环目录中）。 在此确定对话画面的外观。 在对话画面中可以显示用户变量和/或注释文本。	
	对话框名称	单个的对话框通过对话框名称选择。	
	GUD 文件	在读/写变量时存取的用户数据定义文件。	
	图形文件 (仅 HMI 高级)	待显示的 BMP 图形的文件名	
	确认变量(仅 HMI 高级)	确认变量	确认模式“A”下
	或显示时间	显示时间	确认模式“N”下
	文本变量	COM 文件文本变量中的画面标题或者注释文本	
	确认模式	"S" 表示**同步**	通过软键“确定”确认
		"A"表示**异步**	通过设计的软键确认
		"N" 表示 **不退出**	没有确认，而是显示时间

图形的保存结构

*.bmp 文件根据分辨率分别存放在各个子目录中，它们是：

- 对于标准循环：

\CST.DIR\HLP.DIR\640.DIR	对于分辨率 640 dpi
\CST.DIR\HLP.DIR\800.DIR	对于分辨率 800 dpi
\CST.DIR\HLP.DIR\1024.DIR	对于分辨率 1024 dpi

- 对于用户循环：

\CUS.DIR\HLP.DIR\640.DIR	对于分辨率 640 dpi
\CUS.DIR\HLP.DIR\800.DIR	对于分辨率 800 dpi
\CUS.DIR\HLP.DIR\1024.DIR	对于分辨率 1024 dpi

- 对于制造商循环

\CMA.DIR\HLP.DIR\640.DIR	对于分辨率 640 dpi
\CMA.DIR\HLP.DIR\800.DIR	对于分辨率 800 dpi
\CMA.DIR\HLP.DIR\1024.DIR	对于分辨率 1024 dpi

3.3.2 MMC 指令举例

零件程序中的 MMC 指令

```
MMC("CYCLES,PICTURE_ON,T_SK.COM,画面
1,MGUD.DEF,BILD3.BMP,TEST_1,A1","S")
```

CYCLES	操作区
PICTURE_ON	选择对话框
T_SK.COM	循环目录中的文件名
画面 1	对话框的名称
MGUD.DEF	用户数据定义文件
画面 3.BMP	图形文件的名称 (仅 HMI 高级)
TEST_1	确认变量(仅 HMI 高级) 或者模式"N"下的显示时间
A1	COM 文件文本变量中的画面标题或者注释
S	应答方式： 同步

定义目录中的用户变量

```
%_N_UGUD_DEF
;$PATH=/_N_DEF_DIR
DEF CHAN REAL TEST_1
```

CHAN	通道专用的适用范围
REAL	数据类型
TEST_1	用户变量的名称

循环目录中的对话画面文件 (*.COM)

```
//C3(画面 2)
R/ 15   75 / 5 /COMMENT, %1 %2 %3/ W,RJ / TEST_1 / ...
```

R	变量类型： 实型、整型或字符串
15 75	允许范围： 15 至 75
5	用户变量的预设置
COMMENT, %1 %2 %3	注释文本，带可选的文本变量
W,RJ	存取方式： W = 读取和写入 R = 仅读取 W,RJ = 读取和写入，带注释 J = 向右对齐输入/输入栏 <无> = 向左对齐输入/输入栏
TEST_1	用户变量

文本变量

```
[文本变量]
A1 = 举例 2: 无确认情况下的 MMC 指令
```

A1	MMC 指令的参考参数
举例 2: 无确认情况下的 MMC 指令	画面标题或注释文本

注意

用大写方式写入变量名称、文本变量和循环名称。

设计用于对话框调用的软键

MMC 指令的软键占用，带异步确认模式。

[画面 3]
SK1 = 结束
SK2 = 画面 2
可以设计软键 SK0 至 SK15

3.3.3 举例 1：无确认情况下的 MMC 指令

零件程序

```
N10 MMC("CYCLES,PICTURE_ON,T_SK.COM,画面 1,GUD4.DEF,,,A1","N")
N20 TEST_1 = 1
N25 G4 F10
N30 MMC("CYCLES,PICTURE_OFF","N")
M30
```

参数：

对话框文件 (*.COM)	//C1(画面 1) (R///ANW.VAR TEST_1/W/TEST_1///)
文本变量	[文本变量] A1 =.....举例 1： 无确认情况下的 MMC 指令

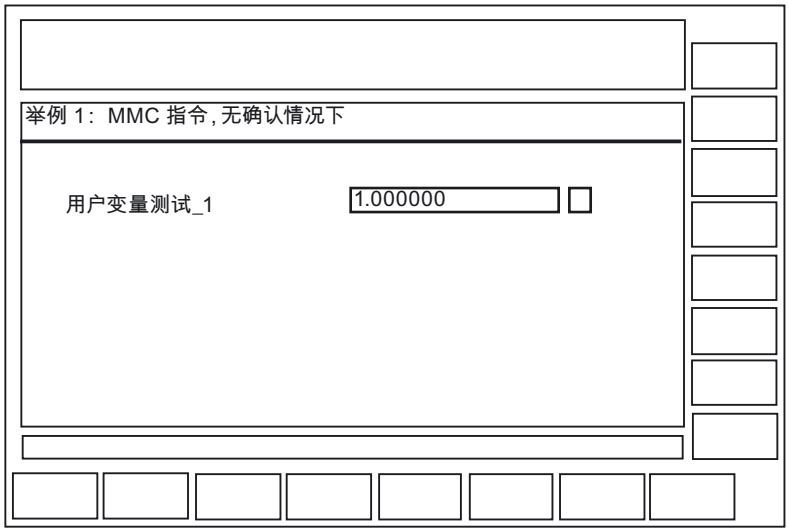

图 3-2 没有确认的举例 1

操作步骤

通过标题 A1 短时显示 GUD4.DEF 中的用户变量 TEST_1。 由零件程序段 N25 得出停留时间。

3.3.4 举例 2： 停留时间和可选的文本变量

零件程序

```
N10 MMC("CYCLES,PICTURE_ON,T_SK.COM,画面 6,GUD4.DEF,,10,T1,G1","N")
N15 G4 F15
N30 MMC("CYCLES,PICTURE_OFF","N")
M30
```

参数：

对话框文件 (*.COM) //C6(画面 6)
R///ANW.VAR TEST_1, %1/W/TEST_1///)

文本变量

```
[文本变量]
T1 = 举例 2： 停留时间和可选的文本变量 ...
G1 = 可选的文本变量
```

在这里将第 7 个参数视为用于无确认情况下模式的显示时间。 随后删除表格内容。
对话框一直保持至 PICTURE_OFF。 第 8 个参数 (T1) 为标题的文本变量。
如果没有条目存在，则显示操作区名称“循环”。 参数 9 至 23

为可选的文本变量（“G1=可选的文本变量”）。 必须在 COM 文件段[文本变量]下预设可选的文本变量。

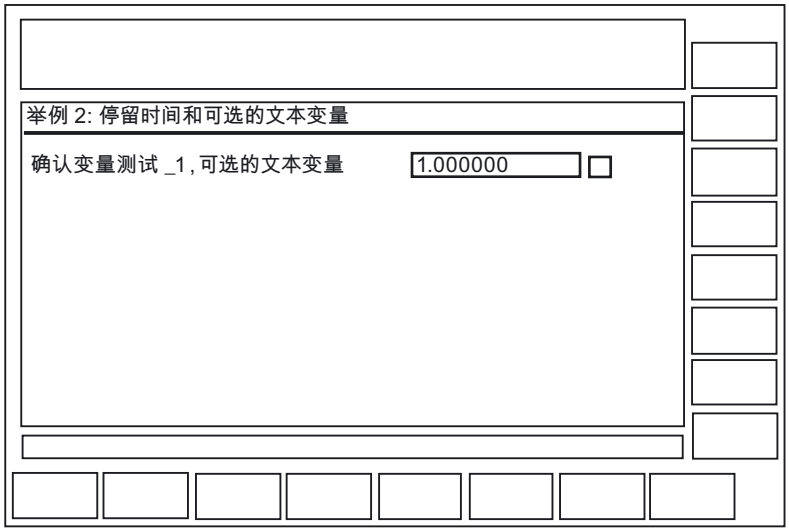

图 3-3 举例 2 停留时间

操作步骤

在该例中，COM 文件（ANW.VAR TEST_1）中的注释文本在第一个占位符（%1）的位置上，以扩展文本变量“G1=可选的文本变量”的内容。 通过调用 MMC 指令中的文本变量（第 9 至第 23 个参数）可以用这种方式“组合”信息或名称。

3.3.5 举例 3： MMC 指令，在同步确认模式下

零件程序

```
N15 MMC("CYCLES,PICTURE_ON,T_SK.COM,画面 1,GUD4.DEF,,,F1","S")
N18 STOPRE
N20 TEST_1 = 5
N25 MMC("CYCLES,PICTURE_OFF","N")
M30
```

参数：

对话框文件（*.COM） //C1(画面 1)
(R///ANW.VAR TEST_1/W/TEST_1///)

文本变量
F1 = ...举例 3: MMC 指令，在同步确认模式下...

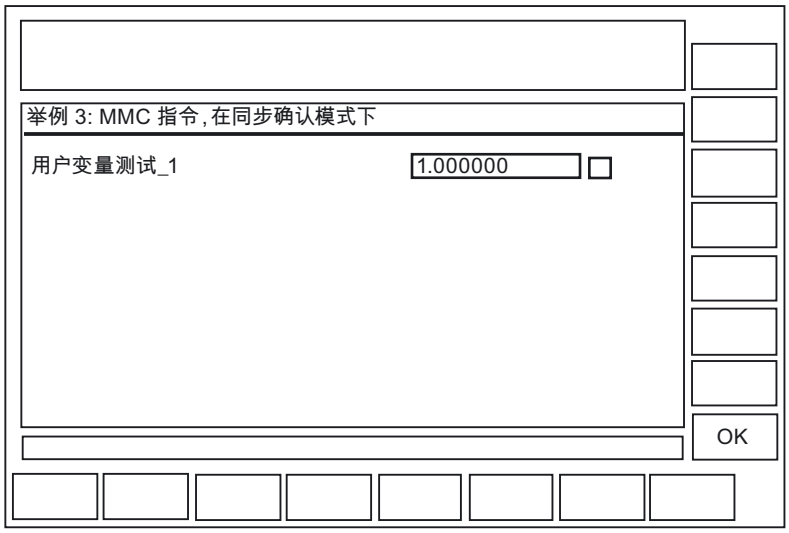

图 3-4 举例 3 同步确认模式

操作步骤

一直显示用户变量 Test_1，直至按下软键“确定”。 此时，在程序中另外用数值 5 覆盖用户变量。

在无STOPRE 的情况下，在键盘输入**之前**进行分配（ 在有 STOPRE 的情况下则在键盘输入后进行分配!）。

3.3.6 举例 4： 输入/输出栏定位

说明

通过在 COM
文件中规定位置参数，可以在显示区域内部的任意位置上显示注释栏或者输入/输出栏。

零件程序

```
N15 MMC("CYCLES,PICTURE_ON, T_SK.COM; 2,GUD4.DEF,,,C1","S")
N20 TEST_3 = 5
N30 MMC("CYCLES,PICTURE_OFF","N")
N40 M30
```

参数：

这两个参数分别由三个数值构成，它们规定了位置和栏长度。 以 Twips
为单位规定数值，一个像素等于 15 个 Twips。 栏高确定为 250 Twips。

对话框文件（*.COM）	//C2(画面 12) (R///Var.Name/R/TEST_3/6000,2800,8000/200,3000,7500)		
	/6000,2800,8000		注释栏位置
	/200,3000,7500		输入/输出栏的位置
	第一个值 = 0	→	按照预设置自动定位
	无规定	→	如同 PCU 20 定位（预设置）

数值含义：

(./6000,2800,8000/....)

6000	到画面左边缘的距离
2800	到画面上边缘的距离
8000	栏长度

操作步骤

可以为 16 个注释栏和 16 个输入或输出栏进行图形设计。 如果多于 16 个栏，可以通过滚动条控制这些栏。

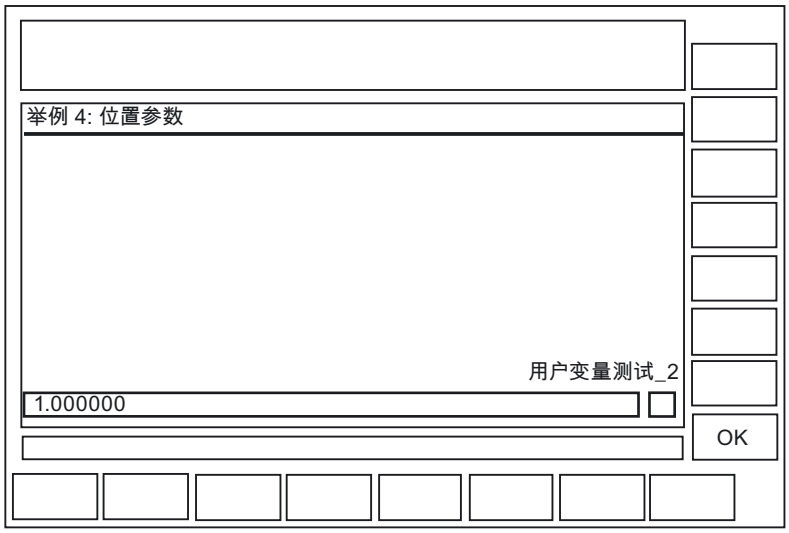

图 3-5　　举例： 4a 定位参数

为此，若要光标控制功能完全正常，必须重叠已设计的栏：

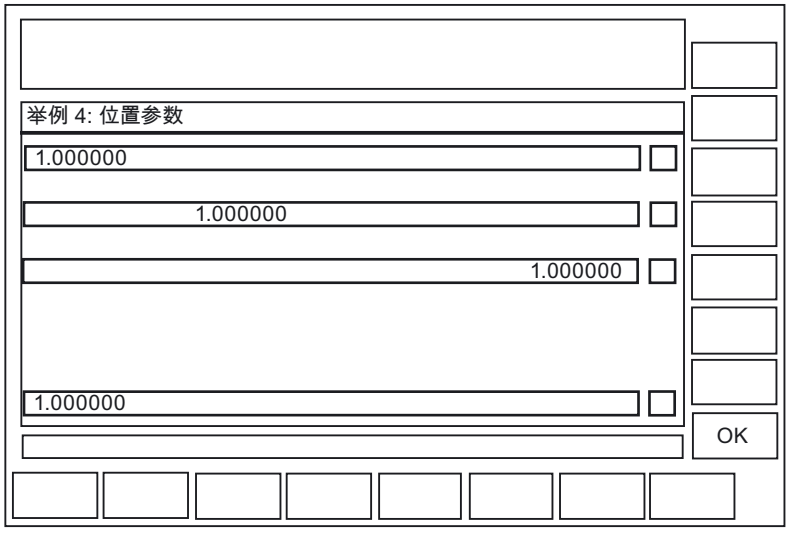

图 3-6　　举例： 4b 定位参数

3.3.7 举例 5： 在对话框画面中显示图形

说明

图形，例如用画笔建立的图形可以通过图形文件数据在对话框中显示。

图形的注释文本可以通过 COM 文件预设。 此时，注释文本也可以通过位置参数定位。

注意

图形仅可以通过在图形程序中移动自行定位。

零件程序

```
N10 MMC("CYCLES,PICTURE_ON,T_SK.COM,画面 8,GUD4.DEF,GRA.BMP,,M1","S")
N20 MMC("CYCLES,PICTURE_OFF","N")
N30 M30
```

参数：

对话框文件 (*.COM)	//C8(画面 8)
	(I/// 画面显示编号 2///4000,3000,7500)
	(I/// 用画笔建立 ///4000,3250,7500)
文本变量	M1 = ...举例 5： 画面显示.....
	例如，通过程序“画笔”建立画面。 画面尺寸： 300X500 像素，画面尺寸仅可以用图形程序更改。

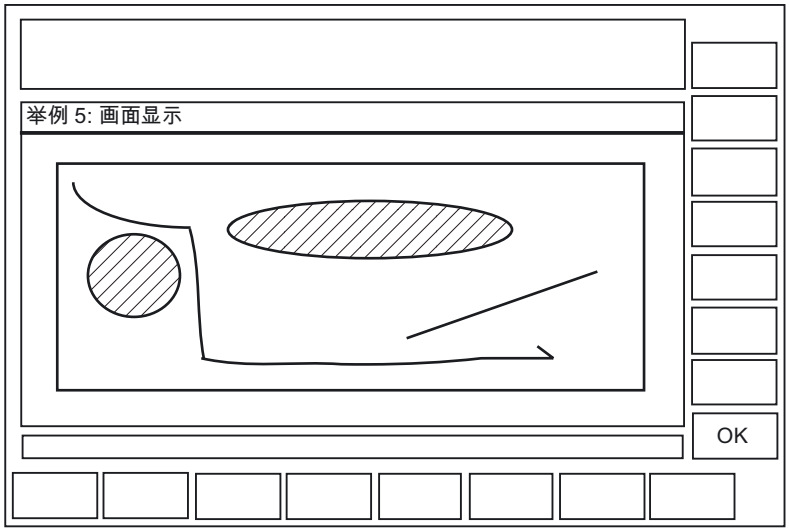

图 3-7 举例 5: 带图形

3.3.8 举例 6： 显示 BTSS 变量

说明

可以显示以下 BTSS 变量。

零件程序

```
MMC("CYCLES,PICTURE_ON,T_SK.COM,画面 7,GUD4.DEF,,TEST_1,J1","S")
```

参数：

对话框文件 (*.COM)

```
//C7(画面 7)
(R///Test_1/R/Test_1)
(I///); ( 视为空行 )
(R///轴实际值1/R/$实际值)
(R//1/R 参数 12/W/$R[12])
```

文本变量

```
J1 = ...举例 7： BTSS 变量

[BTSSVar]
$实际值=/Channel/machineaxis/actToolbasePos[u1,1]
$R[12]=/Channel/Parameter/rpa[u1,12]
```

