



CoDeSys 软件编程

用户指令手册 V1.0



目录

1. 标准数据类型.....	3
1.1 BOOL.....	3
1.2 整型数据类型.....	3
1.3 REAL / LREAL	3
1.4 STRING	3
1.5 时间、日期类型.....	4
1.6 常数.....	4
1.6.1 BOOL – 常数	4
1.6.2 TIME – 常数	4
1.6.3 DATE – 常数.....	4
1.6.4 TIME_OF_DAY 常数.....	4
1.6.5 DATE_AND_TIME – 常数	5
1.6.6 数值常数.....	5
1.6.7 REAL / LREAL – 常数.....	5
1.6.8 STRING – 常数	5
2. 变量类型转换功能.....	6
2.1 BOOL_TO_变换.....	6
2.2 TO_BOOL – 变换.....	6
2.3 整型数类型之间的转换.....	6
2.4 REAL_TO - / LREAL_TO – 转换.....	7
2.5 TIME_TO - / TIME_OF_DAY – 转换.....	7
2.6 DATE_TO - / DT_TO – 转换	7
2.7 STRING_TO – 转换	7
2.8 TRUNC (取整).....	8
3. 用户定义的数据类型.....	8
3.1 数组.....	8
3.2 指针.....	10
3.3 枚举.....	10
3.4 结构.....	10
3.5 参考 (别名类型).....	11
3.6 替代范围类型.....	11
4. 编程方式.....	14
4.1 指令表 IL.....	14
4.2 结构化文本 ST.....	16
4.3 功能块图 FBD.....	17
4.4 梯形图 LD	18
5. CoDeSys 中全部运算符及功能名.....	19



1. 标准数据类型

1.1 BOOL

BOOL 类型变量可取值 TRUE 和 FALSE。保留 8 位内存空间。

1.2 整型数据类型

所有的整型数据类型为：

- BYTE 字节
- WORD 字
- DWORD 双字
- SINT 短整型
- USINT 无符号短整型
- INT 整型
- UINT 无符号整型
- DINT 双精度整型
- UDINT 无符号双精度整型

各个不同的数据类型有不同的值范围。下表为整型数据的值范围和占用的内存空间

类型	下限	上限	内存空间
BYTE	0	255	8 位
WORD	0	65535	16 位
DWORD	0	4294967295	32 位
SINT	-128	127	8 位
USINT	0	255	8 位
INT	-32768	32767	16 位
UINT	0	65535	16 位
DINT	-2147483648	2147483647	32 位
UDINT	0	4294967295	32 位

当用大类型转换为小类型时，将导致丢失信息。

1.3 REAL / LREAL

REAL 和 LREAL 被称为浮点数类型。用于有理数表示。REAL 占用 32 位内存空间，LREAL 占用 64 位。

1.4 STRING

STRING 类型变量可以是包含任何字符的字符串。其容量大小在声明变量时说明，如果不对容量大小进行说明，其缺省值为 80 个字符。

字符串变量声明示例：

```
str : STRING(35) := 'This is a String';
```



1.5 时间、日期类型

TIME、TIME_OF_DAY (缩写 TOD)、DATE 和 DATE_AND_TIME (缩写 DT) 数据类型在内部作为 DWORD 处理。TIME 和 TOD 中的时间用毫秒表示，TOD 中的时间从 12:00 AM 开始。DATE 和 DT 中的时间用秒表示，并从 1970 年 1 月 1 日 12:00 AM 开始。时间数据的格式在常数一节中说明。

1.6 常数

1.6.1 BOOL - 常数

BOOL – 常数为逻辑值 TRUE 和 FALSE。

1.6.2 TIME - 常数

TIME 常数可以在 TwinCAT PLC 控制中声明。主要用于标准库中定时器的操作，格式如下：

T# xx d xx h xx m xx s xx ms

其中：T 表示时间常数起始，# 数值符号，d 天，h 小时，m 分，s 秒，ms 毫秒。

下面是 ST 中分配的正确的时间常数示例：

TIME1 := T#14ms;

TIME1 := T#100s12ms; (* 单位最大的成员允许超过其极限 *)

TIME1 := t#12h34m15s;

不正确的 TIME 常数示例：

TIME1 := t#5m68s; (*单位较小的成员超过其极限 *)

TIME1 := 15ms; (* 遗漏 T# *)

TIME1 := t#4ms13d; (* 单位顺序错 *)

1.6.3 DATE - 常数

该常数用于输入日期。DATE 常数的声明用 d、D、DATE 或 date，后接 # 构成，可以输入格式为 年-月-日的任何日期。

示例：

DATE#1996-05-06

d#1972-03-29

1.6.4 TIME_OF_DAY 常数

该常数主要用于存储一天中的时间。TIME_OF_DAY 声明用 tod#,TOD#,TIME_OF_DAY# 或 time_of_day# 后接一个时间格式：小时：分：秒。秒可以用实数表示。

示例：

TIME_OF_DAY#15:36:30.123

tod#00:00:00



1.6.5 DATE_AND_TIME - 常数

日期常数和时时间常数可以组合成所谓的 DATE_AND_TIME 常数。DATE_AND_TIME 常数用 dt#, DT#, DATE_AND_TIME# 或 date_and_time# 开始，后接日期和时间，日期和时间之间用 - 连接。

示例：

DATE_AND_TIME#1996-05-06-15:36:30

dt#1972-03-29-00:00:00

1.6.6 数值常数

数值可以用二进制、八进制、十进制和十六进制数表示。

示例：

14 (十进制数)

