Honeywell

Actors Manual

SW09-655

Total Plant

Honeywell

Implementation

Engineering Operations - 2

Actors Manual

SW09-655

Release 620

12/00

TotalPlant

Notices and Trademarks

Copyright 2000 by Honeywell International Inc.
Revision 3 December 1, 2000

While this information is presented in good faith and believed to be accurate, Honeywell disclaims the implied warranties of merchantability and fitness for a particular purpose and makes no express warranties except as may be stated in its written agreement with and for its customers.

In no event is Honeywell liable to anyone for any indirect, special or consequential damages. The information and specifications in this document are subject to change without notice.

Honeywell and **TotalPlant** are registered trademarks of Honeywell International Inc.

Other brand or product names are trademarks of their respective owners.

Honeywell
Industrial Automation and Control
16404 North Black Canyon Highway
Phoenix, AZ 85053
1-800 343-0228

About This Publication

This publication provides reference information that helps you understand how to use actor functions of the Picture Editor.

Is this the right publication?

The *Picture Editor Reference Manual* and the Button Configuration manuals will help you understand the situations that call for actors. This manual explains the details of the actors.

What should I be familiar with before using this publication?

The content of these publications:

Picture Editor Reference Manual, in the *Implementation/Engineering Operations - 2* binder.

Button Configuration Data Entry in the *Implementation/Engineering Operations -1* binder.

Implementation Overview in the *Implementation/Startup & Reconfiguration - 1* binder.

Configuration Data Collection Guide, in the *Implementation/Startup & Reconfiguration - 2* binder.

This publication supports **TotalPlant** Solution (TPS) system network through Release 620. TPS is the evolution of TDC 3000^{X}.