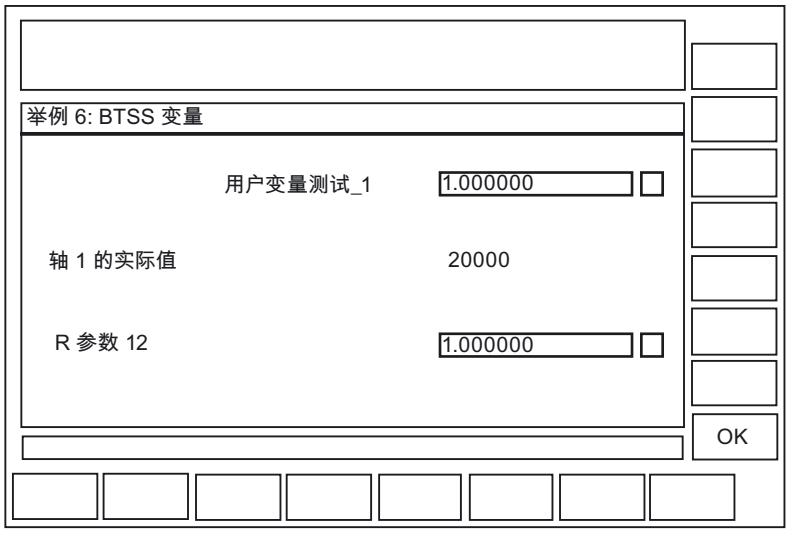

图 3-8 举例： BTSS 变量

操作步骤

在变量 TEST_1 下生成空行。

轴实际值为只读。

R12 预设置为 1。

3.3.9 举例 7： 用软键进行的异步确认模式

说明

软键可以用异步模式通过在 COM 文件中预设显示，与应答变量链接并在零件程序中评估。

零件程序

```
N10 QUIT_1 = "START"
N20 MMC("CYCLES,PICTURE_ON,T_SK.COM,画面 3,GUD4.DEF,"QUIT_1,K1","A")
N30 LABEL0:
N40 STOPRE
N50 IF MATCH (QUIT_1,"SK1") >= 0 GOTOF LABEL1
N60 IF MATCH (QUIT_1,"SK2") >= 0 GOTOF LABEL2
N70 GOTOB LABEL0
N80 LABEL2:
N90 MMC("CYCLES,PICTURE_ON,T_SK.COM,画面 1,GUD4.DEF,"N1","N")
N100 G4F10
N110 LABEL1:
N120 MMC("CYCLES,PICTURE_OFF","N")
N130 M30
```

参数：

对话框文件 (*.COM)	//C3(画面 3) (S/// ANW.VAR QUIT_1/W/QUIT_1//)
文本变量	[文本变量] K1 = .. 举例 8： MMC 指令，异步确认模式下 N1 = .. 举例 8： 画面 2
软键	[画面 3] SK1 = 结束 SK2 = 画面 2

程序结构

将应答变量定义为字符串。

字符串长度： >= 20

（值< 20 仅在内部评估， 在按下软键 SK0 ...SK15 时输入到位置 17...20 上）。

在零件程序中为字符串赋一个值并由此删除可能的旧软键信息。

在可以选用零件程序之前（与确认变量有关），程序段过程通过命令 STOPRE 停止。

```
IF Match (Quit_1," SK1") >= 0 GotoF Label1
```

；在字符串范围内查找一个字符串

如果没有按下软键，则重新循环一次。

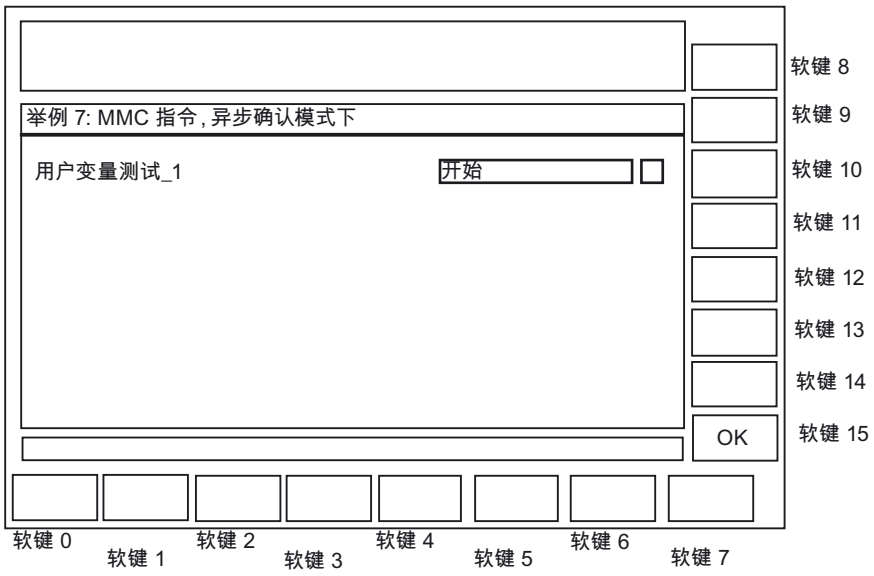

图 3-9 举例 7： 异步确认模式

操作步骤

一直显示通过异步 MMC 指令调用的画面，直至按下以下两个设计的软键：

- 用软键“结束”立即结束用户对话。
- 接着，按下软键“画面 2”持续 10 秒显示一个其它的对话框。

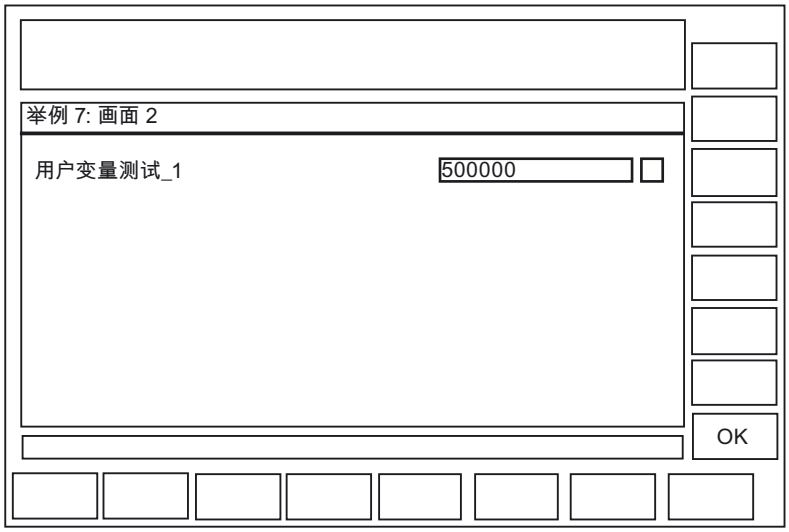

图 3-10　　举例 7： 画面 2

设计热键和 PLC 键

4

4.1 引言

概述

在本章节中说明了以下操作单元的设计方法：

- OP 010 和 OP 010C 以及带热键区的 SINUMERIK 键盘的 6 个热键以及按键 <Machine> 和 <MENU SELECT> ，其布局可以选择性修改
- 由 PLC 分析的按键，例如：机床操作面板上的按键
- 由 PLC 分析为 PLC 按键或者“虚拟按键”并且能够触发 HMI 程序内设计的操作流程的事件。

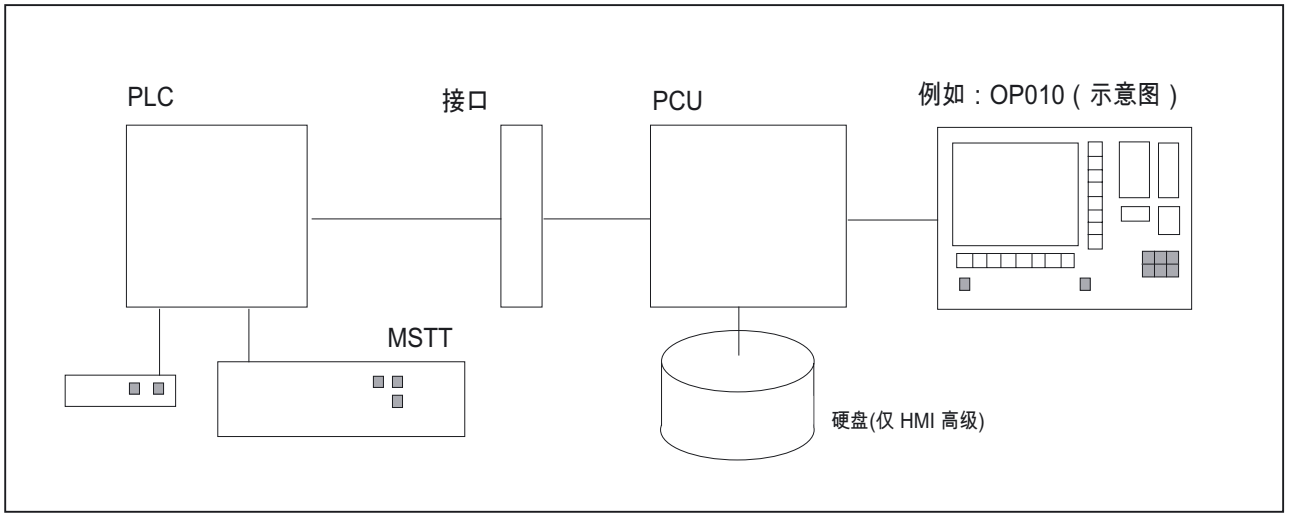

图 4-1 配置 OP010

应用

热键和按键可以用于以下任务，例如：

- 选择操作界面（例如机床，参数等等）
- 选择所选的子菜单（例如在“诊断”操作区选择报警画面）
- 触发动作（例如在“参数”操作区选择刀具列表并按下软键 HS3）
- 有针对性地选择用“补充操作界面”建立的菜单
- 选择其它画面，取决于用“补充操作界面”建立的窗口中当前的操作情况。

设计

设计通过功能“补充操作界面”进行。

- OP 的 6 个热键允许直接显示 HMI 的每个操作区域。
  以此缩短通过基本菜单执行的一般选择过程。 可以由此来改变 6 个热键的标准占用。
- PLC 和 HMI 的接口允许扩展 PLC 键的功能。 为此可以设计 HMI
  上触发的操作处理过程。 对于 PLC 的使用，提供按键号码 50 至 254。
- 按键 <Machine> 和 <MENU SELECT> 可以用与 OP 的 6
  个热键相同的方式设计（可选）。这些热键用作为 HK7 和 HK8。

4.1.1 OP 的热键

排列（预设置）

OP 的 6 个热键分 2 行排列，每行 3 个键

行 1： 标签（不带符号）		设计作为
OP 相关：		
OP10	Machine	HK1
OP10C	Machine	HK1
OP10S	Position	HK1
Program		HK2
Offset		HK3

行 2： 标签（不带符号）	设计作为
Program Manager	HK4
Alarm	HK5
Custom	HK6

可选 HK7 和 HK8

按键 <Machine> 和 <MENU SELECT> 可以如同 HK1 至 HK6 一样进行设计。
由此可以使这些键的预设失效并激活新的自身功能。

标签（不带符号）	设计作为
Machine	HK7
MENU SELECT	HK8

关于 HK7 和 HK8 的详细信息，参见章节“<M> 键和 <MENU-SELECT> 键设计为 HK7, HK8”。

注意

热键 1 和 7（<M>）键对于 OP10S（标签"Position"）无法从硬件上进行区分。
在按下按键的任意一个时总是触发热键 7。 如果已设计 HK1，则这些事件只能通过外部 MF2 键盘触发。

MF2 键盘上的热键排列

热键	OP 上的标签	MF2 的键
HK1	Position	<SHIFT+F11>
HK2	Program	<END> (NB)*
HK3	Offset	<Page Down> (NB)*
HK4	程序管理器	<Home> (NB)*
HK5	Alarm	<Page Up> (NB)*
HK6	Custom	<SHIFT+F12> 或者光标下 (NB)*
HK7	M Machine	<SHIFT+F10>
HK8	Menu Select	<F10>

*) 键位于编号程序块: <NumLock> 必须关闭。

4.1.2 供货状态下键的功能

供货状态

对于提供的系统，热键的功能布置输入在文件 KEYS.INI 中。

对于 HMI 高级文件可以存放于不同的目录下：

- user
- oem
- add_on *)
- mmc0w32 *)
- mmc2 *)
- hmi_adv *)

*) 所标记的目录保留用于西门子。

文件按照此处规定的顺序处理。 最高级目录的条目使得较低一级目录的条目失效。

供货时 HK1 至 HK6 的分配如下存放在标准系统的目录**mmc2**的文件 KEYS.INI 中：

按键		功能
HK1	Position	“加工”操作区，最后的画面
HK2	Program	“程序”操作区，最后的画面
HK3	Offset	“参数”操作区，最后的画面

按键		功能
HK4	Program Manager	编程基本画面
HK5	Alarm	“诊断”操作区，报警画面
HK6	Custom	“默认定制”操作区，最后的画面 （用户设计的操作界面）

对于 ShopMill/ShopTurn，按键功能如下：

按键		功能
HK1	Position	“加工”操作区，最后的画面
HK2	Program	程序编辑器，最后的状态
HK3	Offset	“参数”操作区，最后的状态
HK4	Program Manager	程序目录，最后的状态
HK5	Alarm	“诊断”操作区，报警画面
HK6	Custom	“默认定制”操作区，最后的画面 （用户设计的操作界面）

4.2 设计

4.2.1 设计概述

概述

下图以图表方式描述了 OP 设计的热键和 PLC
按键之间的关系和根据“补充操作界面”进行设计的步骤。

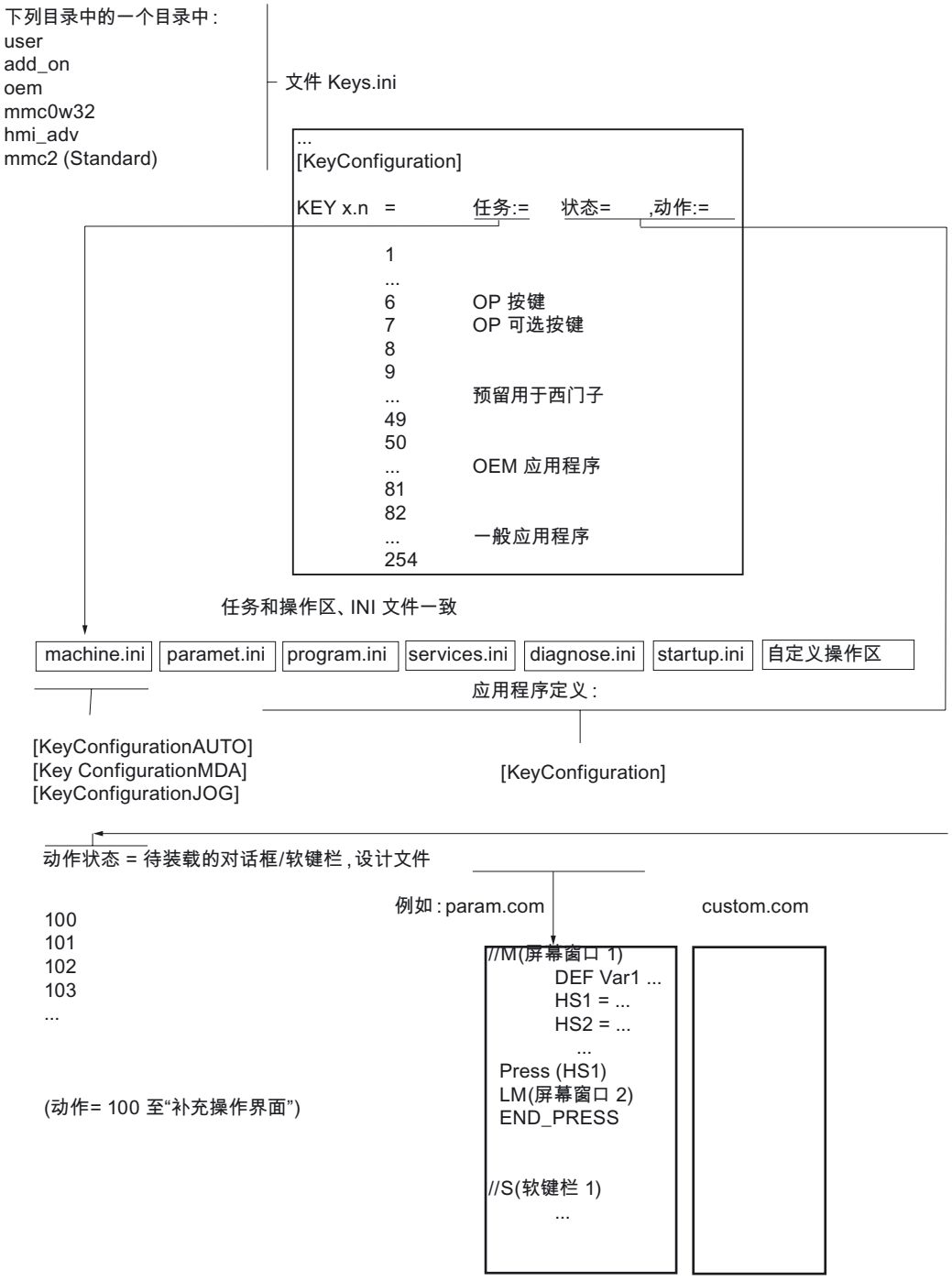
下列目录中的一个目录中：
user
add_on
oem
mmc0w32
hmi_adv
mmc2 (Standard)
文件 Keys.ini
...
[KeyConfiguration]
KEY x.n = 任务:= 状态= ,动作:=
1
...
6 OP 按键
7 OP 可选按键
8
9
... 预留用于西门子
49
50
... OEM 应用程序
81
82
... 一般应用程序
254
任务和操作区、INI 文件一致
machine.ini
paramet.ini
program.ini
services.ini
diagnose.ini
startup.ini
自定义操作区
应用程序定义：
[KeyConfigurationAUTO]
[Key ConfigurationMDA]
[KeyConfigurationJOG]
[KeyConfiguration]
动作状态 = 待装载的对话框/软键栏，设计文件
例如：param.com
custom.com
100
101
102
103
...
(动作= 100 至“补充操作界面”)
//M(屏幕窗口 1)
DEF Var1 ...
HS1 = ...
HS2 = ...
...
Press (HS1)
LM(屏幕窗口 2)
END_PRESS
//S(软键栏 1)
...