2#1001_0011 (二进制数)

8#67 (八进制数)

16#A (十六进制数)

这些数值可以是 BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL 或 LREAL 类型，不允许直接从“大类型”向“小类型”转换。例如，DINT 变量不能简单地以 INT 型变量使用。可以用标准库中的转换功能实现其转换。

1.6.7 REAL / LREAL - 常数

REAL 和 LREAL 常数可以用尾数和指数表示，并使用美国标准。

示例：

7.4 取代 7,4

1.64e+009 取代 1,64e+009

1.6.8 STRING - 常数

字符串是由字符组成的序列。STRING 常数使用单引号对区分。一些特殊的符号可用下表表示：

字符	说明
\$\$	美元符号
\$L 或 \$l	行给进
\$N 或 \$n	新行
\$P 或 \$p	页给进
\$R 或 \$r	行结束
\$T 或 \$t	制表
\$'	单引号

示例：



‘Your Name’

‘Susi and Claus’

‘:-) \$’

2. 变量类型转换功能

不能直接从“大类型”向“小类型”变量转换。(例如：从 INT 到 BYTE，或从 DINT 到 WORD)。要完成此功能，可以使用特殊功能块实现。作为一种规则，你可以用此功能将一种类型的变量转换成任何需要的类型变量。

句法：

<elem.Tpy1>_TO_<elem.Typ2>

2.1 BOOL_TO_变换

BOOL 类型变量到不同类型的变换：对于数值类型变量，操作数为 TRUE 时，结果为 1；操作数为 FALSE 时，结果为 0。对字符串类型变量，其结果分别为 ‘TRUE’ 和 ‘FALSE’。

ST 中的示例：

```
i:=    BOOL_TO_INT(TRUE);      (*结果为 1 *)
str:=  BOOL_TO_STRING(TRUE);  (*结果为 'TRUE' *)
t:=    BOOL_TO_TIME(TRUE);    (*结果为 T#1ms *)
tof:=  BOOL_TO_TOD(TRUE);     (*结果为 TOD#00:00:00.001 *)
dat:=  BOOL_TO_DATE(FALSE);   (*结果为 D#1970-01-01 *)
dandt:= BOOL_TO_DT(TRUE);     (*结果为 DT#1970-01-01-00:00:01 *)
```

2.2 TO_BOOL – 变换

其它类型变量到 BOOL 的转换：当操作数不为零时，结果为 TRUE，当操作数为零时，结果为 FALSE；当字符串变量的操作数为 ‘TRUE’ 时，结果为真，否则，结果为假。

ST 中的示例：

```
b := BYTE_TO_BOOL(2#11010101); (*结果为 TRUE *)
b := INT_TO_BOOL(0);           (*结果为 FALSE *)
b := TIME_TO_BOOL(T#5ms);     (*结果为 TRUE *)
b := STRING_TO_BOOL('TRUE');  (*结果为 TRUE *)
```

2.3 整型数类型之间的转换

整型数值类型到其它数值类型的转换：当从大类型向小类型转换时，存在丢失信息的危险。如果转换的数值超过其极限；则该数的第一个字节将被忽略。

ST 中的示例：

```
si := INT_TO_SINT(4223); (* 结果为 127 *)
```

如果你将整数 4223 (十六进制为 16#107f) 作为 SINT 变量保存，其结果为 127 (十六进制为 16#7f)。

IL 中的示例：

LD 2

INT_TO_REAL



MUL 3.5

2.4 REAL_TO - / LREAL_TO - 转换

REAL 或 LREAL 类型到其它数值类型的转换：数值将向上或向下取整并转换成新的数据类型。但变量类型 STRING, BOOL, REAL 和 LREAL 除外。当从大类型向小类型转换时，存在丢失信息的危险。

请注意：当向字符串变量转换时，保留 16 个数据，如果(L)REAL 数据有更多的数，则第十六个数将被取整。如果字符串的长度定义为短型，则从右端开始截取。

ST 中的示例：

i := REAL_TO_INT(1.5); (* 结果为 2 *)

j := REAL_TO_INT(1.4); (* 结果为 1 *)

IL 中的示例：

LD 2.7

REAL_TO_INT

GE %MW8

2.5 TIME_TO - / TIME_OF_DAY - 转换

TIME 或 TIME_OF_DAY 类型到其它类型的转换：时间在内部以毫秒单位及 DWORD 方式处理(对于 TIME_OF_DAY 变量，用 12:00 AM 起始)。该值将被转换。当从大类型向小类型转换时，存在丢失信息的危险。对于字符串类型变量，其结果为时间常数。

ST 中的示例：

str := TIME_TO_STRING(T#12ms); (* 结果为 'T#12ms' *)

dw := TIME_TO_DWORD(T#5m); (* 结果为 300000 *)

si := TOD_TO_SINT(TOD#00:00:00.012); (* 结果为 12 *)

2.6 DATE_TO - / DT_TO - 转换

DATE 或 DATE_AND_TIME 类型到其它类型的转换：时间在内部以 1970.01.01 开始所经过的时间，并以秒为单位及 DWORD 方式处理。该值将被转换。当从大类型向小类型转换时，存在丢失信息的危险。对于字符串类型变量，其结果为日期常数。