Table of Contents

1 INTRODUCTION TO ACTORS 1

1.1 ACTORS DEFINED 1

1.2 USE OF ACTORS 2

1.3 DISPLAY DATABASE 2

1.4 Examples 3

1.5 ACTOR DATA ENTRY 8

1.5.1 Calling Sequence 9

1.6 ACTOR ERROR HANDLING 10

1.7 SCREEN REGIONS 11

1.8 TESTING ACTOR OPERATION 12

1.9 REFERENCES 12

2 DISPLAY ACTORS 13

2.1 CALL UP THE ALARM ANNUNCIATOR DISPLAY 13

2.2 CALL UP THE ALARM SUMMARY DISPLAY 13

2.3 CALL UP THE ATTRIBUTE POINT SUMMARY DISPLAY 13

2.4 CALL UP BACKGROUND OVERLAY 14

2.5 CALL UP THE BOX POINT SUMMARY DISPLAY 15

2.6 CALL UP THE BOX STATUS DISPLAY 16

2.7 CALL UP THE CUSTOM SAVE DISPLAY 16

2.8 CALL UP CHANGE ZONE (STANDARD) 16

2.9 CALL UP CHANGE ZONE (ENHANCED) 26

2.9.1 Mini-Trend Graph 34

2.9.2 Mini-Trend Graph Scaling 38

2.9.3 Special Cases of Mini-Trend Graph 39

2.9.4 Scenarios of PVEUHI/PVEULO and SPEUHI/SPEULO Values 42

2.10 REPLACING STANDARD CHANGE ZONE WITH ENHANCED CHANGE ZONE 42

2.11 CALL UP THE DETAIL DISPLAY 43

2.12 CALL UP THE DETAIL DISPLAY, PAGE N 44

2.13 CALL UP THE EVENT HISTORY DISPLAY 44

2.14 CALL UP THE FLOPPY INITIALIZATION DISPLAY 45

2.15 CALL UP THE GROUP DISPLAY 45

2.16 CALL UP THE GROUP EDIT DISPLAY 45

2.17 CALL UP GROUP MODULE DISPLAY 46

2.18 CALL UP GROUP TREND DISPLAY 46

2.19 CALL UP THE HARDWARE POINT SUMMARY DISPLAY 47

Table of Contents

2.20 CALL UP THE HIWAY STATUS DISPLAY 47
2.21 CALL UP THE LOGIC MANAGER DIAGNOSTIC DISPLAY 48
2.22 CALL UP THE MESSAGE SUMMARY DISPLAY 48
2.23 CALL UP MULTIPLE SERIAL OVERLAYS....49
2.24 CALL UP THE NIM DIAGNOSTIC DISPLAY 50
2.25 CALL UP THE NODE POINT SUMMARY DISPLAY 51
2.26 CALL UP THE ORGANIZATIONAL SUMMARY MENU 51
2.27 CALL UP OVERLAY....51
2.28 CALL UP OVERVIEW DISPLAY 52
2.29 CALL UP THE OVERVIEW EDIT DISPLAY....52
2.30 CALL UP THE PROCESS MANAGER MODULE/NETWORK INTERFACE MODULE DIAGNOSTIC DISPLAY 53
2.31 CALL UP THE PROCESS MANAGER STATUS DISPLAY 54
2.32 CALL UP THE PROCESS MODULE DISPLAY 55
2.33 CALL UP THE PROCESS VARIABLE RETRIEVAL DISPLAY 55
2.34 CALL UP THE REAL TIME JOURNAL DISPLAY 55
2.35 CALL UP THE REPORT MENU....55
2.36 CALL UP SCHEMATIC 56
2.37 CALL UP THE TITLE SUMMARY DISPLAY 56
2.38 CALL UP SYSTEM STATUS JOURNAL (R600)....56
2.39 CALL UP THE TREND OVERVIEW DISPLAY 57
2.40 CALL UP THE UCN IO MODULE DIAGNOSTIC DISPLAY....58
2.41 CALL UP THE UCN STATUS DISPLAY 58
2.42 CALL UP THE UNIT POINT SUMMARY DISPLAY 59
2.43 CALL UP UNIT (MODULE) SUMMARY DISPLAY....59
2.44 CALL UP UNIT TREND DISPLAY 59
2.45 CALL UP THE USAGE POINT SUMMARY DISPLAY 60
2.46 CANCEL PRINT 60
2.47 CLEAR THE CHANGE ZONE....60
2.48 CLEAR PROMPT REGION....61
2.49 CLEAR THE SCREEN 61
2.50 COLLECT HISTORY....61
2.51 CROSS SCREEN (CONDITIONING)....62
2.52 DEMAND UPDATE DISPLAY 63
2.53 DISPLAY FILE....63
2.54 FROM SCREEN (DISPLAY TRANSFER)....64

Table of Contents

2.55 OUTPUT A PROMPT MESSAGE 65
2.56 PALETTE 65
2.57 PRINT FILE 66
2.58 UPDATE THE DISPLAY 67
2.59 USER CHANGE ZONE 68

3 KEYBOARD EVENT ACTIONS 69
3.1 CROSS KEYBOARD 69
3.2 Enter 73
3.3 ENTER TO EXECUTE 73
3.4 GET PRIMMOD 74
3.5 KEYLEVEL CHANGE 75
3.5.1 Hardware and Software Keylevels 76
3.6 KEYLEVEL RESET 77
3.7 KEY SIMULATION 78
3.8 MOVE THE CURSOR 79
3.9 MULTIPLE REMOTE KEYSTROKES 80
3.10 QUEUE KEY 81
3.11 REMOTE MOVE 82
3.12 STORE PRIMMOD 83

4 OPERATOR INPUT ACTORS 85
4.1 READ BOOLEAN DATA 86
4.2 READ DATE 87
4.3 READ DURATION 88
4.4 READ TIME 89
4.5 READ ENTITY 90
4.6 READ ENTITY NAME 91
4.7 READ ENUMERATION 92
4.8 READ INTEGER DATA 93
4.9 READ PARAMETER ID 94
4.10 READ REAL DATA 95
4.11 READ STANDARD ENUMERATION DATA 96
4.12 READ STANDARD ENUMERATION NAME 97
4.13 READ STRING DATA 98
4.14 READ VARIABLE 99