图 4-2 设计概述

4.2.2 在文件"IF.INI"中进行设计

概述

每次在上述的目录中标示配置文件 KEYS.INI 都将引起程序段 [键配置] 中的热键反应。
每个条目（行）定义某个热键的某个操作（多次操作）上的系统反应，下面称作热键事件。

配置文件 KEYS.INI

配置文件 KEYS.INI 获得一个自身的程序段，该程序段用于分配 PLC 按键的 ini 文件。
没有该条目就无法识别 ini 文件。

在标准情况下存在下列条目：

```
[HMI_INI_FILES]
任务 0 = machine.ini
任务 1 = paramet.ini
任务 2 = program.ini
任务 3 = services.ini
任务 4 = diagnose.ini
任务 5 = startup.ini
任务 6 = shopmill.ini
（或者： 任务 6 = shopturn.ini je，视系统而定, 必须设置！）
任务 11=custom.ini
```

注意

为了使 KEYS.INI 中的更改有效，必须重新启动。

条目形式

条目的形式在“热键事件”下进行定义。
为了可以重复操作那里所使用的相同热键，首先描述多重操作。

多重操作

功能扩展确定热键多重操作时的过程： 在操作热键时由于文件 KEYS.INI 中增加的重复次数“n”，可以分配一个自主的组合：任务/状态/动作。
为此每次按下按键都可以进行状态切换和画面选择以及软键选择。

在多重操作时切换到另一个操作区将删除当前的热键。 为此也适用于未操作的热键。
此时例如实现切换到另一个操作区（操作区菜单等等）与此无关。
此外每次按下功能键（例如软键，操作区切换键，通道切换键等等）时同样可能导致删除当前状态。

在当前的对话框中进行栏输入不会中断重复计划。
在当前的对话框中按下软键，通过用来调用当前对话框的按键中断重复计划。

注意

多重操作可以用于 HK1 – HK8 以及备用的西门子按键 HK9 – HK49。

软键多重操作举例

在第一次操作热键时激活相关的操作区并可能触发该操作区中的一个状态/动作（条目 1）。
在再次操作该热键时，接着执行该热键接下来的条目，操作区切换不再进行。
如果热键的所有设计的条目已运行，则重复该循环。 条目总是以正向（从 0 至 9）运行。
不进行负向的运行。 在设计时不允许存在“事件缺空”（缺少的条目）。
一个缺空将被视作链的结束并在下次按下按键时立即再次在按键 x.0 处开始处理。

设计：

```
[KeyConfiguration]
按键 1.0 = 任务:=0, 状态:=10,          ; 热键 1，第 1 次操作
动作:=2
按键 1.1 = 状态:=10, 动作:=3           ; 热键 1，第 2 次操作
按键 1.2 = 动作:=4                     ; 热键 1，第 3 次操作
```

KEYS.INI 中用来选择操作区“加工”的条目：

```
按键 1.0 = 任务:=0                     ; 第 1 次按下按键的最后画面
按键 1.1 = 任务:=0, 动作:=0            ; 第 2 次按下按键的开始画面
按键 1.2 = 状态:=0, 动作:=3            ; 在第 3 次按下按键和选择第 3
                                       个软键时的开始画面
按键 1.3 = 状态:=0, 动作:=100          在第 4 次按下按键并选择通过
                                       machine.ini 中<= 100
                                       设计的“补充操作界面”时的开始画面
```

针对动作值的软键分配

水平软键 (SKHi)	动作
SKH1	0
SKH2	1
SKH3	2
SKH4	3
SKH5	4
SKH6	5
SKH7	6
SKH8	7

垂直软键(SKVi)	动作
SKV1	8
SKV2	9
SKV3	10
SKV4	11
SKV5	12
SKV6	13
SKV7	14
SKV8	15

按键	动作
<RECALL>	16
<ETC> (仅 HMI 内置 sl)	17

4.2.3 编程热键事件

说明

热键事件由最多四个属性组成。 关键属性
KEY（按键）识别事件，并且必须总是位于第一位。
所有其它的属性可选，但是必须至少再规定一个属性。
该属性的顺序可以在行中的任意位置。

编程热键事件

句法： 按键 x.n = Task:=任务, State:=状态, Action:=动作

参数：

Key (热键)	值 x.n 包含热键号**x**和一个直接的事件**n**，该事件应该为第 n 次操作（对于多次操作）。 热键号的取值范围为 1 - 254。 事件 n 的取值范围是 0 - 9（10 次按键操作）。 热键 1 – 8 分配 OP 的按键。 热键 9 – 49 预留用于西门子。 另外还有 PLC 专用的按键 50 – 254。这些按键不是真正的热键，而是用于通过 PLC 选择画面（虚拟键）。 按键 50 – 254 也可以设计为不带任务。 在这种情况下事件总是指定当前的任务。
Task(操作区)	各操作区（任务）热键的分配通过一个任务号实现。 任务号同时定义水平软键，在该水平软键上操作区挂在操作区切换菜单中。 它也形成一个任务号和附属软键之间的关联。 在 HMI 系统中备有三个（对于 HMI 内置 sl 为两个）软键条用于操作区切换。 为此规定为下列分配方式： 在按下操作区切换键后选择软键栏 1。 SK1 → 任务 0 ... SK8 → 任务 7 接下来按下键<ETC>切换到软键栏 2 SK1 → 任务 8 ... SK8 → 任务 15 然后按下键<ETC>切换到软键栏 3（ 只适用于 HMI 高级）： SK1 → 任务 16 ... SK8 → 任务 23

按下键<ETC>后再次切换到软键栏 1。 HMI 高级的任务号值在 0 至 23 之间；HMI 内置 sl 的任务号值在 0 至 15 之间。如果没有规定任务，则事件对于当前的任务有效（当前的操作区）。

可以获得任务号：

- 对于 HMI 高级，从文件 REGIE.INI 获得 。
- 对于 HMI 内置sl，通过工具**SCK**（软件配置套件）。

可以通过设计在软键和操作区（任务）之间进行用途特定的分配：

- HMI 高级：
  文件 REGIE.INI 的段落[TaskConfiguration]
- HMI 内置 sl： SCK “修改配置”

状态

通过属性“状态”可以在任务中选择某个画面。
值范围取决于相应的应用程序，最多可以在 0 至 65534 之间（65535 由系统占用）。

动作

通过属性“动作”可以在任务内选择某个软键。
只有当处于一个定义的状态中，例如基本菜单，该属性才有意义。

取值范围与对应的应用程序有关，最多在 0 到 17 之间。属性动作这里是比较特殊的，以≥ 100 的值通过“补充操作界面”触发画面选择。

值范围 0 - 99 中的动作在 ShopMill/ShopTurn 上不执行。

另见

章节“可选状态的列表”

4.2.4 扩展及特殊情况

设计 M 键和“Menu Select”键为 HK7，HK8

可以如同热键一样有选择地设计 <M> 键 (Machine) 和 <MENU-SELECT> 键。
由此，这些键失去了其最初的意义和功能。 通过配置文件 KEYS.INI 确定新的功能。
如果在按键的配置文件中有一个定义，但是存放没有反应，则按键无效。

热键的按键布置：

		MF2 键盘	热键
	按键 <Machine>	SHIFT+F10	HK7
	按键 <MENU-SELECT>	F10	HK8

如果对于热键 7 或者热键 8，在配置文件 keys.ini
中不存在**条目**，则该按键没有映射到热键上，而只包含其原来的功能（兼容模式）。
一个没有反应的按键定义通过属性 <empty> 规定。

举例：

```
[KeyConfiguration]
按键 7.0 = 任务:= 3, 状态 := 10    ; 激活 M 键到热键 7
                                     的映射 并确定新的按键反应
                                   ; 激活 <MENU SELECT> 键（F10）到热键
                                     8 的映射 ，按键无效
Key8.0 = <empty>                   ; 未指定反应
```

HMI 高级的扩展

机床制造商可以用自身的设置覆盖目录 mmc2 的 KEYS.INI 中的条目。
为此，机床制造商可在以前执行时查找过的目录 \user和 \oem 的其中一个里面输入其协议。
只需要输入其偏差之处，而无需整个定义语句。

HMI 高级特殊情况下的动作:

<ETC> 键，动作 17 无效。

HMI 内置 sl 上的特殊情况

特殊情况下的任务、状态和动作：

- 用于任务的值范围从 0 至 15。
- HMI 内置 sl 选择最后一次任务切换前已激活的画面组合（状态未设计）。
- 状态的值范围限制为 0。 只能选择操作区的基本菜单。
- 动作的值范围为从 0 至 17：
  - 水平软键 1 到 8
  - 垂直软键 1 到 8
  - 回调
  - ETC

4.2.5 PLC 按键的扩展

概述

对于 PLC 按键，即可以设计一个任务切换，也可以设计一个状态切换。

动作的值域：

标准应用程序	0 – 17
“补充操作界面”应用程序	≥100

PLC 专用键的编号为 50 - 254;其中 50 - 81 号为 OEM 应用程序预留。

举例：

- 这里指示的配置不进行任务切换和状态切换，然而保留在当前的任务中和当前的画面中，并传输一个动作 100。

```
[KeyConfiguration]
HK50.0 = 动作:=100
```

- 同样这里也可以设计一个任务切换和状态切换。

```
HK50.0 = 任务:=1, 状态:=10, 动作:=100
```

4.3 PLC 接口

4.3.1 接口结构

概述

在 PLC 接口中规定范围 DB19.Byte10 用于选择按键。 这里 PLC 可以直接规定 50 至 254 之间的一个按键。

（键 1 至 49 预留用于西门子，键 50 到 254 专门用于 PLC 按键。）

注意

对于 M:N 运行方式，第 2 个 HMI 接口的范围是 DB19.Byte60。

确认

通过 HMI 系统确认分两步进行：

- 在第一步中由 HMI 软件将控制信息 255 传输到 DB19.Byte10 中。
- 接着在第二步中删除 DB19.Byte10 中向 PLC 的自身确认信息。

这是必要的，为此尽管 HMI 和 PLC 不同步，可以立即接连两次由 HMI 识别相同的按键代码。 通过该定义的虚拟按键按下 PLC 可以清楚识别每个按键顺序。 控制信息对于 PLC 程序没有意义（透明）并且不允许改变。

下一次按键规定

如果在传输字节中为 0，则 PLC 程序可以给出一个新的按键。 为此同时在 HMI 系统中处理当前按键的请求。
请求切换到相应的任务或者在当前的任务中触发一条状态/动作命令。

如果无法进行任务切换，则在操作界面上给出一条相应的信息。

对 PLC 程序的请求

只有当 HMI 系统确认了以前的请求（在接口中为 0），才允许处理一条新的请求。 当 PLC 程序从一个机床控制面板的按键或者另一个信息源导出按键，
接着必须有足够的按键中间缓存，以此在快速操作时才不会丢失按键按下信息。

从 PLC 选择对话框

在 PLC 和 HMI 内置 sl 之间存在一个接口用于选择对话框。 在该 PLC 对话框中可以提供与由软键选择的对话框中相同的描述方式和功能。

4.3.2 PLC 画面选择说明

接口说明

接口包含画面号、PLC 传到 HMI 的控制位和 HMI 传到 PLC 的控制位。 该接口在 DB19 中需要总共 8 个字节， HMI 内置 sl 各占用 4 个字节。

由于在“多个 NCU 上的多个操作面板 ”范围内每个 NCU 最多可以同时响应两个 HMI 内置 sl，因此也存在两个这样的接口。

文献： /FB2/, “多个 NCU 上的多个操作面板”(B3)

接口结构

HMI 内置 sl 和 PLC 之间的接口使用下列数据：

HMI 1:	DB19.DBW28:	画面号
	DB19.DBB30:	控制位 PLC → HMI, PLC 字节
	DB19.DBB31:	控制位 HMI → PLC , HMI 字节
HMI 2:	DB19.DBW78:	画面号
	DB19.DBB80:	控制位 PLC → HMI, PLC 字节
	DB19.DBB81:	控制位 HMI → PLC , HMI 字节

PLC 字节	位 0	画面选择
	位 1	画面选择
HMI 字节	位 0	接收画面选择或者接收取消画面选择
	位 1	选择画面或者取消选择
	位 2	已选择画面
	位 3	已取消选择画面
	位 4	错误，画面选择无法进行
	位 7	无效位

对于从 PLC 要传输的**画面号**使用两个字节，每一个字节（PLC 字节）和 HMI（HMI 字节）用于画面选择的**协调**。

运行接口

接口由制造商的 PLC 程序运行，包含以下功能（DB 19 摘录，第一个 HMI 接口）：

	PLC → HMI	选择	撤销选择	HMI → PLC	选择	撤销选择
DBW 28	画面号	(1)				
位	DBB 30			DBB 31		
0	画面选择	1 (2) 0 (4)		已接收选择 / 取消选择	1 (3) 0 (6)	1 (2) 0 (3)
1	取消画面选择		1 (1) 0 (4)	已选择画面 取消选择画面	0 (3) 1 (5)	0 (3)
2				已选择画面	0 (3) 1 (7)	0 (3)
3				已取消选择画面	0 (3)	0 (2) 1 (3)
4				错误，画面选择无法进行	0	0 (2)
5				–		
6				–		
7				无效	0	

画面选择

在上述表格中各个步骤的对象通过括号中的数字（步骤号）指定。

- PLC 将画面号输入在号码字中。
- 当 HMI 字节中位 0 和位 7 为零时，PLC 在 PLC 字节中设定位 0 用于画面选择。
- HMI 内置 sl 通过设置 HMI 字节中的位 0 确认 PLC 接收画面选择。 在 HMI 字节中同时将位 3 和位 4 置零。
- PLC 复位 PLC 字节中的位 0。
- HMI 内置 sl 通过 HMI 字节中的位 1 设定确认 PLC 启动编译器。
- HMI 复位 HMI 内置 sl 字节中的位 0。
- 如果显示相应的画面，在 HMI 字节中设定位 2。

取消画面选择

在上述表格中各个步骤的对象通过括号中的数字（步骤号）指定。

- 如果在 HMI 字节中设定位 1 和位 2，并复位位 7 ，则 PLC 设定 PLC 字节的位 1 用于取消画面选择。
- HMI 内置 sl 通过设置 HMI 字节中的位 0 确认 PLC 接收取消画面选择。 在 HMI 字节中同时将位 3 和位 4 置零。
- 当编译器结束时，HMI 内置 sl 通过设置 HMI 字节的位 3 和复位位 0、位 1 和位 2 来确认 PLC 取消画面选择。
- PLC 复位 PLC 字节中的位 1。
- HMI 内置 sl 通过 PLC 或者通过退出命令在最后选择的 HMI 画面中切换到画面撤销选择。

画面选择时出错

如果编译器在 20 秒后没有报告或者所需的画面无法显示，则设置 HMI 字节的位 4。

通过 HMI 字节中的位 7 显示 HMI 的 PLC 状态，在该状态中无法进行 PLC 画面选择，例如在 HMI 内置 sl 中标准操作界面和 ShopMill/ShopTurn 操作界面之间转换时。

操作区域

对于**HMI 高级**，画面显示在一个自身的操作区中，类似于测量循环画面。
该操作区可以手动通过第六个软键（PLC 画面）在第二个操作区条上选择。
如果选择操作区时没有 PLC 画面激活，则在标题栏中出现文本“目前没有 PLC 画面激活”。
操作区的手动选择或撤销选择由 HMI 字节中的位 2 通知 PLC。

通过自身的操作区也可以在选择画面后通过 PLC 切换到另一个操作区。

对于**HMI 内置 sl**，由 PLC 选择的画面无法通过一个操作区手动选择或者撤销选择。
即使在切换操作范围之后，PLC 画面也始终在前台。

4.3.3 设计对话框选择

说明

在 HMI 启动时通过文件 COMMON.COM 中的段 [PLC_SELECT] 激活接口。 在文件 COMMON.COM 中在段 [PLC_SELECT] 下给设计的画面分配画面号码。

设计

句法：	PC*i* = *画面名称, 文件, 注释*	
说明：	将画面号分配给设计的画面	
参数：	i	接口中的画面号
	画面名称	对话框名称
	文件	设计对话框的文件
	注释	对话框的注释

举例

```
[PLC_SELECT]
PC1= CYC82, bohren.com          ;    分配行
PC2= CYKLE90, gewfraes.com
PC3= ...
```

参见

COMMON.COM 的查找方案 (页 189)

4.4 选择对话框/软键栏

4.4.1 分配 INI 文件到操作区域

概述

HMI 软件根据动作值 ≥ 100 在“补充操作界面”过程中分配。

如果动作 ≥ 100，则在**INI 文件**中设计在哪些状态应显示哪些软键栏或者哪些对话框。

HMI 高级/ HMI 内置 sl

在下列 INI 文件和段中可以在“补充操作界面”中进行跳转设计：

操作区	文件	段
Maschine	machine.ini	[KeyConfigurationAuto] [KeyConfigurationMDA] [KeyConfigurationJOG]
Parameter	paramet.ini	[KeyConfiguration]
Program	program.ini	[KeyConfiguration]
Service	services.ini	[KeyConfiguration]
Diagnose	diagnose.ini	[KeyConfiguration]
Startup	startup.ini	[KeyConfiguration]
Custom	custom.ini	[KeyConfiguration]

NCU 中的 ShopMill / NCU 中的 ShopTurn

对于 NCU 上的 ShopMill 和 ShopTurn，INI 文件相应地称为 SHOPMILL.INI 和 SHOPTURN.INI。文件中包含了下列可以设计功能“补充操作界面”的章节。

```
[MachineManual]
...
[MachineAutomatic]
...
[Programmanager]
...
[Program]
...
[MessagesAlarms]
...
[ToolsZeroOffset]
...
[MachineMDI]
```

4.4.2 设计功能“补充操作界面”

说明

对于每个动作（≥ 100
），都可以设计一个或者多个状态（根据状态列表）和设计应触发哪个“补充操作界面”功能。
如果没有在 INI 文件中设计，则不会显示对话框或者软键栏。

编程任务.状态

句法：	任务.状态 = 要装载的对话框/软键栏，设计文件	
参数：	动作	动作是在 KEYS.INI 中设计的动作 ≥ 100。
	状态	应用程序当前处于的状态
	设计文件	用于存放设计的文件。
	LS/LM	装载软键/对话框的“补充操作界面”命令