ST 中的示例：

b := DATE_TO_BOOL(D#1970-01-01); (* 结果为 FALSE *)

i := DATE_TO_INT(D#1970-01-15); (* 结果为 29952 *)

byt := DT_TO_BYTE(DT#1970-01-15-05:05:05); (* 结果为 129 *)

str := DT_TO_STRING(DT#1998-02-13-14:20); (* 结果为 'DT#1998-02-13-14:20' *)

2.7 STRING_TO - 转换

STRING 类型到其它类型的转换：字符串类型变量中必须包含有效的目标变量类型值，否则其转换结果为零。

ST 中的示例：

b := STRING_TO_BOOL('TRUE'); (* 结果为 TRUE *)



```
w :=STRING_TO_WORD('abc34'); (* 结果为 0 *)  
t :=STRING_TO_TIME('T#127ms'); (* 结果为 T#127ms *)
```

2.8 TRUNC (取整)

从 REAL 到 INT 类型转换。数值的所有部分都将被使用。当从大类型向小类型转换时，存在丢失信息的危险。

ST 中的示例：

```
i:=TRUNC(1.9); (* 结果为 1 *)
```

```
i:=TRUNC(-1.4); (* 结果为 1 *)
```

IL 中的示例：

```
LD 2.7
```

```
TRUNC
```

```
GE %MW8
```

3. 用户定义的数据类型

3.1 数组

支持一维、二维和三维数组的成员数据类型。数组可在 POU 的声明部分和全局变量表中定义。

语法：

```
<数组名> : ARRAY [<ll1>..
```

ll1, ll2 为数组维数的下限标识，ul1 和 ul2 为数组维数的上限标识。数值范围必须为整数。

示例：

```
Card_game: ARRAY [1..13, 1..4] OF INT;
```

数组的初始化：

可以对数组中的所有元素进行初始化，或不进行初始化。

数组初始化示例：

```
arr1 : ARRAY [1..5] OF INT := 1,2,3,4,5;
```

```
arr2 : ARRAY [1..2,3..4] OF INT := 1,3(7); (* 等同 1,7,7,7 *)
```

```
arr3 : ARRAY [1..2,2..3,3..4] OF INT := 2(0),4(4),2,3; (* 等同 0,0,4,4,4,4,2,3 *)
```

结构化中的数组初始化示例：

```
TYPE STRUCT1
```

```
STRUCT
```

```
p1:int;
```

```
p2:int;
```

```
p3:dword;
```

```
END_STRUCT
```

```
ARRAY[1..3] OF STRUCT1:= (p1:=1;p2:=10;p3:=4723),(p1:=2;p2:=0;p3:=299),
```



(p1:=14;p2:=5;p3:=112);

数组部分元素初始化示例：

arr1 : ARRAY [1..10] OF INT := 1,2;

数组中的元素如果没有初始化值，则用基本类型的缺省值初始化其值。在上例中，元素 arr1[3] 到元素 arr1[10] 均被初始化为 0。

二维数组的元素存取，使用下列语法：

<数组名>[Index1,Index2]

示例

Card_game[9,2]

注：

如果你在项目中定义了一个名为 CheckBounds 的功能，则可以自动检查数组的上下限超限错误！下图中给出了如何实现该功能的示例。

```
0001 FUNCTION CheckBounds : INT
0002 VAR_INPUT
0003     index,lower,upper:INT;
0004 END_VAR
0005
0006
0007
0008
0009
0010
0011 IF index <= lower THEN
0012     CheckBounds := lower;
0013 ELSIF index > upper THEN
0014     CheckBounds := upper;
0015 ELSE
0016     CheckBounds := index;
0017 END_IF
```

下面的示例用 CheckBounds 功能测试数组的超限问题。CheckBounds 功能允许 A[0] 到 A[7] 元素分配值 TRUE，而不会给 A[10] 分配值，这样可以避免对数组元素的错误操作。