Table of Contents

5 STORE INTO THE DISPLAY DATABASE 101

5.1 STORE BOOLEAN DATA INTO THE DISPLAY DATABASE 101
5.2 STORE DATE INTO THE DISPLAY DATABASE 102
5.3 STORE DURATION INTO THE DISPLAY DATABASE 103
5.4 STORE TIME INTO THE DISPLAY DATABASE 104
5.5 STORE ENTITY ID INTO THE DISPLAY DATABASE 105
5.6 EQUIPMENT LIST INITIALIZE 106
5.7 STORE INTEGER DATA INTO THE DISPLAY DATABASE 108
5.8 STORE REAL DATA INTO THE DISPLAY DATABASE 108
5.9 STORE STANDARD ENUMERATION DATA INTO THE DISPLAY DATABASE 109
5.10 STORE STRING DATA INTO THE DISPLAY DATABASE 110
5.11 STORE VARIABLE ID INTO THE DISPLAY DATABASE 110

6 READ AND STORE OPERATOR INPUT DATA INTO DISPLAY DATABASE 111

6.1 READ AND STORE INTO LOCAL DISPLAY DATABASE 111

7 READ DATA FROM DISPLAY DATABASE 113

7.1 GET BOOLEAN DATA FROM DISPLAY DATABASE 113
7.2 GET DATE/TIME DATA FROM DISPLAY DATABASE 114
7.3 GET ENTITY ID FROM DISPLAY DATABASE 114
7.4 GET INTEGER DATA FROM DISPLAY DATABASE 115
7.5 GET REAL DATA FROM DISPLAY DATABASE 115
7.6 GET STANDARD ENUMERATION DATA FROM DISPLAY DATABASE 116
7.7 GET STRING DATA FROM DISPLAY DATABASE 116
7.8 GET VARIABLE ID FROM DISPLAY DATABASE 117

8 STORE INTO THE SYSTEM DATABASE 119

8.1 STORE BOOLEAN DATA INTO THE SYSTEM DATABASE 119
8.2 STORE DATE INTO THE SYSTEM DATABASE 120
8.3 STORE DURATION DATA INTO THE SYSTEM DATABASE 121
8.4 STORE TIME INTO THE SYSTEM DATABASE 122
8.5 STORE ENTITY ID INTO THE SYSTEM DATABASE 123
8.6 STORE ENUMERATION DATA INTO THE SYSTEM DATABASE 123
8.7 STORE INTEGER DATA INTO THE SYSTEM DATABASE 124
8.8 STORE REAL DATA INTO THE SYSTEM DATABASE 124
8.9 STORE STRING DATA INTO THE SYSTEM DATABASE 125

Table of Contents

9 READ OPERATOR INPUT DATA AND STORE INTO SYSTEM DATABASE.....127

9.1 READ DATA AND STORE VARIABLE 127

10 READ DATA FROM THE SYSTEM DATABASE.....129

10.1 GET BOOLEAN DATA FROM THE SYSTEM DATABASE 129
10.2 GET DATE/TIME FROM THE SYSTEM DATABASE 130
10.3 GET ENTITY ID FROM THE SYSTEM DATABASE.....130
10.4 GET ENUMERATION FROM THE SYSTEM DATABASE.....131
10.5 GET INTEGER DATA FROM THE SYSTEM DATABASE.....131
10.6 GET REAL DATA FROM THE SYSTEM DATABASE 132
10.7 GET STRING DATA FROM THE SYSTEM DATABASE 132
10.8 GET SYSTEM VARIABLE FROM THE SYSTEM DATABASE.....133
10.9 GET SYSTEM VARIABLE STRING FROM THE SYSTEM DATABASE 134

11 RANGE CHECKING ACTORS.....135

11.1 RANGE CHECK INTEGER DATA 136
11.2 RANGE CHECK LOCAL DDB DATA.....137
11.3 RANGE CHECK REAL DATA 138
11.4 RANGE CHECK STRING DATA.....139
11.5 RANGE CHECK VARIABLE ID DATA 140

12 GENERAL ACTORS 141

12.1 AREA CHANGE 141
12.2 CHECK FOR VARIABLES WITH BAD VALUES (R600) 142
12.3 CHECK FOR NON-EXISTENT VARIABLES (R600) 143
12.4 COMBINE DATE/TIME 144
12.5 COMBINE VARIABLE (ENUMERATION INDEX) 145
12.6 COMBINE VARIABLE ID.....146
12.7 CONCATENATE STRINGS 147
12.8 DELAY.....148
12.9 EXTRACT ENTITY NAME 149
12.10 EXTRACT NETWORK ID.....149
12.11 GET CONSOLE NUMBER.....150
12.12 GET NODE NUMBER 150
12.13 GET PRIMMOD COUNT.....151
12.14 GET PRIMMOD STATUS 152