举例

```
[KeyConfiguration]
100.10=LS("软键 1","param.com")
100.30=LM("屏幕窗口 1","param.com")
101.10=LS("软键 2","param.com")
101.30=LM("屏幕窗口 2","param.com")
102=LM("屏幕窗口 2","param1.com")
```

说明：

- 例如第一行表示：

  如果触发一个动作 100，当前在状态（ 画面 ） 10 中，则显示文件 param.com 中设计的名称为软键 1 的软键栏。

- 例如最后一行表示：

  如果触发动作 102，则在当前的对话框中调用名称为屏幕窗口 2 的对话框，该对话框在文件 param1.com 中设计。

4.5 可选状态的列表

4.5.1 HMI 高级上的可选状态

概述

一般情况下，适用于 HMI 高级的所有任务：

没有设计：	当前的状态保留
设计为 0：	操作区的基本状态跳转。

操作区域“加工”

在“加工”区，状态总是取决于加工状态（AUTO、MDA、JOG、REF）。 通过 PLC 按键可以直接选择：

- 每个运行方式的基本画面

状态	BAG	机床功能	显示
0	JOG	REF	“JOG/REF”基本画面
0	JOG	没有	“JOG”基本画面
0	JOG	REPOS	“REPOS”基本画面
0	MDA	没有	“MDA”基本画面
0	MDA	TEACH	“MDA/示教”基本画面
0	MDA	REF	“MDA/REF”基本画面
0	AUTO	没有	“自动运行”基本画面

- 对应的实际值大画面（垂直软键 6）

状态	BAG	机床功能	显示
10	JOG	没有	缩放 实际值 JOG
20	MDA	没有	缩放 实际值 MDA
30	AUTO	没有	缩放 实际值 Auto

- 还可能在 JOG / MDA

  可以通过 PLC 在 WCS（工件坐标系）和 MCS（机床坐标系）之间切换。

状态	BAG	机床功能	显示
60	JOG	没有	手轮 选择 JOG
70	JOG	没有	增量 选择 JOG
80	MDA	没有	手轮 选择 MDA

操作区“参数”

这里只能在当前的画面中运行。

操作区“程序”

状态	功能	
10	数据选择	
20	程序管理	
70	记录	

操作区“通讯”

状态	功能	
10	读入数据	
20	读出数据	
40	管理数据	
60	数据选择	
80	外部驱动器	
90	批量开机调试	
100	升级	

操作区“诊断”

诊断的基本画面为报警一览。

在该状态下可以通过水平软键到达其它状态：

状态	功能	
10	报警	
20	显示信息	
30	报警记录	
40	服务显示	
50	PLC 状态	

信息、报警记录和服务显示总是可以到达的。

操作区“开机调试”

开机调试中的基本画面为现有的 NC 轴和驱动一览。

状态	功能	
0	NC 轴和驱动	
10	机床数据	
40	PLC 状态	
50	优化/测试	(高于 V7.1)
60	HMI	

操作区“定制”

参见章节“操作区‘定制’”

4.5.2 HMI 内置 sl 上可选的状态

概述

在 HMI 内置 sl 中只有下列状态的设计方法：

一种设计：	当前的状态保留
设计为 0：	操作区的基本状态跳转。

4.5.3 NCU 上 ShopMill 的可选状态

手动加工

图例说明：

*	如果有选项（显示机床数据已设置）
用户屏幕窗口	标记的功能可以通过“补充操作界面”设计。 如果存在这样的设计，则将其激活，否则出现相应的 ShopMill 标准画面。

状态	功能
19	基本画面
2	T, S, M, ...
30	工件零点
5	工件零点- 调整边沿
7	工件零点-/用户屏幕窗口
31	工件零点-校正边沿/用户屏幕窗口
32	工件零点-2 边间距/用户屏幕窗口
33	工件零点-直角
8	工件零点-任意角/用户屏幕窗口
34	工件零点-矩形腔
9	工件零点-钻孔 1/用户屏幕窗口
35	工件零点-钻孔 2
36	工件零点-钻孔 3
37	工件零点-钻孔 4
38	工件零点-矩形轴颈
10	工件零点-圆形轴颈 1/用户屏幕窗口
39	工件零点-圆形轴颈 2
40	工件零点-圆形轴颈 3
41	工件零点-圆形轴颈 4
42	工件零点- 调整平面*
11	工件零点-校正按键长度*/用户屏幕窗口
12	工件零点-校正按键半径*
50	测量刀具
16	测量刀具- 手动长度/用户屏幕窗口
17	测量刀具-直径/用户屏幕窗口
13	测量刀具- 自动长度*/用户屏幕窗口
14	测量刀具- 自动直径*/用户屏幕窗口
51	测量刀具-/用户屏幕窗口
15	测量刀具- 校正测量头*/用户屏幕窗口
52	测量刀具- 校正固定点*/用户屏幕窗口
60	摆动*
4	定位

状态	功能
18	平面铣削
1	ShopMill 设置
90	-/用户屏幕窗口

MDA

状态	功能
20	MDA

自动加工

状态	功能
200	基本画面
210	程序控制
220	程序段查找
230	-/用户屏幕窗口
242	绘制- 顶视图*
243	绘制- 3 面视图*
244	绘制- 立体模型*
250	设置

Program Manager

状态	功能
300	NC 目录
310	零件程序*
320	子程序*
330	用户目录 1*
340	用户目录 2*
350	用户目录 3*
360	用户目录 4*
380	标准循环*
381	制造商循环*
382	用户循环*
383	用户目录 5*
384	用户目录 6*
385	用户目录 7*
386	用户目录 8*

Program

状态	功能
400	工作计划 / G 代码编辑器
412	模拟- 顶视图*
413	模拟- 3 面视图*
414	模拟–立体模型*

报警信息

状态	功能
500	显示信息
510	-/用户屏幕窗口
520	-/用户屏幕窗口

刀具零点偏移

状态	功能
600	刀具表
610	刀具磨损
620	用户刀具列表*
630	刀具库
640	零点偏移
650	R 参数
660	-/用户屏幕窗口
680	用户数据
690	机床数据

4.5.4 NCU 上 ShopTurn 的可选状态

手动加工 (无选项 “手动加工”)

图例说明：

*	如果有选项（显示机床数据已设置）
用户屏幕窗口	标记的功能可以通过“补充操作界面”设计。 如果存在这样的设计，则将其激活，否则出现相应的 ShopMill 标准画面。

状态	功能
19	基本画面
2	T, S, M, ...
30	工件零点
31	工件零点-/用户屏幕窗口
34	工件零点-/用户屏幕窗口
35	工件零点-/用户屏幕窗口
36	工件零点-/用户屏幕窗口
37	工件零点-/用户屏幕窗口
38	工件零点-/用户屏幕窗口
40	工件零点-/用户屏幕窗口
5	工件零点-测量边沿 Z
50	测量刀具
51	测量刀具- 手动长度 X /用户屏幕窗口
52	测量刀具- 手动长度 Z /用户屏幕窗口
53	测量刀具-放大镜*/用户屏幕窗口
54	测量刀具-/用户屏幕窗口
55	测量刀具-/用户屏幕窗口
56	测量刀具- 校正测量头*/用户屏幕窗口
57	测量刀具-/用户屏幕窗口
58	测量刀具-自动 Z*
59	测量刀具-自动 X*
4	Position
18	平面铣削*
80	切削*
90	-/用户屏幕窗口（尾座）
1	ShopTurn 设置

手动加工（带选项 “手动机床”）

状态	功能
19	基本画面
50	测量刀具
51	测量刀具- 手动长度 X /用户屏幕窗口
52	测量刀具- 手动长度 Z /用户屏幕窗口
53	测量刀具-放大镜*/用户屏幕窗口
54	测量刀具-/用户屏幕窗口
55	测量刀具-/用户屏幕窗口
56	测量刀具- 校正测量头*/用户屏幕窗口
57	测量刀具-/用户屏幕窗口
58	测量刀具-自动 Z*
59	测量刀具-自动 X*
1300	直线
1400	钻削
1410	钻孔-中心
1420	钻孔-螺纹中心
1433	钻削–定中心*
1434	钻削–钻削*
1435	钻削–铰孔*
1440	钻削–深钻孔*
1453	钻削–螺纹钻削*
1454	钻削–螺纹铣削*
1500	车削
1513	车削 – 切削 1
1514	车削 – 切削 2
1515	车削 – 切削 3
1523	车削 – 切槽 1
1524	车削 – 切槽 2
1525	车削 – 切槽 3
1533	车削- E 型退刀槽
1534	车削- F 型退刀槽
1535	车削- 螺纹 DIN 退刀槽
1536	车削- 螺纹退刀槽
1543	车削- 螺纹长度
1544	车削- 螺纹锥
1545	车削- 螺纹端面
1550	车削- 断削
1600	铣削*
1613	铣削-矩形腔*
1614	铣削-圆形腔*
1623	铣削-矩形轴颈*

状态	功能
1624	铣削-圆形轴颈*
1633	铣削-长形腔*
1634	铣削-圆形槽*
1640	铣削-多边形*
1670	铣削-雕刻*
1730	模拟- 3 窗口视图*
1740	模拟- 侧视图*
1750	模拟- 端视图*
90	-/用户屏幕窗口（尾座）
1	ShopTurn 设置

MDA

状态	功能
20	MDA

自动加工

状态	功能
200	当前的程序段显示
210	程序控制
220	程序段查找
230	-/用户屏幕窗口
242	绘制- 3 窗口视图*
243	绘制- 侧视图*
244	绘制- 端视图*
250	设置

Program Manager

状态	功能
300	NC 目录
310	零件程序*
320	子程序*
330	用户目录 1*
340	用户目录 2*
350	用户目录 3*
360	用户目录 4*
380	标准循环*
381	制造商循环*
382	用户循环*

状态	功能
383	用户目录 5*
384	用户目录 6*
385	用户目录 7*
386	用户目录 8*

Program

状态	功能
400	工作计划 / G 代码编辑器
412	模拟- 3 窗口视图*
413	模拟- 侧视图*
414	模拟- 端视图*

报警信息

状态	功能
500	显示信息
510	-/用户屏幕窗口
520	-/用户屏幕窗口

刀具零点偏移

状态	功能
600	刀具表
610	刀具磨损
620	OEM 刀具列表*
630	刀具库
640	零点偏移
650	R 参数
660	-/用户屏幕窗口
670	主轴
680	用户数据
690	机床数据

操作区“Custom（定制）”

5

5.1 供货状态和应用

概述

前面描述的操作区可以通过“补充操作界面”方式补充和修改。
补充只能放置在还未使用的软键上。

使用下面说明的方法都可以在 HMI 内置 sl 和 HMI
高级上设计一个独立的操作区，该操作区所有的 8 个水平和 8
个垂直软键都供用户专用的操作界面使用。

操作区的默认名为“定制”。

对于带热键区的 OP（例如：OP 010, OP 010C）可以通过以下按键直接选择操作区“定制”：

- 热键“定制”
- 第 1 个水平扩展栏的软键 4（预设）

供货状态下的属性

供货状态下，“定制”操作区为覆盖整个本地菜单区域的空窗口，其中，标题可以设计。
此外，“定制”操作区还允许在全局菜单的操作区显示栏内输入可配置的文本。

全部软键都是空的，用户可以通过“补充操作界面”任意定义软键。

在离开并再次返回“定制”操作区后，离开操作区“定制”前的有效画面生效。

- **HMI 高级**

  “定制”操作区属于供货的标准范畴，可以通过文件 REGIE.INI
  中的条目释放或者定制在其他任意一个水平键上。

- **HMI 内置 sl**

  “定制”操作区属于标准配置。
  它与应用程序磁盘一起提供，并且可以由客户通过软键配置套件（SCK.exe）定制在任意一个水平软键上。

5.2 激活操作区

HMI 高级

操作区在文件 REGIE.INI 中释放并定制在软键上。

举例：

通过第 1 个扩展栏的水平软键 4 激活（预设）。

- 条目在段落 [TaskConfiguration] 中：

```
[TaskConfiguration]
Task11 = name := custom, Timeout := 12000
```

- 在文件 REGIE.INI 中也可以选择该操作区用于标准启动。 条目在段落 [Miscellaneous] 中：

```
[Miscellaneous]
PoweronTaskIndex = 11
```

当不选择“定制”操作区或者在启动时不激活情况下，相应的行可以通过行开始处的注释标记“;”退出激活。 通过更改任务号可以将操作区定制在另一个软键上。

HMI 内置 sl

您可以通过显示机床数据 MD 9016： MM_SWITCH_TO_AREA
确定，应该在哪个操作区中运行 HMI 内置 sl，此时也可以规定操作区“定制”。

机床数据中的值规定定制所需的操作区的软键号码。

预设置：	12
水平软键 1 – 8：	1 – 8
扩展栏软键	9 – 16

用于应用程序 CUSTOM（定制）附属的 HMI 软件分析文件 CUSTOM.INI
并决定是否要显示操作区。 对于 HMI 内置 sl，文件包含段落 [Activate]：

```
[Activate]
Activate=True
```

操作区可以通过软件配置套件（SCK.EXE），菜单项“修改配置”由客户在任意一个水平软键上定制。

为了使用标准启动的定制操作区，软键号必须输入在显示机床数据 MD 9016：
MM_SWITCH_TO_AREA 中。

如果显示机床数据有值 -1，则 HMI 内置 sl
在该操作区中启动，该操作区已通过软件配置套件（SCK）指定作为启动操作区。
在供货状态下，基本栏的软键 1 在操作区“加工”中。

通过键激活

- **热键**

  通过操作面板 OP 010，OP 010C 上的“定制”热键总是可以激活该操作区（预设）。
  可以进行其它和附加的设计。

- **水平软键**

  通过第 1 个扩展栏的水平软键 4 激活（预设）。 在 HMI
  高级上可以更改软键的布局，而在 HMI 内置 sl 上则需要借助软键配置包(SCK)。

操作区域切换时的特性

如果由“定制”操作区切换到另一个操作区并再次返回，则在返回后显示离开“定制”操作区前已激活的画面。

5.3 定义开始对话框

概述

文件 CUSTOM.INI 中包含设计开始对话框的条目；对于 HMI 高级，文件 RE_xx.INI
可用于定义操作区的名称。

- **对话框标题**

  在段落 [Header] 中可以输入一个带有对话框标题的文本。
  标题可以是文本或者是报警文本号，从而可以根据不同语言设计标题：

```
[Header]
Text="XY 特殊功能"
Text=$80XXX
预设置： Text="Custom"
```

- **开始对话框中的图形**

  在段落 [Picture] 中可以输入在启动应用程序时显示的图形的路径。

```
[Picture]
Picture=\directory\bild.bmp
```

- **操作区名称**

HMI 高级	HMI 内置 sl
给定的名称出现在所显示的开始对话框的左上方。 [HSoftkeyTexts] HSK11 = "Custom"	给定的名称出现在所显示的开始对话框的左上方。 [Taskname] 标记： Text = $80xxx

- **软键标签**

HMI 高级	HMI 内置 sl
文件 RE_xx.IN 中的操作区名称显示在配置的软键上。 "xx" 为语言缩写。	规定的文本显示在分配的软键上。 如果没有规定，则标准设置为 “Custom”。 [Softkey] Text = $80xxx

操作区“定制”的其他所有单元，例如：软键栏或者输入/输出栏以及附属功能必须通过“补充操作界面”设计。

在应用程序“定制”中**所有**软键作为“补充操作界面”的软键使用。

根据上述说明，在文件 CUSTOM.COM 中（类似与供货范围内的 AEDITOR.COM）设计软键。

对于 HMI 内置 sl，文件 COMMON.COM 必须参照文件 COMMON.COM（用户登入软键）。而供货状态下的 COMMON.COM，操作区“定制”的所有软键都参照 CUSTOM.COM。

设计环境

6

6.1 供货范围

概述

用于编译操作界面设计文件和编制或者激活上述功能的软件都是 HMI 软件供货范围的一部分；同样用于建立设计文件内容的 ASCII 编辑器（程序编辑器）也是 HMI 软件供货范围的一部分。

产品

NCU 上的产品 ShopMill 和 ShopTurn 是以 HMI 内置 sl 为基础。 HMI 内置 WIN32 和 HMI 高级组合可以在一个 HW 平台上运行。 为此该产品的操作系统可以作为 HMI 高级的选择之一在 HW 平台上运行。

系统区别

由于不同的硬件，设计文件的保存也各不相同：

- PCU 50 上的 HMI 高级使用硬盘。
- HMI 内置 sl 只使用 CF 卡上的工作存储器和用户存储器。
- PCU 50 或者装有 Windows 的计算机上的 HMI 内置 WIN32 分析硬盘上的设计文件，除此之外仍然和 HMI 内置 sl 一样运行。

图形描述

如果要在设计的对话框中使用图形，则还需要一个合适的图形程序（例如：微软画笔）。

参见

查找方案的原理 (页 186)

6.2 创建设计文件

6.2.1 使用文件 COMMON.COM

概述

在使用 HMI 高级时不需要 COMMON.COM 中的条目。

HMI 内置 sl 上该中央控制文件包含下列信息：

- 分配登入软键给设计文件
- 分配画面号给 PLC 接口 DB19 中的设计文件
- 控制条目（LOG 文件的大小、设计文件临时目录中可用的存储位置）。

HMI 高级/ HMI 内置 sl 的预设：

基本画面	水平软键	设计文件
加工 JOG	1	MA_JOG.COM
加工 MDA	1	MA_MDA.COM
加工 自动方式	2	MA_AUTO.COM
Parameter	7	PARAM.COM
Program	8	PROG.COM
Service	7	SERVICE.COM
Diagnose	7	DIAG.COM
Startup 扩展软键栏	7 6, 7	STARTUP.COM
编辑器（已占用）	2, 3, 4, 5	AEDITOR.COM
编辑器 扩展软键栏	6 6, 7	AEDITOR.COM