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     A:ARRAY[0..7] OF BOOL;
0004     B:INT:=10;
0005
0006
0007
0008
0009
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
0100
0101
0102
0103
0104
0105
0106
0107
0108
0109
0110
0111
0112
0113
0114
0115
0116
0117
0118
0119
0120
0121
0122
0123
0124
0125
0126
0127
0128
0129
0130
0131
0132
0133
0134
0135
0136
0137
0138
0139
0140
0141
0142
0143
0144
0145
0146
0147
0148
0149
0150
0151
0152
0153
0154
0155
0156
0157
0158
0159
0160
0161
0162
0163
0164
0165
0166
0167
0168
0169
0170
0171
0172
0173
0174
0175
0176
0177
0178
0179
0180
0181
0182
0183
0184
0185
0186
0187
0188
0189
0190
0191
0192
0193
0194
0195
0196
0197
0198
0199
0200
0201
0202
0203
0204
0205
0206
0207
0208
0209
0210
0211
0212
0213
0214
0215
0216
0217
0218
0219
0220
0221
0222
0223
0224
0225
0226
0227
0228
0229
0230
0231
0232
0233
0234
0235
0236
0237
0238
0239
0240
0241
0242
0243
0244
0245
0246
0247
0248
0249
0250
0251
0252
0253
0254
0255
0256
0257
0258
0259
0260
0261
0262
0263
0264
0265
0266
0267
0268
0269
0270
0271
0272
0273
0274
0275
0276
0277
0278
0279
0280
0281
0282
0283
0284
0285
0286
0287
0288
0289
0290
0291
0292
0293
0294
0295
0296
0297
0298
0299
0300
0301
0302
0303
0304
0305
0306
0307
0308
0309
0310
0311
0312
0313
0314
0315
0316
0317
0318
0319
0320
0321
0322
0323
0324
0325
0326
0327
0328
0329
0330
0331
0332
0333
0334
0335
0336
0337
0338
0339
0340
0341
0342
0343
0344
0345
0346
0347
0348
0349
0350
0351
0352
0353
0354
0355
0356
0357
0358
0359
0360
0361
0362
0363
0364
0365
0366
0367
0368
0369
0370
0371
0372
0373
0374
0375
0376
0377
0378
0379
0380
0381
0382
0383
0384
0385
0386
0387
0388
0389
0390
0391
0392
0393
0394
0395
0396
0397
0398
0399
0400
0401
0402
0403
0404
0405
0406
0407
0408
0409
0410
0411
0412
0413
0414
0415
0416
0417
0418
0419
0420
0421
0422
0423
0424
0425
0426
0427
0428
0429
0430
0431
0432
0433
0434
0435
0436
0437
0438
0439
0440
0441
0442
0443
0444
0445
0446
0447
0448
0449
0450
0451
0452
0453
0454
0455
0456
0457
0458
0459
0460
0461
0462
0463
0464
0465
0466
0467
0468
0469
0470
0471
0472
0473
0474
0475
0476
0477
0478
0479
0480
0481
0482
0483
0484
0485
0486
0487
0488
0489
0490
0491
0492
0493
0494
0495
0496
0497
0498
0499
0500
0501
0502
0503
0504
0505
0506
0507
0508
0509
0510
0511
0512
0513
0514
0515
0516
0517
0518
0519
0520
0521
0522
0523
0524
0525
0526
0527
0528
0529
0530
0531
0532
0533
0534
0535
0536
0537
0538
0539
0540
0541
0542
0543
0544
0545
0546
0547
0548
0549
0550
0551
0552
0553
0554
0555
0556
0557
0558
0559
0560
0561
0562
0563
0564
0565
0566
0567
0568
0569
0570
0571
0572
0573
0574
0575
0576
0577
0578
0579
0580
0581
0582
0583
0584
0585
0586
0587
0588
0589
0590
0591
0592
0593
0594
0595
0596
0597
0598
0599
0600
0601
0602
0603
0604
0605
0606
0607
0608
0609
0610
0611
0612
0613
0614
0615
0616
0617
0618
0619
0620
0621
0622
0623
0624
0625
0626
0627
0628
0629
0630
0631
0632
0633
0634
0635
0636
0637
0638
0639
0640
0641
0642
0643
0644
0645
0646
0647
0648
0649
0650
0651
0652
0653
0654
0655
0656
0657
0658
0659
0660
0661
0662
0663
0664
0665
0666
0667
0668
0669
0670
0671
0672
0673
0674
0675
0676
0677
0678
0679
0680
0681
0682
0683
0684
0685
0686
0687
0688
0689
0690
0691
0692
0693
0694
0695
0696
0697
0698
0699
0700
0701
0702
0703
0704
0705
0706
0707
0708
0709
0710
0711
0712
0713
0714
0715
0716
0717
0718
0719
0720
0721
0722
0723
0724
0725
0726
0727
0728
0729
0730
0731
0732
0733
0734
0735
0736
0737
0738
0739
0740
0741
0742
0743
0744
0745
0746
0747
0748
0749
0750
0751
0752
0753
0754
0755
0756
0757
0758
0759
0760
0761
0762
0763
0764
0765
0766
0767
0768
0769
0770
0771
0772
0773
0774
0775
0776
0777
0778
0779
0780
0781
0782
0783
0784
0785
0786
0787
0788
0789
0790
0791
0792
0793
0794
0795
0796
0797
0798
0799
0800
0801
0802
0803
0804
0805
0806
0807
0808
0809
0810
0811
0812
0813
0814
0815
0816
0817
0818
0819
0820
0821
0822
0823
0824
0825
0826
0827
0828
0829
0830
0831
0832
0833
0834
0835
0836
0837
0838
0839
0840
0841
0842
0843
0844
0845
0846
0847
0848
0849
0850
0851
0852
0853
0854
0855
0856
0857
0858
0859
0860
0861
0862
0863
0864
0865
0866
0867
0868
0869
0870
0871
0872
0873
0874
0875
0876
0877
0878
0879
0880
0881
0882
0883
0884
0885
0886
0887
0888
0889
0890
0891
0892
0893
0894
0895
0896
0897
0898
0899
0900
0901
0902
0903
0904
0905
0906
0907
0908
0909
0910
0911
0912
0913
0914
0915
0916
0917
0918
0919
0920
0921
0922
0923
0924
0925
0926
0927
0928
0929
0930
0931
0932
0933
0934
0935
0936
0937
0938
0939
0940
0941
0942
0943
0944
0945
0946
0947
0948
0949
0950
0951
0952
0953
0954
0955
0956
0957
0958
0959
0960
0961
0962
0963
0964
0965
0966
0967
0968
0969
0970
0971
0972
0973
0974
0975
0976
0977
0978
0979
0980
0981
0982
0983
0984
0985
0986
0987
0988
0989
0990
0991
0992
0993
0994
0995
0996
0997
0998
0999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
```



3.2 指针

当程序运行时，变量或功能块地址保存在指针中。指针声明为如下句法形式：

<指针名> : POINTER TO <数据类型 / 功能块>;

指针可指向任何数据类型、功能块和用户定义的数据类型。对地址操作的 ADR 功能，可将变量或功能块的地址指向指针。指针后加内容操作符“^”可取出指针中的数据。

示例：