Table of Contents

12.15 GET STATION NUMBER 152
12.16 NO OPERATION 153
12.17 OPTION 153

13 TREND ACTORS 155
13.1 ADD TRACE TO TREND 155
13.2 CENTERLINE TRENDING 157
13.3 DEFINE DATA SOURCE FOR TREND TRACE 159
13.4 DELETE TRACE FROM TREND 160
13.5 TREND MODE 160
13.6 HAIR LINE CURSOR MOVE 162
13.7 SAVE TREND SETTINGS 163
13.9 SET RANGE FOR TREND TRACE 165
13.10 SET SCALE FOR TREND 166
13.11 SET SCROLL FOR TREND 167
13.12 SET TIME BASE FOR TREND 168
13.13 TREND FUNCTIONALITY 170

14 ARITHMETIC ACTORS 173
14.1 ADD INTEGERS 173
14.2 SUBTRACT INTEGERS 173
14.3 MULTIPLY INTEGERS 174
14.4 DIVIDE INTEGERS 174
14.5 MODULO INTEGERS 175
14.6 NEGATE INTEGER 175
14.7 ADD REALS 176
14.8 SUBTRACT REALS 176
14.9 MULTIPLY REALS 177
14.10 DIVIDE REALS 177
14.11 NEGATE REAL 178

15 LOGICAL ACTORS 179
15.1 AND 179
15.2 OR 179
15.3 XOR (EXCLUSIVE OR) 180
15.4 NOT 180

Table of Contents

16 COMPARE ACTORS 181

16.1 COMPARE INTEGERS 181

16.2 COMPARE REALS 182

16.3 COMPARE STRINGS 183

16.4 COMPARE ENUMERATIONS 184

17 CONDITIONAL ACTORS 185

17.1 IF 186

17.2 ELSE 186

17.3 ENDIF 186

18 CONVERSION ACTORS 189

18.1 INTEGER TO REAL CONVERSION 189

18.2 INTEGER TO STRING CONVERSION 190

18.3 INTERNAL DISPLAY DATABASE ENTITY ID TO EXTERNAL ENTITY ID 191

18.4 INTERNAL DISPLAY DATABASE VARIABLE ID TO EXTERNAL ENTITY ID 191

18.5 INTERNAL DISPLAY DATABASE VARIABLE ID TO EXTERNAL PARAMETER ID 192

18.6 INTERNAL DISPLAY DATABASE VARIABLE ID TO EXTERNAL VARIABLE ID 192

18.7 REAL TO INTEGER CONVERSION BY ROUNDING 193

18.8 REAL TO INTEGER CONVERSION BY TRUNCATION 193

18.9 STRING TO INTERNAL ENTITY ID CONVERSION 194

18.10 UNIT ID TO UNIT NUMBER 195

19 QUERY ACTORS 197

19.1 INVOKE QUERY WITH DISPLAY OUTPUT 198

19.2 INVOKE QUERY WITH FILE OUTPUT 199

19.3 INVOKE QUERY WITH PRINTER OUTPUT 201

19.4 CANCEL QUERY 202

19.5 CLEAR QUERY STATUS 202

19.6 DELETE QUERY RESULTS FILE 203

19.7 DISPLAY QUERY RESULTS 204

APPENDIX A DISPLAY DATABASE VARIABLES 205

A.1 STANDARD LOCAL DDB VARIABLES 205

A.1.1 Initial State of Standard Local DDB Variables 206

A.2 STANDARD GLOBAL DISPLAY DATABASE VARIABLES 207

A.3 USER-DEFINED DDB VARIABLES 207

Table of Contents

A.3.1 DDB Statement Syntax 209
A.3.2 Rules 210
A.3.3 Comments 211
A.4 DISPLAY DATABASE–INDIRECTION 213
A.5 SPECIAL DISPLAY DATABASE VARIABLES 213
A.6 HONEYWELL DEVELOPERS GLOBAL DISPLAY DATABASE 216
A.6.1 Avoiding Conflicts 216
A.6.2 Assignments 216