命名惯例和文件大小

- HMI 内置 sl

  所有文件名必须符合 DOS 规定(xxxxxxxx.com)。

  最多允许 10 个设计文件。

  帮助图形的颜色格式： bmp 格式的 256 色位图。

  由所用 CF 卡的大小决定图形和设计文件的存储位置需求。 可以不规定文件最大数量。

- HMI 高级

  根据上述给定的顺序在目录中查找带有登入软键名称的文件。
  如果在不同的目录中存放相同文件名的文件，则找到的文件为按照查找顺序首先找到的文件。 设计文件的大小没有特别的限制。 然而应该注意，文件越大，文件处理速度就越慢。

6.2.2 文件 COMMON.COM 的结构

概述

文件 COMMON.COM 连同循环一起提供。 它包含特定硬件设置的不同段落。 对于 HMI 内置 sl，段落[MMC_DOS] 关系到“补充操作界面”。

设计 COMMON.COM

句法	参数 =*值*	;	在参数、'=' 和*值*之间可以有多个任意空格。
说明	[MMC_DOS]	;	DOS 段开始
		;	分号“;”后行内的文本是注释语句，不作分析。
参数	所有参数都是可选的。		
	SCxxx=*文件*	软键连接： 连接软键和设计文件	
		"xxx" 表示登入软键的内部软键属性。 软键属性必须紧跟在 SC... 后面。 仅显示已定义软键连接的软键。	
	HCyyy=*文件*	热键连接： 连接实际按键和设计文件。	
		"yyy" 表示登入热键的热键属性，热键属性必须紧跟在 HC... 后面。 只有已定义热键连接的热键才有效。	
	文件	包含软键和对话框定义的配置文件。 文件名最多允许 8 个字符长。 文件扩展名用点隔开。	
		举例： SC101= my_file.com ; (my_file.com 在 HMI, NC 上)	

注意

COMMON.COM 中的更改只有在重新启动后才生效。

控制项

句法： CHK_FILE_EXIST=*ram*

名称： 控制标志： 规定每次的设计文件是否要由 NC 复制，或者是否要进行检查，是否该文件已位于 HMI 的 RAM 驱动器上了。

参数： ram 可能的值：

0: 不检查文件是否已存放在 临时目录上。 该模式仅在 NC 上在线建立设计文件阶段设置。 为此 NC 设计文件的更改立即在 HMI 内置 sl 上生效，这导致图形建立的速度较慢。

1: 预设（如果在 CHK_FILE_EXIST 未规定情况下）： 设计文件只向临时目录中读入一次并于日后从那里进行处理。 这表明： 运行时间得到改善，然而 NC 上设计文件中的改变不会生效。

句法： LOGSIZE=*kB*

名称： 设立 HMI 的临时目录上的 LOG 文件名称为 ERROR.COM，它的大小通过该参数确定。

参数： kB LOG 文件的大小，以 KB 为单位（最大可以是 64 KB）。

句法： RAMDISK_SIZE = *kBrd*

名称： RAMDISK_SIZE 可以在任意位置上，在段内或者段外。 如果该单元多次出现，则取**第一次出现**的单元。

如果在复制过程**后**超过设置的大小，则在下一次复制过程**前**删除临时目录中的所有 COM 文件。（操作区切换时后台的对话框的文件保持不变。）

参数： kBrd 可支配的 RAMDISK 大小，以 kB 为单位。

预设置： 300 kByte （如果没有规定单元 RAMDISK_SIZE）

COMMON.COM 举例

```
[ MMC_DOS]
sc101=verzahn.com                ; MASCHINE
sc111=mda.com                    ; 文件位于 HMI 的闪存存储器上。
sc122=auto.com
sc207=param.com                  ; PARAMETER
sc314=aeditor.com                ; 程序编辑器
sc315=aeditor.com
sc316=aeditor.com
sc407=dienste.com                ; Service
sc507=diagnose.com               ; DIAGNOSE
sc607=inbetrn.com                ; Startup
sc826=cmm.com                    ; ShopMill, 加工, AUTO
sc857=cmm.com                    ; 报警/信息
sc858=cmm.com
sc867=cmm.com                    ; 刀具，零偏
CHK_COMMON.COM=1                 ; 更快处理 HMI
LOGSIZE=30                       ; 故障纪录（LOG 文件）的大小：30 kB

[PLC_SELECT]                     ; 定义可由 PLC 调用的画面
PC1= CYC82 bohren.com            ; 画面 1
PC2= CYCLE90, gewfraes.com       ; 画面 2
```

6.2.3 设计登入软键

概述

借助于规定的登入软键可以激活分配的设计文件。 固定定义可以对话框的登入软键。
其它登入软键是不可以的。 登入软键在各个操作区各有不同。

编程

句法	SCxxx=*文件*
名称	软键连接： 连接软键和设计文件 "xxx" 表示登入软键的内部软键属性
参数	*文件* 设计文件名称

操作区中的登入点

操作区	SCxxx	初始对话框	
MASCHINE	SC101	加工 JOG	水平软键 1
	SC111	加工 MDA	水平软键 1
	SC122	加工 AUTO	水平软键 2
PARAMETER	SC207	参数基本画面	水平软键 7
PROGRAM	SC308	程序基本画面	水平软键 8
	SC312	1. 编辑器软键行	水平软键 2
	SC313	编辑器的第 1 软键栏	水平软键 3
	SC314	编辑器的第 1 软键栏	水平软键 4
	SC315	编辑器的第 1 软键栏	水平软键 5
	SC316	编辑器的第 1 软键栏	水平软键 6
	SC326	编辑器的第 2 软键栏	水平软键 6
	SC327	编辑器的第 2 软键栏	水平软键 7
Service	SC407	通讯基本画面	水平软键 7
DIAGNOSE	SC507	诊断基本画面	水平软键 7
开机调试	SC607	开机调试基本画面	水平软键 7
	SC616	编辑器的第 2 软键栏	水平软键 6
	SC617	编辑器的第 2 软键栏	水平软键 7

章节“登入软键表”中给出的名称是预设。 然而附属的文件仍然必须在 NC 或者 HMI 上由用户设立。

参见

登入软键表 (页 191)

6.2.4 和语言相关的文本

概述

对话框中和语言相关的文本保存在 ASCII 文本文件中；句法和报警文本文件一致。

可以使用和语言相关的文本：

- 软键标记
- 标题
- 辅助文本
- 其它任意文本

允许的文件名

按如下方式确定文件名：

Alsc.txt	用于西门子标准循环、和语言相关的文本
Almc.txt	用于制造商循环、和语言相关的文本
Aluc.txt	和语言相关的用户文本

文本输入形式

句法	8xxxx 0 0 "文本"		
说明	文件中文本号和文本的排列		
参数	xxxx	5000 至 9899	预留于用户的文本识别号范围。 号码必须是唯一的。
	“文本”		显示在对话框中的文本

两个通过空格分开的参数 2 和 3 是用于报警文本输出的控制符号。
鉴于带有报警文本的文本格式的一致性，在任何情况下控制符号必须置零。

在文本中可以有下列控制符号：

%n	换行
%@x	x. 轴的轴名称 (x 为轴编号)，仅适用于 HMI 内置 sl 显示轴名称（HMI 内置 sl 和 HMI 高级）： NC 存取包含各轴名称的相应机床数据，文本组合通过包含的字符串功能。

举例：

85000 0 0	“退回平面”
85001 0 0	“钻削深度”
85002 0 0	“螺距”
85003 0 0	“凹槽半径”

6.3 设计文件的存档结构

6.3.1 HMI 内置 sl

存档

在 Linux 环境下，将用户设计文件复制到 CF 卡的目录**/user/sinumerik/hmi/proj** 下（适用于所有“标准”的用户对话框，即：除用于循环支持外的所有对话框）。
将供用户循环支持使用的用户设计文件复制到 CF 卡的目录**/user/sinumerik/cycles/proj** 下。
一般来说，所有文件**未压缩** 地直接复制到各目录下。

与此类似，将制造商的设计文件复制到目录**/oem/**sinumerik/.... 下。

路径	内容
/card/user/sinumerik/hmi/proj	用户设计文件（“标准”用户对话框的 com 文件，即：除用于循环支持外的所有对话框）：
/card/user/sinumerik/cycles/proj	用于用户循环支持的的用户设计文件
/card/oem/sinumerik/cycles/proj /card/oem/sinumerik/hmi/proj	制造商设计文件
/card/user/sinumerik/cycles/ico/icoxxx /card/oem/sinumerik/cycles/ico/icoxxx /card/user/sinumerik/hmi/ico/icoxxx /card/oem/sinumerik/hmi/ico/icoxxx	位图
/card/user/sinumerik/hmi/cfg /card/oem/sinumerik/hmi/cfg	Ini 文件
/card/user/sinumerik/hmi/ico/icoxxx /card/oem/sinumerik/hmi/ico/icoxxx	标题图标
/card/user/sinumerik/cycles/lng/xxx /card/oem/sinumerik/cycles/lng/xxx /card/user/sinumerik/hmi/lng/xxx	附属的文本(aluc.txt oder aluctx.s0x)，其中 xxx 表示语言

供货时，“程序”操作区设置了 3 个 USB 驱动器并能访问 CF 卡。

USB 存储器上设置了下列目录结构：

\cycles

\cycles\proj	(com 文件)
\cycles\prog	(用户循环 (.spf))
\cycles\lng	(语言目录– 仅包含子目录)
\cycles\lng\xxx	(语言目录，例如： deu, eng... – 每种语言一个目录)。（这里保存每个语言的**aluc.txt**文件。）
\cycles\ico	(图形目录– 仅包含各分辨率的子目录)
\cycles\ico\ico640	对于分辨率 640*480 的图形目录为**.bmp** 或 **.bin**
\cycles\ico\ico800	
\cycles\ico\ico1024	

分配给其他操作区的对话框：

\hmi\proj

\hmi\lng\...

\hmi\ico\...

在操作区“程序”中，把整个目录\cycles 或 \hmi 从 USB 存储器复制到 CF 卡的目录 /user/sinumerik 中。

6.3.2 HMI 高级

概述

在使用 HMI 高级时不需要控制文件条目。

在目录中按给出的顺序查找设计文件。
如果在不同的目录中存放相同文件名的文件，则找到的文件为按照查找顺序首先找到的文件。

在 PG/标准 PC 上测试

在 PG/标准 PC 上测试设计完成的对话框时，需满足以下边界条件：

- 已经在 PC/PG 上安装 HMI 高级软件的 PC 版本。
- 目录结构与 HMI 高级上的结构一致。
- 故障记录建立在： \DH\COM.DIR\ERROR.COM

存储报警文本文件

报警文本文件保存在下列目录中： \DH\MB.DIR\

文件名： ALUC_*xx*.COM

文献： HMI 高级版开机调试手册

6.4 HMI 系统使用共同的硬件平台时的查找方案

6.4.1 查找方案的原理

概述

HMI 内置 WIN32 如同 HMI 高级一样以相同的路径查找用于“补充操作界面”的配置文件。开始处是数据维护路径。

当 HMI 高级和 HMI 内置 sl在同一硬件平台上共同运行时，下列描述中的查找顺序对于 NCU 上的 ShopMill / ShopTurn 比较重要。 在这种情况下，HMI 内置 sl 上运行的 NCU 上的 ShopMill / ShopTurn 下的“补充操作界面”使用相同的配置文件，如同 HMI 高级下的操作。

框架条件

HMI 内置 WIN32 分析配置文件 DH.INI 的段 [DHSTART] 的特性 mmchome，该配置文件包含数据维护路径的根目录。 DH.INI 必须存放在 BIN 目录中，在该目录中启动 MMC0.EXE。 数据维护路径的数据最大长度为 100 个字符。

引导启动

在启动时 HMI 内置 WIN32 根据注册表中的条目首先确定 HMI 高级安装在哪里。在注册表规定的目录内在子目录**..\user, ..\oem, ..\add_on, ..\mmc2** 中以所述的顺序查找文件 DH.INI。 如果那里没有找到 DH.INI，则在 HMI 内置 WIN32 的当前设置的目录中查找文件。

举例：

如果 HMI 高级可在 f:\HMI\HMI Advanced 上找到；然后按照下列顺序查找 DH.INI：

- F:\HMI\HMI-Advanced\user
- F:\HMI\HMI-Advanced\oem
- F:\HMI\HMI-Advanced\add_on
- F:\HMI\HMI-Advanced\mmc2

查找顺序控制大小

HMI 高级的查找顺序

- CUS 目录，在 dh.ini 内特定的数据维护路径中
- CMA 目录，在 dh.ini 内特定的数据维护路径中
- CST 目录，在 dh.ini 内特定的数据维护路径中
- COM 目录，在 dh.ini 内特定的数据维护路径中

RAMDISK 上的目录：

Filename.bin

Filename.bmp .

不带路径的位图名称

对于 HMI 高级，这种文件有文件后缀 .bin

在没有规定文件夹和文件名不包含路径的情况下，查找顺序为：

Filename.bin ，在数据维护路径的**CUS**目录中

Filename.bmp ，在数据维护路径的**CUS**目录中

Filename.bin ，在文件夹*Filename.bi_* 中，数据维护路径的**CUS**目录下

Filename.bmp ，在文件夹*Filename.bm_* 中，数据维护路径的**CUS**目录下

Filename.bin ，在数据维护路径的**CUS\分辨率**目录中

Filename.bmp ，在数据维护路径的**CUS\分辨率**目录中

Filename.bin ，在文件夹*Filename.bi_* 中，数据维护路径下的 **CUS\分辨率**目录下。

Filename.bmp ，在文件夹*Filename.bm_* 中，数据维护路径下的 **CUS\分辨率**目录下。

Filename.bin ，在数据维护路径的**CMA**目录中

Filename.bmp ，在数据维护路径的**CMA**目录中

Filename.bin ，在文件夹*Filename.bi_* 中，数据维护路径的**CMA**目录下

Filename.bmp ，在文件夹*Filename.bm_* 中，数据维护路径的**CMA**目录下

Filename.bin ，在数据维护路径的**CMA\分辨率**目录中

Filename.bmp ，在数据维护路径的**CMA\分辨率**目录中

Filename.bin ，在文件夹*Filename.bi_* 中，数据维护路径下的 **CMA\分辨率**目录下。

Filename.bmp ，在文件夹*Filename.bm_* 中，数据维护路径下的 **CMA\分辨率**目录下。

Filename.bin ，在数据维护路径的**CST**目录中

Filename.bmp ，在数据维护路径的**CST**目录中

Filename.bin ，在文件夹*Filename.bi_* 中，数据维护路径的**CST**目录下

Filename.bmp ，在文件夹*Filename.bm_* 中，数据维护路径的**CST**目录下

Filename.bin ，在数据维护路径的**CST\分辨率**目录中

Filename.bmp ，在数据维护路径的**CST\分辨率**目录中

Filename.bin ，在文件夹*Filename.bi_* 中，数据维护路径下的 **CST\分辨率**目录下。

Filename.bmp ，在文件夹*Filename.bm_* 中，数据维护路径下的 **CST\分辨率**目录下。

Filename.bin 在当前目录中（Bin 目录）

Filename.bmp 在当前目录中（Bin 目录）

Filename.bin ，在文件夹*Filename.bi_* 中，当前目录下（Bin 目录）

Filename.bmp ，在文件夹*Filename.bm_* 中，当前目录下（Bin 目录）

Filename.bin ，在文件夹**CUS.ARJ** 中，数据维护路径的**CUS**目录下

Filename.bmp ，在文件夹**CUS.ARJ** 中，数据维护路径的**CUS**目录下

Filename.bin ，在文件夹**CUS.ARJ** 中，数据维护路径的**CUS\分辨率**目录下

Filename.bmp ，在文件夹**CUS.ARJ** 中，数据维护路径的**CUS\分辨率**目录下

Filename.bin ，在文件夹**CUS.ARJ** 中，数据维护路径的**CMA**目录下

Filename.bmp ，在文件夹**CUS.ARJ** 中，数据维护路径的**CMA**目录下
Filename.bin ，在文件夹**CUS.ARJ** 中，数据维护路径的**CMA\分辨率**目录下
Filename.bmp ，在文件夹**CUS.ARJ** 中，数据维护路径的**CMA\分辨率**目录下
Filename.bin ，在文件夹**CUS.ARJ** 中，数据维护路径的**CST**目录下
Filename.bmp ，在文件夹**CUS.ARJ** 中，数据维护路径的**CST**目录下
Filename.bin ，在文件夹**CUS.ARJ** 中，数据维护路径的**CST\分辨率**目录下
Filename.bmp ，在文件夹**CUS.ARJ** 中，数据维护路径的**CST\分辨率**目录下
对于 CMA.ARJ，步骤 29 至 40 现在重复进行
Filename.bin 在文件夹 CMA.ARJ 中， 数据维护路径的 CUS 目录下
....
Filename.bmp在文件夹 CMA.ARJ中， 数据维护路径的 CST\分辨率目录下
对于 CST.ARJ，步骤 41 至 52 现在重复进行
Filename.bin 在文件夹 CST.ARJ 中，数据维护路径的 CUS 目录下
....
Filename.bmp，在文件夹 CST.ARJ 中，数据维护路径的 CST\分辨率目录下
Filename.bin，在文件夹 CUS.ARJ 中，当前目录下（Bin 目录）
Filename.bmp，在文件夹 CUS.ARJ 中，当前目录下（Bin 目录）
Filename.bin，在文件夹 CMA.ARJ 中，当前目录下（Bin 目录）
Filename.bmp，在文件夹 CMA.ARJ 中，当前目录下（Bin 目录）
Filename.bin，在文件夹 CST.ARJ 中，当前目录下（Bin 目录）
Filename.bmp，在文件夹 CST.ARJ 中，当前目录下（Bin 目录）
Filename.bxx