```
pt:POINTER TO INT;
var_int1:INT := 5;
var_int2:INT;
pt := ADR(var_int1);
var_int2:= pt^;      (* var_int2 的值为 5 *)
```

3.3 枚举

枚举为用户定义的数据类型，并由一组字符串常数组成。这些常数被视为枚举值。枚举值在项目中为全局使用的变量，即使它们在 POU 中为本地声明的变量。创建枚举变量的最好方法是在数据类型对象组织下创建。用关键字 TYPE 开始，END_TYPE 结束。

句法：

TYPE <枚举变量>:(<Enum_0> ,<Enum_1>, ...,<Enum_n>);END_TYPE

枚举变量可以取枚举值中的任何一个值。缺省情况下，第一个枚举值为零，其后依次递增。

示例：

```
TRAFFIC_SIGNAL: (Red, Yellow, Green:=10); (*每个颜色的初始值为 red 0, yellow 1, green 10*)
```

```
TRAFFIC_SIGNAL:=0; (* 交通信号值为 red*)
```

```
FOR i:= Red TO Green DO
```

```
    i := i + 1;
```

```
END_FOR;
```

不能对同一个枚举值多次使用。

示例：

```
TRAFFIC_SIGNAL: (red, yellow, green);
```

```
COLOR: (blue, white, red);
```

错误：red 不能对 TRAFFIC_SIGNAL 和 COLOR 变量同时使用。

3.4 结构

结构作为对象在数据类型页中创建。使用 TYPE 关键字开始，END_TYPE 关键字结束。结构声明的句法如下：

TYPE <结构名>:

```
    STRUCT
```

```
    <声明变量 1>
```

```
    .
```



<声明变量 n>

END_STRUCT

END_TYPE

<结构名>是一种类型，在项目中为全程识别，并且可作为标准数据类型使用。允许内嵌结构。唯一的限制是变量不能带地址(不允许用 AT 声明!)

下例为多边形的结构示例：

TYPE Polygonline:

STRUCT

Start: ARRAY [1..2] OF INT;

Point1: ARRAY [1..2] OF INT;

Point2: ARRAY [1..2] OF INT;

Point3: ARRAY [1..2] OF INT;

Point4: ARRAY [1..2] OF INT;

End: ARRAY [1..2] OF INT;

END_STRUCT

END_TYPE

可以使用下面的句法存取结构中的成员。

<结构_名>.<成员名>

例如：结构名为“Week”，其中包含一个成员“Monday”，可以用 Week.Monday 获取该值。

3.5 参考 (别名类型)

可以使用用户定义的参考数据类型，创建已经更名的变量、常数或功能块。在数据类型页中创建参考对象。使用 TYPE 关键字开始，END_TYPE 关键字结束。

句法：

TYPE <标识符>: <分配 项>;

END_TYPE

示例：

TYPE message:STRING[50];

END_TYPE;

3.6 替代范围类型

替代范围类型，是对其基本数据类型重新设置范围的一种数据类型。声明可以在数据类型页中进行，但变量也可直接用子范围类型声明：

在数据类型页中声明的句法如下：

TYPE <Name> : <Inttype> (<ug>..<og>) END_TYPE;

类型	说明
<Name>	必须为有效的 IEC 标识符
<Inttype>	数据类型中的一种。如 SINT,USINT,INT,DINT,UDINT,BYTE,WORD, DWORD(LINT,UINT,LWORD).



ABB(中国)有限公司 武汉分公司

地址：中国湖北省武汉市武昌中南路7号3408室

电话：(027)87259222 传真：(027)87259233

手机：15927508200 直拨：(027)87259230

<ug>	常数，必须为基本类型，设定的下边界在其类型范围之内。
<og>	常数，必须为基本类型，设定的上边界在其类型范围之内。

示例：

TYPE

SubInt : INT (-4095..4095);

END_TYPE

用子范围类型直接声明的变量：

VAR

i1 : INT (-4095..4095);

i2 : INT (5...10):=5;

ui : UINT (0..10000);

END_VAR

如果常数被分配为一个子范围类型(在声明或实现段中)，但其值没有落在该范围之内(例如 $i := 5000$)，系统将会发出错误信息。

为了在运行期间检查边界范围，推荐使用功能 `CheckRangeSigned` 或 `CheckRangeUnsigned`。这样，边界有效性验证可通过合适的方法和手段捕获(例：数值可以截取或设置错误标志)。

示例：

当变量属于有符号子范围类型时(如上例中的 i)，则功能 `CheckRangeSigned` 被调用；可以通过编程的方法使其值在允许范围之内。

FUNCTION `CheckRangeSigned` : DINT

VAR_INPUT

value, lower, upper: DINT;

END_VAR

IF (value < lower) THEN

`CheckRangeSigned` := lower;

ELSIF(value > upper) THEN

`CheckRangeSigned` := upper;

ELSE

`CheckRangeSigned` := value;

END_IF

为了自动调用功能，功能名 `CheckRangeSigned` 被指定，并且接口也被指定：返回值和三个 DINT 类型的参数。

当调用时，功能参数如下：

值	分配给范围类型的值
下限	下限边界范围
上限	上限边界范围
返回值	实际分配给范围类型的值



对 $i := 10 * y$ 进行边界有效性验证的示例：