APPENDIX B CUSTOM CHANGE ZONE CONSTRUCTION 219

B.1 CUSTOM CHANGE ZONE EXAMPLE 219
B.1.1 Part 1—Display Database Variables Used 220
B.1.2 Part 2—Subpictures 221
B.1.3 Part 3—Constructing the Display 227

APPENDIX C KEY/BUTTON IDENTIFIERS 229

1 INTRODUCTION TO ACTORS

1.1 ACTORS DEFINED

Action procedures consist of one or more actors that specify a desired action. This manual describes actors in detail and provides examples of their use. The Action Procedures apply when building custom graphic displays with the Picture Editor or when configuring keyboard buttons with the Button Configurator. Button Configuration is described in Honeywell manuals, *Button Configuration Data Entry* and *Button Configuration Form Instructions*, in the *Implementation/Engineering Operations - 2* binder.

Actors are triggered by activating (e.g., touching) a target or pressing one of the configurable buttons on the Operator's Keyboard. The station must be running the Operator Personality or Operator functions in the Universal Personality to execute the actor.

The resulting action depends on which actor or actors are assigned to the target or button.

Typically, actors provide a way to call up specified displays on the same or other screens in the Universal Station console. They can output messages to the screen, update displays, read data from the screen and store it, or read information from various databases and present it on the screen.

Actors are grouped according to common functions (refer to the Table of Contents).

- **Display actors** call up various displays and provide screen-management functions.
- **Keyboard actors** simulate keyboard buttons.
- **Operator input actors** allow input from the operator.
- **Store actors** store values in either the local Display Database or the system database. Subsection 1.3 and Appendix A describe the Display Database.
- **Read and Store actors** are a combination of the two previous types.
- **Get actors** retrieve data from a specified database.
- **General actors** are miscellaneous functions that do not fit in other categories.
- **Range Checking actors** check to see if input values fall within specified limits.
- **Trend actors** are used to manipulate trend lines in custom displays.
- **Arithmetic, Logical, Compare, and Conditional actors** are used to perform mathematical, conditional, and comparative operations.

Examples in the next few pages show some of the many ways that actors can be used.

1.2 USE OF ACTORS

Each actor has a unique calling sequence, typically of the following form: ACTOR NAME (parameters). For example, GROUP(1, 6). This actor is used to call up a specific group display with a specific point selected (group 1, point 6 in this case).

Some actors have no parameters (see example 1), others have six or more parameters. An actor can return a value that is then used as a parameter for another actor. In such cases, the form is typically

ACTOR1 NAME(ACTOR2 NAME(parameters)).

ACTOR2 executes first to provide the parameter for ACTOR1 (see example 4).

When separated by semicolons, a series of actors can be executed in sequence from left to right, for example,

ACTOR1(parameters);ACTOR2(parameters);....ACTORn(parameters).

Example 3 and example 6 show how this is done. Of course, an actor that is used to return a parameter executes before the actor that uses that parameter, as explained above.

1.3 DISPLAY DATABASE

Several groups of actors store into, or read from, the Display Database (DDB). The local Display Database is a small scratch-pad memory in the US for display variables and exists from the time a display is called up until it is canceled. The Standard Display Database provides the following variables (also called local variables) for use in the display:

- 20 local and 20 global Boolean variables
- 4 local and 4 global Date/Time variables
- 20 local and 20 global Entity variables
- 10 local and 10 global Enumeration variables
- 20 local and 20 global Integer variables
- 20 local and 20 global Real variables
- 20 local and 20 global String variables
- 12 Trend Records (for use with Trend Application Subpictures only).

A user-defined display database is also available. It includes many more of the same variable types as the standard DDB but you can define the names. DDB variables are described in Appendix A.

Example 6 provides several examples of how DDB variables are used. Actors used with the Button Configurator can refer to only the Global Display Database variables.

1.4 Examples

By reading through the following examples, you should get a good idea of how actors can be used. These are only a few of the many combinations that are possible. Appendix B, entitled Custom Change Zone Construction, provides an advanced example.