部分二进制文件，出于与实际模式和保护模式开始的兼容性原因还必须处理。
这些文件仅在当前设置的目录中查找。

6.4.2 COMMON.COM 的查找方案

COMMON.COM 的存档位置

中央控制文件 COMMON.COM 必须位于 NC 的下列目录中的一个：

- CUS: 用户循环目录
- CMA: 制造商循环目录
- CST: 标准循环目录
- COM: 注释目录

目录按规定的顺序查找 COMMON.COM。 分析首先找到的符合该名称的文件。

此外， 对于 HMI 高级，控制文件 COMMON.COM 在数据维护的路径下：

..\dh\cus.dir

..\dh\cma.dir

..\dh\cst.dir

..\dh\com.dir

注意

必须根据 HMI 高级(ShopMill / ShopTurn)**重新启动** HMI 内置 WIN32，
文件中的修改才能生效。

6.4.3 图形的查找方案

查找顺序

HMI 内置 sl 的图形（位图）的扩展的查找方案也适用于“补充操作界面”的图形。

对于 HMI 内置 WIN32（ShopMill/ShopTurn），在一个硬件平台上与 HMI 高级一起，可以规定一个相关的数据维护路径。

如果已定义数据维护路径，则在当前目录前要首先处理，为此 HMI 内置 WIN32 和 HMI 高级使用 **相同的图形**。

新的查找结构也包括与补充操作界面相关的数据维护路径目录，以及图形可行的文件夹（cus.arj, cma.arj, cst.arj）。

原则上总是首先查找单个文件，然后才在可行的文件夹中查找。
为此得到下列图形的查找顺序：

- 文件夹前的单个图形（首先查找 .bin，然后查找 .bmp。
  在单个文件后在仅包含一个文件的文件夹中查找（.bi_，然后 .bm_）。
- 先是带有路径的位图名称，然后是不带路径的位图名称

参见

查找方案的原理 (页 186)

附录

A

A.1 登入软键表

ShopMill 和 ShopTurn 的登入软键

ShopMill	SCxxx	初始对话框	
	SC818	手动加工操作区（大对话框）	水平软键 8
	SC8181	手动加工操作区（中等对话框）	水平软键 8
	SC8182	手动加工操作区（小对话框）	水平软键 8
	SC8131	手动加工操作区-工件零点	垂直软键 1
	SC8132	手动加工操作区-工件零点	垂直软键 2
	SC8133	手动加工操作区-工件零点	垂直软键 3
	SC8134	手动加工操作区-工件零点	垂直软键 4
	SC8135	手动加工操作区-工件零点	垂直软键 5
	SC8136	手动加工操作区-工件零点	垂直软键 6
	SC8137	手动加工操作区-工件零点	垂直软键 7
	SC8141	手动加工操作区-刀具测量	垂直软键 1
	SC8142	手动加工操作区-刀具测量	垂直软键 2
	SC8143	手动加工操作区-刀具测量	垂直软键 3
	SC8144	手动加工操作区-刀具测量	垂直软键 4
	SC8145	手动加工操作区-刀具测量	垂直软键 5
	SC8146	手动加工操作区-刀具测量	垂直软键 6
	SC8147	手动加工操作区-刀具测量	垂直软键 7
	SC826	自动加工操作区（大对话框）	水平软键 6
	SC8261	自动加工操作区（中等对话框）	水平软键 6
	SC8262	自动加工操作区（小对话框）	水平软键 6
	SC8426	程序操作区-钻削	垂直软键 6
	SC8436	程序操作区-铣削	垂直软键 6
	SC8454	程序操作区-不同的	垂直软键 4
	SC8951	程序操作区-不同的-工件零点	垂直软键 1
	SC8952	程序操作区-不同的-工件零点	垂直软键 2
	SC8953	程序操作区-不同的-工件零点	垂直软键 3
	SC8954	程序操作区-不同的-工件零点	垂直软键 4
	SC8955	程序操作区-不同的-工件零点	垂直软键 5
	SC8956	程序操作区-不同的-工件零点	垂直软键 6

ShopMill	SCxxx	初始对话框	
	SC8957	程序操作区-不同的-工件零点	垂直软键 7
	SC8961	程序操作区-不同的-刀具测量	垂直软键 1
	SC8962	程序操作区-不同的-刀具测量	垂直软键 2
	SC8963	程序操作区-不同的-刀具测量	垂直软键 3
	SC8964	程序操作区-不同的-刀具测量	垂直软键 4
	SC8965	程序操作区-不同的-刀具测量	垂直软键 5
	SC8966	程序操作区-不同的-刀具测量	垂直软键 6
	SC8967	程序操作区-不同的-刀具测量	垂直软键 7
	SC857	信息 / 报警操作区	水平软键 7
	SC858	信息 / 报警操作区	水平软键 8
	SC867	刀具 / 零点偏移操作区	水平软键 7
	SC8492	程序操作区-G 代码编辑器	水平软键 2（循环支持轮廓）*
	SC8493	程序操作区-G 代码编辑器	水平软键 3（循环支持 钻削）*
	SC8494	程序操作区-G 代码编辑器	水平软键 4（循环支持 铣削）*
	SC8495	程序操作区-G 代码编辑器	水平软键 5（循环支持 车削）*
	SC8496	程序操作区-G 代码编辑器	水平软键 6*
	SC8406	程序操作区-G 代码编辑器（扩展范围）	水平软键 6（测量循环支持）*
	SC8407	程序操作区-G 代码编辑器（扩展范围）	水平软键 7（测量循环支持）*

* 即西门子对话框。

ShopTurn	SCxxx	初始对话框	
	SC818	手动加工操作区（大对话框）	水平软键 8
	SC8181	手动加工操作区（中等对话框）	水平软键 8
	SC8182	手动加工操作区（小对话框）	水平软键 8
	SC8131	手动加工操作区-工件零点	垂直软键 1
	SC8132	手动加工操作区-工件零点	垂直软键 2
	SC8133	手动加工操作区-工件零点	垂直软键 3
	SC8134	手动加工操作区-工件零点	垂直软键 4
	SC8135	手动加工操作区-工件零点	垂直软键 5
	SC8136	手动加工操作区-工件零点	垂直软键 6
	SC8137	手动加工操作区-工件零点	垂直软键 7
	SC8141	手动加工操作区-刀具测量	垂直软键 1
	SC8142	手动加工操作区-刀具测量	垂直软键 2
	SC8143	手动加工操作区-刀具测量	垂直软键 3
	SC8144	手动加工操作区-刀具测量	垂直软键 4
	SC8145	手动加工操作区-刀具测量	垂直软键 5
	SC8146	手动加工操作区-刀具测量	垂直软键 6

ShopTurn	SCxxx	初始对话框	
	SC8147	手动加工操作区-刀具测量	垂直软键 7
	SC826	自动加工操作区（大对话框）	水平软键 6
	SC8261	自动加工操作区（中等对话框）	水平软键 6
	SC8262	自动加工操作区（小对话框）	水平软键 6
	SC8246	程序操作区-钻削	垂直软键 6
	SC9436	程序操作区-车削	垂直软键 6
	SC9456	程序操作区-铣削	垂直软键 6
	SC8454	程序操作区-不同的	垂直软键 4
	SC857	信息 / 报警操作区	水平软键 7
	SC858	信息 / 报警操作区	水平软键 8
	SC867	刀具 / 零点偏移操作区	水平软键 7
	SC8492	程序操作区-G 代码编辑器	水平软键 2（循环支持轮廓）*
	SC8493	程序操作区-G 代码编辑器	水平软键 3（循环支持 钻削）*
	SC8494	程序操作区-G 代码编辑器	水平软键 4（循环支持 铣削）*
	SC8495	程序操作区-G 代码编辑器	水平软键 5（循环支持 车削）*
	SC8496	程序操作区-G 代码编辑器	水平软键 6*
	SC8406	程序操作区-G 代码编辑器（扩展范围）	水平软键 6（测量循环支持 - 车削）*
	SC8407	程序操作区-G 代码编辑器（扩展范围）	水平软键 7（测量循环支持 - 铣削）*

* 即西门子对话框。

A.2 颜色表

可以使用的颜色

对于 HMI 高级 和 HMI 内置
sl，提供统一的颜色表用于对话框设计（各个标准颜色的部分数量）。

序号	颜色
1	黑色
2	红褐色
3	深绿色
4	浅灰
5	深蓝色
6	蓝色
7	红色
8	棕色
9	黄色
10	白色

每个 HMI 程序中，颜色可能略有不同。

HMI 高级

对于 HMI 高级中的位图，在字符程序中必须使用软件提供的当前颜色表。

HMI 内置 sl

对于 HMI 内置 sl中的位图，在字符程序中必须使用软件提供的当前颜色表。
颜色表取决于选项“新式”。

您可在工具箱 8x0d\examples_tools\wizard.bsp\hmi_emb\... 下找到颜色表。

文件名给出了使用各个表格的提示：

- HMI_EMB_NEW_FASHION.PAL:

  该调色板用于“新式”的 HMI 内置 sl

  可提供颜色索引 160 至 231。

- HMI_EMB_OLD_AND_NEW_FASHION.PAL:

  该调色板用于 HMI 内置
  sl，和新式老式无关。用该调色板建立的位图对于新老式外观上完全相同。
  可提供颜色索引
  160、163、184、187、196、199、204、205、207、217、219、220、221、223、226 和 228。

以前的颜色表 HMI_EMB.PAL 已由上述表格替换。 仅允许使用颜色 160 至 231。
只有这样才能确保 HMI 内置 sl和 HMI 高级 上的图形外观上完全相同。

在 Paint Shop Pro 下激活调色板：

- 文件 → 打开 → ...*.bmp
- 颜色 → 打开图形板 → ...*.PAL
- 通过选项“颜色索引： 打开”使用调色板

系统颜色

对于单元（文本、输入栏、背景等等）可以从 10 种颜色中选择一种。
系统颜色上得以扩展，老式和新式之间颜色有所不同（例如：标题颜色）。

为了可以区别单一颜色和**设计相关**的颜色，确定单一颜色位于颜色索引 0 至 128 之间。
扩展框架内新引入的与设计相关的颜色从颜色索引 128 起定义。
以此即使对于单一颜色的扩展（最大至 128），也可以避免两种颜色类型的混淆。

新定义的颜色

索引	颜色描述	颜色	
		老式	新式
128	聚焦系统颜色	黄色	桔黄色
129	背景颜色	灰色	浅灰
130	标题颜色（激活）	黄色	蓝色
131	标题字体颜色（激活）	黑色	白色

A.3 可用的系统变量列表

名称	索引	说明
$A_DBB[x]	x=字节号	数据位 由 / 位于 PLC 上
$A_DBD[x]	x=偏移	数据双字（32 位）由 / 位于 PLC 上
$A_DBR[x]	x=偏移	实数数据（32 位）由 / 位于 PLC 上
$A_DBW[x]	x=偏移	数据字 （16 位）由 / 位于 PLC 上
$A_DLB[索引]	索引=偏移	左边区域中的数据位
$A_DLD[索引]	索引=偏移	在左边区域中访问双字节数据
$A_DLR[索引]	索引=偏移	左边区域中的实数数据
$A_DLW[索引]	索引=偏移	左边区域中的数据字
$A_IN[x]	x=数字输入号	HW 数字输入端的值
$A_INA[x]	x=模拟输入号	HW 模拟输入端的值
$A_INCO[x]	x=输入号	NC 比较仪输入端
$A_INSE		安全可编程逻辑： NCK 外设外部输入端
$A_INSED		安全可编程逻辑： 外部 NCK 输入端等价
$A_INSEP		安全可编程逻辑： PLC 外设外部输入端
$A_INSEPD		安全可编程逻辑： 等价外部 PLC 输入端
$A_INSI		安全可编程逻辑： 内部 NCK 安全输入端
$A_INSID		安全可编程逻辑： 等价内部 NCK 安全输入端
$A_INSIP		安全可编程逻辑： 内部 PLC 安全输入端
$A_LINK_TRANS_RATE		连接传输率
$A_MARKERSI		安全可编程逻辑： NCK 标记
$A_MARKERSIP		安全可编程逻辑： 等价 PLC 标记
$A_OUT[x]	x=数字输出号	HW 数字输出端值
$A_OUTA[x]	x=模拟输出号	HW 模拟输出端的值
$A_OUTSE		安全可编程逻辑： NCK 外设外部输出端
$A_OUTSED		安全可编程逻辑： 等价外部 NCK 输出端
$A_OUTSEP		安全可编程逻辑： 外部 PLC 外设输出端
$A_OUTSEPD		安全可编程逻辑： 等价外部 PLC 输出端

名称	索引	说明
$A_OUTSI		安全可编程逻辑： 内部 NCK 安全输出端
$A_OUTSID		安全可编程逻辑： 等价内部 NCK 安全输出端
$A_OUTSIP		安全可编程逻辑： 内部安全 PLC 输出端 611D
$A_OUTSIPD		安全可编程逻辑： 等价内部安全 PLC 输出端 611D
$A_TIMERSI		安全可编程逻辑： NCK 计时器
$A_PBB_IN[索引]	索引=偏移	IN 数据位
$A_PBB_OUT[索引]	索引=偏移	OUT 数据位
$A_PBD_IN[索引]	索引=偏移	IN 数据双字
$A_PBD_OUT[索引]	索引=偏移	OUT 数据双字
$A_PBR_IN[索引]	索引=偏移	IN 实数数据
$A_PBR_OUT[索引]	索引=偏移	OUT 实数数据
$A_PBW_IN[索引]	索引=偏移	IN 数据字
$A_PBW_OUT[索引]	索引=偏移	OUT 数据字
$A_TC_FCT		命令号
$A_TC_LFN		源位置号
$A_TC_LFO		源位置号
$A_TC_LTN		目标位置号
$A_TC_LTO		目标位置号
$A_TC_MFN		源库
$A_TC_MFO		源库号
$A_TC_MTN		目标库号
$A_TC_MTO		目标库号
$A_TC_STATUS		命令状态
$A_TC_THNO		刀架号
$A_TC_TNO		T 号码
$A_TOOLMLN[x]	x=刀具号 T	当前位置
$A_TOOLMN[x]	x=刀具号 T	当前库
$AA_COUP_ACT[x]	x=跟随主轴	跟随主轴当前的耦合状态
$AA_COUP_OFFS[x]	x=轴	与导向轴 / 导向主轴的偏移，额定值
$AA_COUP_OFFS[x]	x=主轴	同步主轴位置偏移，额定值一方
$AA_CURR[x]	x=轴	轴或主轴的电流实际值
$AA_DELT[x]	x=轴	工件坐标系中的驱动专用余程
$AA_DTBB[x]	x=轴	基本坐标系中从程序段开始的驱动专用路程
$AA_DTBW[x]	x=轴	工件坐标系中从程序段开始的驱动专用路程
$AA_DTEB[x]	x=轴	基本坐标系中程序段结束的驱动专用路程
$AA_DTEPB[x]	x=轴	基本坐标系中摆动进给的驱动专用余程
$AA_DTEPW[x]	x=轴	工件坐标系中摆动进给的驱动专用余程
$AA_DTEW[x]	x=轴	工件坐标系中程序段结束的驱动专用路程

名称	索引	说明
$AA_EG_ACTIVE [a,b]	a=随动轴 b=导向轴	EG 耦合激活
$AA_EG_AX[n,a]	n=索引导向轴 a=随动轴	EG 导向轴号码
$AA_EG_DENOM [a,b]	a=随动轴 b=导向轴	EG 命名器耦合系数
$AA_EG_NUM_LA[a]	a=随动轴	EG 导向轴数量
$AA_EG_NUMERA [a,b]	a=随动轴 b=导向轴	EG 计数器耦合系数
$AA_EG_SYN[a,b]	a=随动轴 b=导向轴	EG 同步位置 导向轴
$AA_EG_SYNCDIFF[a]	a=轴命名符	EG 同步运行偏差
$AA_EG_SYNFA[a]	a=随动轴	EG 同步位置 随动轴
$AA_EG_TYPE[a,b]	a=随动轴 b=导向轴	EG 耦合类型
$AA_ESR_ENABLE[a]	a=轴	ESR 轴使能
$AA_ESR_ENABLE[a]	a=轴	ESR 许可
$AA_ESR_STAT[a]	a=轴	ESR 状态
$AA_ETRANS[x]	x=框架号	外部框架偏移
$AA_FXS[x]	x=轴	运行到固定挡块后状态
$AA_IBN[x]	x=轴	刀具托架实际值
$AA_IEN[x]	x=轴	接收参照工件零点的有效刀具
$AA_IM[x]	x=轴	刀具托架
$AA_IW[x]	x=轴	刀具托架位置，额定值
$AA_LEAD_P[x]	x=轴	实际的导向值 - 位置
$AA_LEAD_SP[x]	x=轴	模拟的导向值 - 位置
$AA_LEAD_SV[x]	x=轴	模拟的导向值 - 速度
$AA_LEAD_TYP[x]	x=轴	引导值源
$AA_LEAD_V[x]	x=轴	实际的导向值 - 速度
$AA_LOAD[x]	x=轴	驱动载荷（以百分比为单位）（仅对于 611D）
$AA_MM[x]	x=轴	机床坐标系中的测量值
$AA_MM1[x]	x=轴	访问机床坐标系中的测量结果
$AA_MM2[x]	x=轴	访问机床坐标系中的测量结果
$AA_MM3[x]	x=轴	访问机床坐标系中的测量结果
$AA_MM4[x]	x=轴	访问机床坐标系中的测量结果
$AA_MW[x]	x=轴	工件坐标系中的测量值
$AA_OFF[x]	x=轴	编程轴的叠加运动
$AA_OFF_LIMIT[x]	x=轴	达到驱动专用极限值，用于补偿 $AA_OFF
$AA_OSCILL_REVERSE_POS1[x]	x=轴	同步指令往复运动中的当前返回位置 1
$AA_OSCILL_REVERSE_POS2[x]	x=轴	同步指令往复运动中的当前返回位置 2
$AA_OVR[x]	x=轴	运动同步指令的驱动专用倍率
$AA_POWER[x]	x=轴	驱动有效功率[Watt]
$AA_S[x]	x=主轴号（主轴号）	主轴转速，标准值
$AA_SOFTENDN[x]	x=轴	软件限位，负方向