```
i := CheckRangeSigned(10 * y, -4095, 4095);
```

示例中，y 即使是 1000，i 经过上例赋值后其值仍然为 4095。

同样，功能 CheckRangeUnsigned 过程同上：功能名和接口必须正确。

```
FUNCTION CheckRangeUnsigned : UDINT
```

```
VAR_INPUT
```

```
    value, lower, upper: UDINT;
```

```
END_VAR
```

注意：

如果没有 CheckRangeSigned 和 CheckRangeUnsigned，则运行时，没有子类型的类型检验发生，变量 i 可以在 -32768 和 32767 之间取任何值。

注意：

如果功能 CheckRangeSigned 和 CheckRangeUnsigned 按照上例实现，则在 FOR 循环中可对子范围类型连续使用循环。

示例：

```
VAR
```

```
    ui : UINT (0..10000);
```

```
END_VAR
```

```
FOR ui:=0 TO 10000 DO
```

```
...
```

```
END_FOR
```

FOR 循环不会剩余，因为 ui 不会大于 10000。

象 CheckRange 功能内容一样，当在 FOR 循环中使用增量值时，也应考虑这些问题。



4. 编程方式

4.1 指令表 IL

指令表(IL)由一系列指令组成。每条指令都由一个标号开始，包含一个操作符以及和操作符类型相关的一个或多个操作数，并用逗号分开。在指令前可以有标号，后接一个冒号。

注解必须在一行的最后，指令之间可以插入空行。

示例

标号	操作符	操作数	注解
	LD	17	
	ST	lint	(* comment *)
	GE	5	
	JMPC	next	
	LD	idword	
	EQ	istruct.sdword	
	STN	test	
next:			

在 IL 语言中，可以使用下面的操作符和修饰符。

修饰符：

- JMP、CAL、RET 中带 C：指令在预置表达式结果为 TRUE 时执行。
- JMPC、CALC、RETC 中带 N：指令在预置表达式结果为 FALSE 时执行。
- 其它指令中带 N：操作数取反 (不是累加器)。

下表为 IL 中全部的操作符及可能的修饰符和相关的意义：

操作符	修饰符	意义
LD	N	使当前结果等于操作数
ST	N	在操作数位置保存当前结果
S		如果当前结果为 TRUE，置位布尔操作数为 TRUE
R		如果当前结果为 TRUE，复位布尔操作数为 FALSE
AND	N,(位与
OR	N,(位或
XOR	(位异或
ADD	(加
SUB	(减
MUL	(乘
DIV	(除
GT	(>
EQ	(=
NE	(<>
LE	(<=
LT	(<
JMP	CN	跳转到标号
CAL	CN	调用功能块
RET	CN	从调用的功能块返回
)		评估括号操作



ABB(中国)有限公司 武汉分公司

地址：中国湖北省武汉市武昌中南路7号 3408室

电话：(027)87259222 传真：(027)87259233

手机：15927508200 直拨：(027)87259230

IL 是一种面向行的语言。

标号	:	操作符/功能	操作数(表)	注释
跳转标号	分隔符	IL 操作符或功能名	用于操作符的零个，一个或多个常数、变量，或用于功能的输入参数，由逗号分隔。	在(*...*)中的注释，可选

通过不同的操作符组修改 CR

影响 CR 数据类型的操作符组	缩写	操作符示例
Create (建立)	C	LD
Process (处理)	P	GT
Leave unchanged (保持不变)	U	ST: JMPC
Set to undefined (设置为未定义的)	-	CAL = 功能块的无条件调用，

带布尔操作数(BOOL 类型)的操作符

操作符		操作符组	描述
LD	LDN	C	装入操作数(操作数的反值)到 CR
AND AND(ANDN ANDN(P	操作数(操作数的反值)和 CR 的布尔 AND (“与”运算)
OR OR(ORN ORN(P	操作数(操作数的反值)和 CR 的布尔 OR (“或”运算)
XOR XOR(XORN XORN(P	操作数(操作数的反值)和 CR 的布尔 XOR (“异或”运算)
ST	STN	U	将 CR 存到操作数
S		U	若 CR = 1，则将操作数设置为 TRUE
R		U	若 CR = 1，则将操作数设置为 FALSE
)		U	结束括号：对递延操作求值

用于类属数据类型(类型 ANY) 操作数的操作符

操作符		操作符组	描述
LD		C	操作数装入 CR
ST		U	将 CR 存储到操作数
ADD	ADD(P	加操作数，结果存入 CR
SUB	SUB(P	从 CR 减去操作数，结果存入 CR
MUL	MUL(P	操作数乘以 CR
DIV	DIV(P	CR 除以操作数
GT	GT(P	CR > 操作数 (大于)
GE	GE(P	CR >= 操作数 (大于或等于)
EQ	EQ(P	CR = 操作数 (等于)
NE	NE(P	CR <> 操作数 (不等于)
LE	LE(P	CR <= 操作数 (小于或等于)
LT	LT(P	CR < 操作数 (小于)
)		U	结束括号级



ABB(中国)有限公司 武汉分公司

地址：中国湖北省武汉市武昌中南路7号3408室

电话：(027)87259222 传真：(027)87259233

手机：15927508200 直拨：(027)87259230

跳转或调用

操作符		操作符组	描述
JMP		- 或 U	(无)/有条件跳转到一个跳转标号
JMPC	JMPCN	U	
CAL		- 或 U	(无)/有条件调用一个功能块
CALC	CALCN	U	
RET		- 或 U	(无)/有条件从一个功能或功能块返回
RETC	RETCN	U	
功能名		P	功能调用

使用功能和功能块

A. 调用一个功能

在 IL 语言中，调用一个功能只是简单地写入该功能名即可。随后的实际参数用逗号分隔。这种语法和带有几个操作数的操作符的语法相同。

功能的第一个参数是当前结果(CR)。因此必须正好在功能调用之前将该值装入 CR 中。

用于功能调用的第一个操作数实际上是功能的第二个参数，并依次类推。

B. 调用一个功能块

操作符 CAL (或条件调用 CALC 和条件取反调用 CALCN)可以激活一个功能块。

IEC61131-3 描述 IL 语言中给一个 FB 传送参数的三种方法：

- 1). 使用一个调用，它包括在括号内的实际输入和输出参数的一个列表
- 2). 在调用 FB 前，装载和保存输入参数
- 3). 用输入参数作为操作符“隐性地”调用

第三种方法只对标准 FB 有效，不适合用户定义的 FB。

4.2 结构化文本 ST

ST 语言的优点 (与 IL 语言相比较)：

- 编程任务高度压缩化的表达格式，
- 在语句块中清晰的程序结构，
- 控制命令流的强有力结构

这些优点亦带来其本身的缺陷：

- 由于它借助于编译程序自动地执行程序，因此用户不能直接影响其翻译成机器码。
- 高度抽象导致效率降低(通常，编译程序的时间更长且执行速度更慢)

ST 语句

关键字	说明	示例	说明
:=	赋值	d := 10	将右边的一个供计算的数值赋值给左边的标识符
	调用 FB	FB Name(Par1 := 10, Par2 := 20);	调用另一个类型为 FB 的 POU，包括其参数
RETURN	返回	RETURN	脱离当前的 POU 和返回到调用



ABB(中国)有限公司 武汉分公司

地址：中国湖北省武汉市武昌中南路7号3408室

电话：(027)87259222 传真：(027)87259233

手机：15927508200 直拨：(027)87259230

			POU
IF	选择	IF d < e THEN f := 1; ELSEIF d = e THEN f := 2; ELSE f := 3; END_IF	通过布尔表达式选择替代值
CASE	多重选择	CASE f OF 1: g := 11; 2: g := 12; ELSE g := FunName(); END_CASE	根据表达式“F”的值选择一个语句块
FOR	迭代 (1)	FOR h:=1 TO 10 BY 2 DO F[h/2] := h; END_FOR	一个多循环语句块，带有起始和结束条件以及一个增量值
WHILE	迭代 (2)	WHILE m > 1 DO N := n / 2; END_WHILE	一个多循环语句块，具有在开始端的结束条件
REPEAT	迭代 (3)		一个多循环语句块，具有在结束端的结束条件
EXIT	循环的结束	EXIT;	一个迭代语句的结束条件。
;	空白语句	::	

ST 语言不包括跳转指令 (GOTO)。

4.3 功能块图 FBD

功能块图(FBD)语言起源于信号处理领域，对信号处理而言，整数与/或浮点数是很重要的。

使用图形化语言 FBD 或 LD 的 POU 表达式包括的部分与文本化语言相同。

- 1). POU 的引导部分和结束部分
- 2). 说明部分
- 3). 代码部分

代码部分 分为若干个网络。网络有助于构造 POU 的控制流。

一个网络包括

- 1). 网络标号
- 2). 网络注释
- 3). 网络图形



ABB(中国)有限公司 武汉分公司

地址：中国湖北省武汉市武昌中南路7号3408室

电话：(027)87259222 传真：(027)87259233

手机：15927508200 直拨：(027)87259230

4.4 梯形图 LD

梯形图语言 (LD) 源自机电一体化的继电器系统的应用领域，它描述一个 POU 的网络自左至右的能量流。编程语言主要是设计用于处理布尔信号。

梯形图 LD 接点分类：

常开接点	常闭接点	上升沿接点	下降沿接点
------	------	-------	-------

梯形图 LD 线圈分类：

线圈 --()--	线圈的取反 --(/)--	置位 (锁存) 线圈 --(S)--	复位(解除锁存)线圈 --(R)--
保持 (记忆) 线圈 --(M)--	置位保持(记忆)线圈 --(SM)--	复位保持(记忆)线圈 --(RM)--	
上升沿线圈 --(P)--	下降沿线圈 --(N)--		

梯形图 LD 执行控制分类：

无条件返回	条件返回	无条件跳转	条件跳转
-------	------	-------	------

调用功能和功能块



ABB(中国)有限公司 武汉分公司

地址：中国湖北省武汉市武昌中南路7号 3408室

电话：(027)87259222 传真：(027)87259233

手机：15927508200 直拨：(027)87259230

5. CoDeSys 中全部运算符及功能名

ST 中操作符	IL 中操作符	IL 中的修饰符	意义
'			字符串分界符(例如: 'string1')
[..]			数组大小范围(例如: ARRAY[0..3] OF INT)
:			操作数和类型声明之间的分界符(例如: var1 : INT;)
^			指针引用 (例如: pointer1^)
	LD var1	N	装入 var1 值到缓冲器中
:=	ST var1	N	存入实际结果到 var1 中
	S boolvar		当实际结果为 TRUE 时, 设置布尔变量 boolvar 为 TRUE
	R boolvar		当实际结果为 TRUE 时, 设置布尔变量 boolvar 为 FALSE
	JMP marke	CN	跳转到标号