名称	索引	说明
$AA_SOFTENDP[x]	x=轴	软件限位，正方向
$AA_STAT[x]	x=轴	轴状态
$AA_SYNA_MEM		运动同步指令的空余存储器
$AA_SYNC[x]	x=轴	导向值耦合时随动轴耦合
$AA_TORQUE[x]	x=轴	驱动力矩额定值单位[Nm]
$AA_TYP[x]	x=轴	轴类型
$AA_VACTB[x]	x=轴	驱动专用进给，实际值
$AA_VACTM[x]	x=轴	驱动专用进给，额定值
$AA_VACTW[x]	x=轴	驱动专用进给，实际值
$AA_VC[x]	x=轴	驱动专用进给 附加路径进给补偿
$AC_ALARM_STAT		ESR 报警状态
$AC_AXCTSWA[CTn]	CTn=轴容器号	轴容器状态
$AC_DELT		工件坐标系余程轨迹
$AC_DRF[x]	x=轴	DRF 值
$AC_DTBB		基本坐标系中与程序段开始的距离
$AC_DTBW		工件坐标系与程序段开始的距离
$AC_DTEB		基本坐标系中与程序段结束的距离
$AC_DTEPB		基本坐标系中往复运动进给的余程
$AC_DTEPW		工件坐标系中往复运动进给的余程
$AC_DTEW		工件坐标系与程序段结束的距离
$AC_FCT0[x]	x=多项式号	a0-系数 n. 同步措施多项式
$AC_FCT1[x]	x=多项式号	a1-系数 n. 同步措施多项式
$AC_FCT2[x]	x=多项式号	a2-系数 n. 同步措施多项式
$AC_FCT3[x]	x=多项式号	a3-系数 n. 同步措施多项式
$AC_FCTLL[x]	x=多项式号	下限值 n. 同步措施多项式
$AC_FCTUL[x]	x=多项式号	上限值 n. 同步措施多项式
$AC_FIFOx[y]	x=FIFONo (1-10) y=参数号	FIFO 同步措施变量
$AC_MARKER[x]	x=标记号	用于运动同步指令的计数器
$AC_MEA		已接通测量探头
$AC_OVR		同步指令的轨迹倍率
$AC_PARAM[x]	x=参数号	运动同步指令动态参数
$AC_PATHN		标准轨迹参数
$AC_PLTBB		基本坐标系中程序段开始的轨迹路程
$AC_PLTEB		基本坐标系中程序段结束轨迹行程
$AC_PRESET[x]	x=轴	PRESET 值
$AC_PROG		程序状态
$AC_RETPOINT[x]	x=轴	再次返回的轮廓复位点
$AC_SDIR[x]	x=主轴号（主轴号）	旋转状态
$AC_SMODE[x]	x=主轴号（主轴号）	主轴工作方式
$AC_STAT		通道状态

名称	索引	说明
$AC_TIME		程序段开始时间（秒）（包括内部生成的中间程序段时间）
$AC_TIMES		程序段开始时间（以秒为单位）（不包括内部生成的中间程序段时间）
$AC_TIMEC		程序段开始时间（以插补周期为单位）（包括内部生成的中间程序段周期）
$AC_TIMESC		程序段开始时间（以插补周期为单位）（不包括内部生成的中间程序段周期）
$AC_TIMER[x]	x=定时器号	时间单元（以秒为单位）
$AC_VACTB		插补进给，额定值
$AC_VACTW		工件坐标系中的轨迹速度
$AC_VC		同步指令的增加轨迹进给补偿
$AN_AXCTAS[n]		轴容器实际地址（旋转位置）
$AN_AXCTSWA[CTn]	CTn=轴容器号	轴容器旋转激活
$AN_ESR_TRIGGER		ESR 启动信号
$AN_MARKER[x]	x=标记号	运动同步措施的标记变量
$MC_DISPLAY_AXIS	位 16-31	识别机床轴
$MC_DISPLAY_AXIS	位 0-15	几何轴 / 辅助轴标识
$MC_MM_NUM_BASE_FRAMES		通道中基本框架数目
$MN_EXTERN_LANGUAGE $MN_EXTERN_CNC_SYSTEM	和	CNC 系统语言
$MN_MAX_CUTTING_EDGE_NO		D 号码最大值
$MN_MAX_CUTTING_EDGE_ PER_TOOL		每个刀具的最大刀沿数
$MN_MAX_SUMCORR_ PERCUTTING_EDGE		每个刀沿的最大总和修正数
$MN_MM_KIND_OF_SUMCORR		NCK 中总和修正特性
$MN_MM_NUM_CC_MAGAZINE_ PARAM		刀具库的参数数目
$MN_MM_NUM_CC_MAGLOC_ PARAM		刀具库位置的参数数目
$MN_MM_NUM_CC_MON_ PARAM		刀沿监控用户数据的参数数目
$MN_MM_NUM_GLOBAL_BASE_ FRAMES		与通道相关的基本框架数
$MN_MM_NUM_SUMCORR		NCK 中总和修正的总数目
$MN_MM_NUM_TOOL_ADAPTER		运行的刀具适配器的最大数组数
$MN_MM_TOOL_MANAGEMENT_ MASK		设置 NCK 刀具管理
$P_UBFR[x ,MI]	x=框架号	镜像可设置框架
$P_UBFR[x,RT]	x=框架号	旋转可设置框架
$P_UBFR[x,SC]	x=框架号	可设置框架校准系数
$P_UBFR[x,SI]	x=框架号	框架的微小偏移
$P_UBFR[x,TR]	x=框架号	转换可设置框架
$P_CHBFRMASK		有效的通道专用基本框架
$P_EG_BC[a]		EG 程序段切换标准

名称	索引	说明
$P_NCBFRMASK		有效的与通道相关的基本框架
$P_OFFN		偏移标准
$P_PFRAME[x,TR] $P_ACTFRAME 比如在编程 G54 之后，$P_IFRAME包含由 G54 定义的平移、旋转、比例和镜像。	或者 或者	转换激活的框架
$P_TOOL		激活的刀具刀沿号码
$P_TOOLL[1]		激活的刀具长度 1
$P_TOOLL[2]		激活的刀具长度 2
$P_TOOLL[3]		激活的刀具长度 3
$P_TOOLND[x]	x=刀具号	刀沿数目
$P_TOOLNO		激活的刀具号码
$P_TOOLR		激活的刀具半径
$P_UIFR[x,y,MI]	x=框架号，y=轴	镜像可设置框架
$P_UIFR[x,y,RT]	x=框架号，y=轴	旋转可设置框架
$P_UIFR[x,y,SC]	x=框架号，y=轴	可设置框架校准系数
$P_UIFR[x,y,SI]	x=框架号，y=轴	框架的微小偏移
$P_UIFR[x,y,TR]	x=框架号，y=轴	转换可设置框架
$P_UIFRNUM		激活的设置的框架索引
$R[x]	x=参数号	R 参数
$SC_PA_ACTIV_IMMED[x]	x=保护区号	返回参考点后立即激活
$SC_PA_CENT_ABS[x,0]	x=保护区号	第 1 个轮廓元素的圆心横坐标
$SC_PA_CENT_ABS[x,1]	x=保护区号	第 2 个轮廓元素的圆心横坐标
$SC_PA_CENT_ABS[x,2]	x=保护区号	第 3 个轮廓元素的圆心横坐标
$SC_PA_CENT_ABS[x,3]	x=保护区号	第 4 个轮廓元素的圆心横坐标
$SC_PA_CENT_ABS[x,4]	x=保护区号	第 5 个轮廓元素的圆心横坐标
$SC_PA_CENT_ABS[x,5]	x=保护区号	第 6 个轮廓元素的圆心横坐标
$SC_PA_CENT_ABS[x,6]	x= 号码保护区	第 7 个轮廓元素的圆心横坐标
$SC_PA_CENT_ABS[x,7]	x=保护区号	第 8 个轮廓元素的圆心横坐标
$SC_PA_CENT_ABS[x,8]	x=保护区号	第 9 个轮廓元素的圆心横坐标
$SC_PA_CENT_ABS[x,9]	x=保护区号	第 10 个轮廓元素的圆心横坐标
$SC_PA_CENT_ORD[x,0]	x=保护区号	第 1 个轮廓元素的圆心纵坐标
$SC_PA_CENT_ORD[x,1]	x=保护区号	第 2 个轮廓元素的圆心纵坐标
$SC_PA_CENT_ORD[x,2]	x=保护区号	第 3 个轮廓元素的圆心纵坐标
$SC_PA_CENT_ORD [x,3]	x=保护区号	第 4 个轮廓元素的圆心纵坐标
$SC_PA_CENT_ORD[x,4]	x=保护区号	第 5 个轮廓元素的圆心纵坐标
$SC_PA_CENT_ORD[x,5]	x=保护区号	第 6 个轮廓元素的圆心纵坐标
$SC_PA_CENT_ORD[x,6]	x=保护区号	第 7 个轮廓元素的圆心纵坐标
$SC_PA_CENT_ORD[x,7]	x=保护区号	第 8 个轮廓元素的圆心纵坐标
$SC_PA_CENT_ORD[x,8]	x=保护区号	第 9 个轮廓元素的圆心纵坐标
$SC_PA_CENT_ORD[x,9]	x=保护区号	第 10 个轮廓元素的圆心纵坐标
$SC_PA_CONT_ABS[x,0]	x=保护区号	第 1 个轮廓元素的终点横坐标

名称	索引	说明
$SC_PA_CONT_ABS[x,1]	x=保护区号	第 2 个轮廓元素的终点横坐标
$SC_PA_CONT_ABS[x,2]	x=保护区号	第 3 个轮廓元素的终点横坐标
$SC_PA_CONT_ABS[x,3]	x=保护区号	第 4 个轮廓元素的终点横坐标
$SC_PA_CONT_ABS[x,4]	x=保护区号	第 5 个轮廓元素的终点横坐标
$SC_PA_CONT_ABS[x,5]	x=保护区号	第 6 个轮廓元素的终点横坐标
$SC_PA_CONT_ABS[x,6]	x=保护区号	第 7 个轮廓元素的终点横坐标
$SC_PA_CONT_ABS[x,7]	x=保护区号	第 8 个轮廓元素的终点横坐标
$SC_PA_CONT_ABS[x,8]	x=保护区号	第 9 个轮廓元素的终点横坐标
$SC_PA_CONT_ABS[x,9]	x=保护区号	第 10 个轮廓元素的终点横坐标
$SC_PA_CONT_NUM[x]	x=保护区号	有效的轮廓元素数目
$SC_PA_CONT_ORD[x,0]	x=保护区号	第 1 个轮廓元素的终点纵坐标
$SC_PA_CONT_ORD[x,1]	x=保护区号	第 2 个轮廓元素的终点纵坐标
$SC_PA_CONT_ORD[x,2]	x=保护区号	第 3 个轮廓元素的终点纵坐标
$SC_PA_CONT_ORD[x,3]	x=保护区号	第 4 个轮廓元素的终点纵坐标
$SC_PA_CONT_ORD[x,4]	x=保护区号	第 5 个轮廓元素的终点纵坐标
$SC_PA_CONT_ORD[x,5]	x=保护区号	第 6 个轮廓元素的终点纵坐标
$SC_PA_CONT_ORD[x,6]	x=保护区号	第 7 个轮廓元素的终点纵坐标
$SC_PA_CONT_ORD[x,7]	x=保护区号	第 8 个轮廓元素的终点纵坐标
$SC_PA_CONT_ORD[x,8]	x=保护区号	第 9 个轮廓元素的终点纵坐标
$SC_PA_CONT_ORD[x,9]	x=保护区号	第 10 个轮廓元素的终点纵坐标
$SC_PA_CONT_TYP[x,0]	x=保护区号	第 1 个轮廓元素的轮廓类型
$SC_PA_CONT_TYP[x,1]	x=保护区号	第 2 个轮廓元素的轮廓类型
$SC_PA_CONT_TYP[x,2]	x=保护区号	第 3 个轮廓元素的轮廓类型
$SC_PA_CONT_TYP[x,3]	x=保护区号	第 4 个轮廓元素的轮廓类型
$SC_PA_CONT_TYP[x,4]	x=保护区号	第 5 个轮廓元素的轮廓类型
$SC_PA_CONT_TYP[x,5]	x=保护区号	第 6 个轮廓元素的轮廓类型
$SC_PA_CONT_TYP[x,6]	x=保护区号	第 7 个轮廓元素的轮廓类型
$SC_PA_CONT_TYP[x,7]	x=保护区号	第 8 个轮廓元素的轮廓类型
$SC_PA_CONT_TYP[x,8]	x=保护区号	第 9 个轮廓元素的轮廓类型
$SC_PA_CONT_TYP[x,9]	x=保护区号	第 10 个轮廓元素的轮廓类型
$SC_PA_LIM_3DIM[x]	x=保护区号	保护区极限 垂直坐标
$SC_PA_MINUS_LIM[x]	x=保护区号	保护区下限，垂直坐标
$SC_PA_ORI[x]	x=保护区号	保护区层面分配
$SC_PA_PLUS_LIM[x]	x=保护区号	保护区上限，垂直坐标
$SC_PA_T_W[x]	x=保护区号	工件或者刀具相关的保护区
$SN_PA_ACTIV_ IMMED[x]	x=保护区号	返回参考点后立即激活
$SN_PA_CENT_ABS[x,0]	x=保护区号	第 1 个轮廓元素的圆心横坐标
$SN_PA_CENT_ABS[x,1]	x=保护区号	第 2 个轮廓元素的圆心横坐标
$SN_PA_CENT_ABS [x,2]	x=保护区号	第 3 个轮廓元素的圆心横坐标
$SN_PA_CENT_ABS[x,3]	x=保护区号	第 4 个轮廓元素的圆心横坐标
$SN_PA_CENT_ABS[x,4]	x=保护区号	第 5 个轮廓元素的圆心横坐标
$SN_PA_CENT_ABS[x,5]	x=保护区号	第 6 个轮廓元素的圆心横坐标

名称	索引	说明
$SN_PA_CENT_ABS[x,6]	x=保护区号	第 7 个轮廓元素的圆心横坐标
$SN_PA_CENT_ABS[x,7]	x=保护区号	第 8 个轮廓元素的圆心横坐标
$SN_PA_CENT_ABS[x,8]	x=保护区号	第 9 个轮廓元素的圆心横坐标
$SN_PA_CENT_ABS[x,9]	x=保护区号	第 10 个轮廓元素的圆心横坐标
$SN_PA_CENT_ORD[x,0]	x=保护区号	第 1 个轮廓元素的圆心纵坐标
$SN_PA_CENT_ORD[x,1]	x=保护区号	第 2 个轮廓元素的圆心纵坐标
$SN_PA_CENT_ORD [x,2]	x=保护区号	第 3 个轮廓元素的圆心纵坐标
$SN_PA_CENT_ORD[x,3]	x=保护区号	第 4 个轮廓元素的圆心纵坐标
$SN_PA_CENT_ORD[x,4]	x=保护区号	第 5 个轮廓元素的圆心纵坐标
$SN_PA_CENT_ORD[x,5]	x=保护区号	第 6 个轮廓元素的圆心纵坐标
$SN_PA_CENT_ORD[x,6]	x=保护区号	第 7 个轮廓元素的圆心纵坐标
$SN_PA_CENT_ORD[x,7]	x=保护区号	第 8 个轮廓元素的圆心纵坐标
$SN_PA_CENT_ORD[x,8]	x=保护区号	第 9 个轮廓元素的圆心纵坐标
$SN_PA_CENT_ORD[x,9]	x=保护区号	第 10 个轮廓元素的圆心纵坐标
$SN_PA_CONT_ABS[x,0]	x=保护区号	第 1 个轮廓元素的终点横坐标
$SN_PA_CONT_ABS[x,1]	x=保护区号	第 2 个轮廓元素的终点横坐标
$SN_PA_CONT_ABS[x,2]	x=保护区号	第 3 个轮廓元素的终点横坐标
$SN_PA_CONT_ABS [x,3]	x=保护区号	第 4 个轮廓元素的终点横坐标
$SN_PA_CONT_ABS[x,4]	x=保护区号	第 5 个轮廓元素的终点横坐标
$SN_PA_CONT_ABS[x,5]	x=保护区号	第 6 个轮廓元素的终点横坐标
$SN_PA_CONT_ABS[x,6]	x=保护区号	第 7 个轮廓元素的终点横坐标
$SN_PA_CONT_ABS[x,7]	x=保护区号	第 8 个轮廓元素的终点横坐标
$SN_PA_CONT_ABS[x,8]	x=保护区号	第 9 个轮廓元素的终点横坐标
$SN_PA_CONT_ABS[x,9]	x=保护区号	第 10 个轮廓元素的终点横坐标
$SN_PA_CONT_NUM[x]	x=保护区号	有效的轮廓元素数目
$SN_PA_CONT_ORD[x,0]	x=保护区号	第 1 个轮廓元素的终点纵坐标
$SN_PA_CONT_ORD[x,1]	x=保护区号	第 2 个轮廓元素的终点纵坐标
$SN_PA_CONT_ORD[x,2]	x=保护区号	第 3 个轮廓元素的终点纵坐标
$SN_PA_CONT_ORD[x,3]	x=保护区号	第 4 个轮廓元素的终点纵坐标
$SN_PA_CONT_ORD[x,4]	x=保护区号	第 5 个轮廓元素的终点纵坐标
$SN_PA_CONT_ORD[x,5]	x=保护区号	第 6 个轮廓元素的终点纵坐标
$SN_PA_CONT_ORD[x,6]	x=保护区号	第 7 个轮廓元素的终点纵坐标
$SN_PA_CONT_ORD[x,7]	x=保护区号	第 8 个轮廓元素的终点纵坐标
$SN_PA_CONT_ORD[x,8]	x=保护区号	第 9 个轮廓元素的终点纵坐标
$SN_PA_CONT_ORD[x,9]	x=保护区号	第 10 个轮廓元素的终点纵坐标
$SN_PA_CONT_TYP[x,0]	x=保护区号	第 1 个轮廓元素的轮廓类型
$SN_PA_CONT_TYP[x,1]	x=保护区号	第 2 个轮廓元素的轮廓类型
$SN_PA_CONT_TYP[x,2]	x=保护区号	第 3 个轮廓元素的轮廓类型
$SN_PA_CONT_TYP[x,3]	x=保护区号	第 4 个轮廓元素的轮廓类型
$SN_PA_CONT_TYP[x,4]	x=保护区号	第 5 个轮廓元素的轮廓类型
$SN_PA_CONT_TYP[x,5]	x=保护区号	第 6 个轮廓元素的轮廓类型
$SN_PA_CONT_TYP[x,6]	x=保护区号	第 7 个轮廓元素的轮廓类型