<程序名>	CAL prog1	CN	调用程序 prog1
<句柄名>	CAL inst1	CN	调用功能块句柄 inst1
<功能名>(vx,vy,...)	<功能名>(vx,vy,...)	CN	调用功能 fctname 并传送变量 vx,vy
RETURN	RET	CN	离开 POU 并返回到调用者
	(括号之后的值作为操作数处理, 不执行括号之前的运算。
)		执行括号返回的操作运算
AND	AND	N, (位与
OR	OR	N, (位或
XOR	XOR	N, (位异或
NOT	NOT		位取反
+	ADD	(加
-	SUB	(减
*	MUL	(乘
/	DIV	(除
>	GT	(大于
>=	GE	(大于或等于
=	EQ	(等于
<	LT	(小于
<>	NE	(不等于
<=	LE	(小于或等于
MOD(in)	MOD		取模除
INDEXOF(in)	INDEXOF		POU 内部索引 in1; [INT]
SIZEOF(in)	SIZEOF		数据类型 in 所需字节数
SHL(K,in)	SHL		in 数据向左位移 K 位
SHR(K,in)	SHR		in 数据向右位移 K 位
ROL(K,in)	ROL		in 数据向左循环位移 K 位
ROR(K,in)	ROR		in 数据向右循环位移 K 位
SEL(G,in0,in1)	SEL		选择器, G 为 FALSE 选 in0 G 为 TRUE 选 in1



ABB(中国)有限公司 武汉分公司

地址：中国湖北省武汉市武昌中南路7号 3408 室

电话：(027)87259222 传真：(027)87259233

手机：15927508200 直拨：(027)87259230

MAX(in0,in1)	MAX		取极大值
MIN(in0,in1)	MIN		取极小值
LIMIT(Min,in,Max)	LIMIT		取限幅值，当 in 超过限幅值时，取 Min 或 Max 值
MUX(K,in0,..in_n)	MUX		多值选择器 (in0,..in_n)
ADR(in)	ADR		取操作数的地址到 [DWORD] 中
BOOL_TO_<type>(in)	BOOL_TO_<type>		布尔操作数类型转换
<type>_TO_BOOL(in)	<type>_TO_BOOL		类型转换到布尔值
INT_TO_<type>(in)	INT_TO_<type>		INT 转换为其他成员类型
REAL_TO_<type>(in)	REAL_TO_<type>		REAL 转换为其他成员类型
LREAL_TO_<type>(in)	LREAL_TO_<type>		LREAL 转换位其他成员类型
TIME_TO_<type>(in)	TIME_TO_<type>		TIME 转换为其他成员类型
TOD_TO_<type>(in)	TOD_TO_<type>		TOD 转换为其他成员类型
DATE_TO_<type>(in)	DATE_TO_<type>		DATE 转换为其他成员类型
DT_TO_<type>(in)	DT_TO_<type>		DT 转换为其他成员类型
STRING_TO_<type>(in)	STRING_TO_<type>		STRING 转换为其他成员类型
TRUNC(in)	TRUNC		REAL 向 INT 转换
ABS(in)	ABS		in 操作数取绝对值
SQRT(in)	SQRT		in 操作数取平方根
LN(in)	LN		in 操作数取自然对数
LOG(in)	LOG		in 操作数取底数为 10 的对数
EXP(in)	EXP		in 操作数进行指数运算 (e^x)
SIN(in)	SIN		in 操作数进行正弦运算
COS(in)	COS		in 操作数进行余弦运算
TAN(in)	TAN		in 操作数进行正切运算
ASIN(in)	ASIN		in 操作数进行反正弦运算
ACOS(in)	ACOS		in 操作数进行反余弦运算
ATAN(in)	ATAN		in 操作数进行反正切运算
EXPT(in,expt)	EXPT expt		in 为底数，expt 为指数运算
LEN(in)	LEN		in 操作数取字符串长度
LEFT(str,size)	LEFT		从 str 左边取 size 个字符串
RIGHT(str,size)	RIGHT		从 str 右边取 size 个字符串
MID(str,size,pos)	MID		从 str 的 pos 位置取 size 个字符串
CONCAT(str1,str2)	CONCAT		合并 str1 和 str2 两个字符串
INSERT(str1,str2, pos)	INSERT		在 str2 的 pos 处插入 str1 字符串
DELETE(str1,len, pos)	DELETE		在 str1 的 pos 处删除 len 个字符串
REPLACE(str1,str2 len,pos)	REPLACE		在 str1 的 pos 处替换 str2 中的 len 个



ABB(中国)有限公司 武汉分公司

地址：中国湖北省武汉市武昌中南路7号3408室

电话：(027)87259222 传真：(027)87259233

手机：15927508200 直拨：(027)87259230

			字符串
FIND(str1,str2)	FIND		在 str1 中寻找 str2 字符串
SR	SR		置位优先的 SR 触发器
RS	RS		复位优先的 RS 触发器
SEMA	SEMA		软件信号器(可中断)
R_TRIG	R_TRIG		上升沿检测
F_TRIG	F_TRIG		下降沿检测
CTU	CTU		向上计数
CTD	CTD		向下计数
CTUD	CTUD		向上和向下计数
TP	TP		脉冲计时器
TON	TON		延时开计时器
TOF	TOF		延时断计时器