名称	索引	说明
$SN_PA_CONT_TYP[x,7]	x=保护区号	第 8 个轮廓元素的轮廓类型
$SN_PA_CONT_TYP[x,8]	x=保护区号	第 9 个轮廓元素的轮廓类型
$SN_PA_CONT_TYP[x,9]	x=保护区号	第 10 个轮廓元素的轮廓类型
$SN_PA_LIM_3DIM[x]	x=保护区号	保护区极限 垂直坐标
$SN_PA_MINUS_LIM[x]	x=保护区号	保护区下限，垂直坐标
$SN_PA_ORI[x]	x=保护区号	保护区层面分配
$SN_PA_PLUS_LIM[x]	x=保护区号	保护区上限，垂直坐标
$SN_PA_T_W[x]	x=保护区号	工件或者刀具相关的保护区
$TC_ADPT ...		适配器数据
$TC_ADPTx $TC_ADPTT	x=1 ... 3	每个适配器的参数数目
$TC_DPCE		转换的刀沿修正值
$TC_DPCx[y,z]	x=参数号 y=刀具号, z=刀沿号	用户定义的刀具刀沿参数
$TC_DPx[y,z]	x=参数号 y=刀具号, z=刀沿号	刀沿修正值
$TC_DPx[y,z]	x=参数号 y=刀具号, z=刀沿号	转换的刀沿修正值
$TC_ECP ...		转换的位置相关的设立修正
$TC_MAMP3		磨损相关策略
$TC_MAP1		刀库类型
$TC_MAP2		刀库命名符
$TC_MAP3		刀库状态
$TC_MAP4		到后一个刀库的刀库链 1
$TC_MAP5		到前一个刀库的刀库链 2
$TC_MAP6		刀库尺寸
$TC_MAP9		激活的磨损组号码
$TC_MAPCx[y]	x=参数号 y=刀库号	每个刀库的用户数据
$TC_MOP1(x,y) ?$TC_MOP15(x,y)	x=刀具号 y=刀沿号	每个刀沿的监控数据
$TC_MOPCx[y,z]	x=参数号 y=T 号 z=刀沿	刀沿的监控用户数据
$TC_MPPCx[y,z]	x= 参数号 y= 刀库号 z=刀库位置号	一个刀库的刀库位置用户数据
$TC_MPPx	x=1,...7	每个刀库位置的参数数目
$TC_SCP...		转换的位置相关的磨损修正
$TC_SCP...		与位置相关的磨损修正
$TC_SCPx	x=13,...21,....71	每个总补偿程序段的总补偿参数数目
$TC_TP1		DUPLO 号码
$TC_TP10		替换刀具的刀具查找类型
$TC_TP11		HMI 的刀具信息

名称	索引	说明
$TC_TP2		刀具命名符
$TC_TP3		在半位置向左的大小
$TC_TP4		在半位置向右的大小
$TC_TP5		在半位置向上的大小
$TC_TP6		在半位置向下的大小
$TC_TP7		刀库位置类型
$TC_TP8		刀具状态
$TC_TP9		刀具监控类型
$TC_TPCx[y]	x=参数号 y=刀具号	用户定义的刀具参数
$TC_TPG1		主轴号码
$TC_TPG2		级联规则
$TC_TPG3		最小砂轮直径
$TC_TPG4		最小砂轮宽度
$TC_TPG5		当前砂轮宽度
$TC_TPG6		砂轮最大速度
$TC_TPG7		砂轮最大圆周速度（SUG）
$TC_TPG8		斜砂轮的倾斜角度
$TC_TPG9		SUG 的修正参数
$VA_COUP_OFFS[x]	x=轴	对导向轴 / 导向主轴的偏移，实际值
$VA_IS[x]	x=轴	轴的安全实际位置
$VA_VACTM[x]	x=轴	机床坐标系（MCS）中实际值侧负载侧轴速度

A.4 PI 服务列表

编程

句法	**PI_SERVICE***(服务, n 参数)*
服务	PI 服务的标识
n 参数	PI 服务的参数列表。 参数用逗号隔开。

服务	**说明**
参数	**举例**
_N_ASUP_	一种零件程序，位于 NCK 中（通过路径名称和程序名称表示），在规定的通道中分配一个中断号。 该 PI 服务与程序指令'SETINT'相同。
	Par1 中断号（0 – 8） *Par2* 优先级（0 – 8） *Par3* 快速提升（0，1） *Par4* 块同步（0，1） *Par5* 最大为 32 位的路径数据
	中断指令 5 用于当前通道中的程序 MPF_DIR/TEST_MPF。 中断有优先级 3 并且在不快速提升的情况下在轮廓上执行 *PI_SERVICE("_N_ASUP__",5, 3, 0, 0,"/_N_MPF_DIR/_N_TEST_MPF")*
_N_CANCEL	分类为"取消报警"的所有报警可以用该命令确认。 无法进行某些报警的个别确认。

	删除分类为"取消报警"的所有报警。 *PI_SERVICE("_N_CANCEL")*
_N_CRCEDN	依据刀沿号码规定设立一个刀具刀沿。 如果在 PI 服务中的参数 T 号码下已规定了一个存在的刀具 T 号码，则设立该刀具的刀沿（在这种情况下参数 D 号码 – 设立的刀沿号码 – 的值范围为 1 - 9）。 如果规定一个正的 T 号码作为参数并且不存在该规定 T 号码的刀具，则 PI 服务失败。 如果对于该 T 号码规定值为 0（绝对 D 号码的模型），则 D 号码的值范围扩展为 1 ?31999。新的刀沿通过规定的 D 号码产生。 新的刀沿通过规定的 D 号码产生。 如果已存在规定的刀沿，则在两种情况下 PI 服务失败。
	*Par1*T 号码 *Par2* D 号码 T 号== 0 ==> 1 – 31999 T 号 > 0 ==> 1 – 9
	对于在当前的 TO 范围中带有号码 17 的刀具，设立带有号码 3 的刀沿。 *PI_SERVICE("_N_CRCEDN",17,3)*

服务	**说明**
参数	**举例**
_N_CREACE	产生一个用于特定刀具的刀具刀沿。 自动产生下一个较高的空闲 D 号码。 通过该 PI 服务影响下列激活的文件系统模块： 刀具修正 TO： 设立各个刀沿（带有内容零） 监控数据 TS： （如果存在）设立各个刀沿（带有内容零） 用户刀沿数据 TUE： （如果存在）设立各个刀沿（带有内容零） (SW 版本 NCK < 10.x)
	Par1 刀具号码 1 至 31999
	在以 T 范围 1 中的号码 55 设立刀具后，还设立用于该刀具的另 2 个刀沿。 该刀具总共具有 3 个刀沿。 *PI_SERVICE("_N_CREATO",55)* *PI_SERVICE("_N_CREACE",55)* *PI_SERVICE("_N_CREACE",55)*
_N_CREATO	以某个 T 号码设立刀具。 通过该 PI 服务影响下列激活的文件系统模块： 刀具目录 TV： 刀具作为存在输入 刀具修正 TO： 设立第一个刀沿 D1 （带有内容零）。 用户刀沿数据 TUE： （如果存在）设立第一个刀沿 D1（带有内容零） 用户刀具数据 TU： （如果存在）准备一个空的数组用于该刀具
	Par1 刀具号码 1 至 31999
	在以 T 范围 1 中的号码 55 设立刀具后，还设立用于该刀具的另 2 个刀沿。 该刀具总共具有 3 个刀沿。 *PI_SERVICE("_N_CREATO",55)* *PI_SERVICE("_N_CREACE",55)* *PI_SERVICE("_N_CREACE",55)*
_N_DELECE	删除一个刀具刀沿。 如果在 PI 服务中的参数 T 号码下已规定了一个存在的刀具 T 号码，则删除针对该刀具的刀沿（在这种情况下参数 D号码 - 要删除的刀沿号码 - 的值范围为 1 - 9）。 如果规定一个正的 T 号码作为参数并且不存在该规定 T 号码的刀具，则 PI 服务失败。 如果对于该 T 号码规定值为 0 （绝对 D 号码的模型），则 D 号码的值范围扩展为 1 – 31999。新的刀沿通过规定的 D 号码产生。 如果不存在规定的刀沿，则在两种情况下 PI 服务失败。
	Par1 刀具 T 号码，该刀具应删除刀具刀沿。 值表示为 0，针对该刀具不应该存在参考（绝对 D 号码）。 *Par2* 应删除的刀具刀沿号码。 值范围： T 号 == 0 ⇒ 1 – 31999 T 号 > 0 ⇒ 1 – 9
	对于在当前的 TO 范围中带有号码 17 的刀具，删除带有号码 3 的刀沿。 *PI_SERVICE("_N_DELECE",17,3)*
_N_DELETO	删除数据模块中存储的刀具及其全部刀沿。 在下列数据模块中（如果存在），同样删除刀具： TO, TU, TUE, TV, TG (类型 400), TD, TS.
	Par1 刀具号码 1 至 31999

服务	**说明**
参数	**举例**
	在当前的 T 范围中删除带有 T 号码 21 的刀具。 *PI_SERVICE("_N_DELETO",21)*
	Par1 查找模式标识 1: 无计算搜索程序段 2: 带计算的查找 3: 在考虑主程序段情况下查找
	在当前的通道中通过计算启动查找。 要有效启动 PI 服务，必须之前满足数据段查找的数据结构（模块 SPRAF；通过变量服务在 HMI 内置上编译地址， /Channel/Search/..）。 *PI_SERVICE("_N_FINDBL",2)*
_N_LOGIN_	发送一个口令到 NCK，通过该口令设置当前的存取等级。
	*Par1*口令（正好 8 个字符，少于 8 个字符必须用空格代替）
	传送一个口令到 NCK 并以此设置一个新的存取等级。 *PI_SERVICE("_N_LOGIN_","TESTWORD")*
_N_LOGOUT	复位当前的存取等级。

	复位当前的存取等级。 *PI_SERVICE("_N_LOGOUT")*
_N_SETUFR	通过通道特定的数据模块 FU 中系统或者用户变量'linShift', 'mirrorImgActive', 'rotation'和'scaleFact'可以为每个通道定义最多 8 个零点偏移。 为了可以激活用户定义的零点偏移，必须调用 PI 服务 _N_SETUFR。

	激活一个用户框架。 *PI_SERVICE("_N_SETUFR")*

缩略语列表

B

B.1 缩略语

A	输出端
ASCII	American Standard Code for Information Interchange: 美国信息互换标准码
BAG	工作方式组
BTSS	操作面板接口
CAD	计算机辅助设计
CNC	Computerized Numerical Control（计算机数字控制）
CR	回车
DAU	数字模拟转换器
DB	PLC中数据块
DBB	PLC中数据块字节
DBW	PLC中数据块字
DBX	PLC中数据块位
DDE	Dynamic Data Exchange: 动态数据交换
DIN	德国工业标准
DIR	Directory: 目录
DPM	双端口内存
DOS	磁盘操作系统
DRAM	Dynamic Random Access Memory（动态存储器）
DRF	Differential Resolver Function: 微分旋转变压器功能（手轮）
DRY	Dry Run： 空运行进给
DW	数据字
E	输入端
EG	扩展设备
ESR	扩展的停止和退回
FRAME	数据段(框架)
FIFO	先进-先出： 数据如何保存在存储器以及如何重新调用的过程。
GP	主程序
GUD	Global User Data（全局用户数据）
HD	Hard Disk: 硬盘
HMI	Human Machine Interface: 控制系统操作区
HSA	主轴驱动
HW	硬件

开机调试	Startup
IKA	Interpolative Compensation: 可插补补偿
INC	Increment: 增量尺寸
INI	Initializing Data: 初始化数据
IPO	插补器
ISO	国际标准组织
JOG	Jogging: 手动工作方式
K1 .. K4	通道1到通道4
LED	Light Emitting Diode： 发光二极管
LF	线路馈电
K_v	回路放大系数
LUD	Local User Data: 局部用户数据
MB	兆字节
MCP	Machine Control Panel 机床控制面板 (→ MSTT)
MD	机床数据
MDA	Maual Data Automatic: 手动数据输入
MCS	机床坐标系
MLFB	机器可识别产品符
MPF	Main Program File: NC 零件程序（主程序）
MPI	Multi Port Interface: 多端口接口
MSTT	（机床控制面板）
NC	数字控制: 数字控制装置
NCK	Numerical Control Kernel: 带有程序段处理，运行范围等等的数字内核
NCU	Numerical Control Unit： NCK 硬件单元
NV	零点偏移
OEM	原装设备制造商
OP	Operation Panel： 操作面板
PCU	可编程控制单元
PCMCIA	Card International Association: 存储卡标准
PG	编程器
PLC	Programmable Logic Control:
REF	返回参考点功能
REPOS	再定位功能
ROV	Rapid Override: 输入端校正
RPA	R-Parameter Active: NCK 中的存储范围用于 R 参数编号的 R- NCK
SBL	Single Block: 单程序段
SD	设定数据
SDB	系统数据块
SEA	Setting Data Active: 设定数据符号（文件类型）
SK	软键
SKP	Skip: 跳过程序段
SPF	Sub Program File: 子程序
SRAM	静态存储器（缓存）

SUG	砂轮圆周速度
SW	软件
SYF	System Files: 系统文件
TEA	Testing Data Aktive: 机床数据标识
TO	Tool Offset: 刀具补偿
TOA	Tool Offset Active: 刀具补偿符号（文件类型）
UFR	User Frame（用户框架）
VSA	进给驱动
WCS	工件坐标系
WZK	刀具补偿
WZW	换刀
ZOA	Zero Offset Active: 零点偏移数据符号（文件类型）

词汇表

动作

所有在 →方式中设计的： →功能，→ 计算变量，→ 更改特性，...

用户变量

由用户在 → 零件程序或者数据块中定义的变量。

数组

通过数组可以归类、保存同一数据类型的数据，从而可以通过索引存取数据。

属性

属于某个对象（→ 对话框或者 → 变量）的特定 → 特性。

操作树

多个相互连接的 → 对话框

块

用于 → 设计文件的装载单元

对话框

→ 操作界面的显示

- **和对话框相关的软键栏**
  由一个新设计的对话框调用的软键栏。
- **和对话框无关的软键**
  不由对话框调用的软键，即由第一个新对话框设计的登入软键和软键栏。

定义行

定义 → 变量和 → 软键的程序部分。

编辑器

ASCII 编辑器可以通过符号在文件中输入和编辑。

属性

对象的特征（例如：→ 变量）

输入/输出栏

即 I/O 栏： 用来输入或者输出变量值。

登入软键

→ 启动第一次新建立的 → 对话框的软键。

事件

所有触发处理→方法的事件： 输入符号，按下 → 软键， ...

焦点

屏幕上的突出显示处，表示当前 → 单元，例如：光标所在处。

功能

根据 → 参数在 → 方法中编程的流程。

辅助变量

没有 → 特性并因此不显示在 → 对话框中内部计算变量。

热键

OP 010、OP 010C 和带有热键块的 SINUMERIK 键盘上的 6
个按键，按下此键直接选择一个操作区。 也可以选择 2 个其它的按键作为热键操作。

HSx

水平 → 软键 x

编译器

编译器自动将 → 设计文件中定义的代码转换成 → 对话框并控制其使用。

机床数据

由西门子 / 机床制造商 / 最终用户存放在系统中的 SINUMERIK 系统特性的设置。 它分为：

$MN_... 一般 NC 机床数据

$MC_... 通道专用机床数据

$MA_... 轴专用机床数据

$MM_... 操作面板机床数据

此外存在的 → 设定数据和驱动机床数据。

方法

如果出现相应的 → 事件，则执行编程的步骤。

HMI 高级/ HMI 内置 sl

控制系统的操作界面

NC

数字控制: 根据 → 零件程序控制轴运动过程的 SINUMERIK 系统组件。

NC 代码

建立 SINUMERIK → 零件程序允许的语言单元

NC 功能

→ PI 服务

实用注释

NC 代码生成时自动产生的注释。

参数

参数是编程语句的可改变部分，在 → 设计文件中可用另一个字 / 符号替换。

PI 服务

在 → NC 上执行固定操作的 → 功能。

PI 服务可以由 → PLC 和 → HMI 高级/HMI 内置 sl 调用。

PLC

Programmable Logic Control: 执行大部分 SINUMERIK 系统中逻辑链接的存储器可编程的控制器。

PLC 按键

PLC 按键通过 HMI 软件的 PLC 接口提供，如同热键。 在 HMI 中由它触发的功能都可以设计。

该按键可以是机床操作面板上的按键或者是 PLC 的信号连接端。 因此，它们也被称作“虚拟按键”。

编程支持

提供对话框以支持 → 建立零件程序，组件版本较高

设计文件

包含定义和指令，并确定 → 对话框的外观和 → 功能的文件。

寄存器

用于 → 对话框间数据交换的存储器。

反编译

由 → 编程支持 → 对话框的输入栏可以在 → 零件程序中产生 NC 代码段。 反编译描述相反的过程。 产生一个所选择的 NC 代码段的输入栏由 NC 代码重新恢复并显示在原来的对话框中。

设定数据

按照 NC 控制系统软件定义的方法表明机床特性的数据。 与 → 机床数据相反，可由操作人员立即更改。

ShopMill

优化的 SINUMERIK 使用和操作界面，用于 2½D 铣削。

ShopTurn

优化的 SINUMERIK 使用和操作界面，用于车削。

模拟

模仿 → 零件程序过程，但没有实际的机床轴运动。

软键

触发所属屏幕上显示的功能的操作面板按键。

软键标签

→ 软键在屏幕上的文本/图形。

软键栏

所有水平或者所有垂直的 → 软键

栏索引

数组栏索引

标准应用程序

存在的 → 标准操作界面。

系统变量

提供零件程序存取和 → HMI 存取时的 NC 状态的 NC 变量。

零件程序

规定轴的运动过程以及各种特殊动作的、以 NC 语言编制的程序。

转换栏

→ 输入 / 输出栏中的值列表；通过转换栏检查： 某栏中的输入必须与列表值中的一个相符合。

变量

通过 → 特性分配显示在 → 对话框中并可以记录到输入数据和计算结果中的存储器空间。

VSx

垂直→ 软键 x

行索引

数组行号码

存取等级

权限分级，即根据用户的不同权限而使用 → 操作界面上的功能。

索引

前

功

动

十

单

取

变

命

图

多

子