The actors are only briefly described here and you should look them up in the pages that follow for a complete description.

Example 1:

Problem Statement— Provide a way for the operator to call up the Overview display by touching a target on a custom graphic display.

Solution— Add a target that has the action sequence

OVERVIEW

When the operator touches the target, the actor executes and the Overview display is called up on the same screen as the target.

Example 2:

Problem Statement— Provide a way for the operator to call up the display FRED by touching a target on a custom graphic display. The display FRED is to be called up on the same screen as the target.

Solution— Use the actor SCHEM (Call Up Schematic). Add a target that has the action sequence

SCHEM("FRED")

When this actor executes, the named display is called up.

Example 3:

Problem Statement— Provide a way for the operator to call up the display FRED on screen number 2 in the Universal Station console.

Solution— Use the actors CROSSCRN (Cross Screen) and SCHEM (Call Up Schematic). Add a target that has the action sequence

CROSSCRN(2);SCHEM("FRED")

When the target is touched, the Cross Screen actor conditions the system so that actor SCHEM calls up display FRED on screen number 2.

Alternate Solution— Assume that

- Button 1 is configured CROSSCRN(2)
- A display target is configured SCHEM("FRED")

Pressing button 1 and then touching the target causes the same result as before, that is display FRED is called up on screen number 2.

Example 4:

Problem Statement— Provide a way for the operator to type in the name of a display that is then called up on the same Universal Station screen.

Solution— Add a target to the display that has the action sequence

SCHEM (R_STR (..parameters...))

Explanation— When the target is touched, the Read String actor (R_STR) executes. This actor outputs a prompt message, opens a Text Input Port (Tip) on the screen, and waits for the operator to type in the name of a display. When the operator presses the Enter key, the Read String actor completes by reading the input data and passing it as a parameter to the actor SCHEM (Call Up Schematic).

The SCHEM actor then executes and calls up the named display.

Example 5:

Problem Statement— Provide a way for the operator to change the value of point D100.PV, which is an enumeration with the values "ON" and "OFF" The display is to have two targets. You want the first target to set the value to "ON" and the second target to set the value to "OFF."

Solution— Add two targets to the display. Assign the following action sequence to the first target, SS_ENM (D100.MODE,"ON"), which means Store Enumeration Data in the system database. D100.MODE is the variable and "ON" is the string value that is stored when the actor executes.

Assign the following action sequence to the second target, SS_ENM (D100.MODE,"OFF"), which is similar to the first, except that it stores the string "OFF" for the value of D100.MODE in the system database.

When one of the targets is touched, the appropriate string ("ON" or "OFF") is stored in the system database as the value for D100.PV.

Example 6: AC_RSS_1

Problem Statement— Create a display that allows the operator to specify a variable and change the value of that variable in the system database. The operator is to type in the variable ID and value.

The functions of the actors and variants are briefly described for each step, then the actual operation is reviewed. After you understand the overall operation, you can gain a better understanding by looking up the detailed description for each actor and examining the parameters that are used in this example.

Solution— The model for this example has been useful in operating displays and for troubleshooting. Although a touch screen is presumed, it will also work by selecting the targets. Figure 1-1 illustrates the model and it is described below.

1. Use the Add Solid command to draw a large solid blue background.
2. Draw a large white box on the solid. Divide the box into three sections. Do this with a single Add Line command so the result is one line-object. Select the line (SEL LINE); then invoke the command: ADD CONDITION. Set initial behavior to WHITE NO BLINK NO REV FULL. No other entries are required. Select the line again and invoke the command: ADD PRIORITY GFB
3. Add white text to the top of the upper section that says
   Touch here to enter Variable Id.
4. In the middle section, add white text
   near the top that says **Touch here to change Current Value**
   near the bottom that says **Variable Id:**
   just below the words Variable Id: that says **Current Value:**
5. Add text to the lower section that says **Touch here to return**. You may want to customize these words after reading through step 10.
6. In the middle section, add a Value just to the right of the words Variable Id: (+ in figure EX-1). On the Value screen form, enter STRING01 for the expression; enter TEXTL1:34 for the format.
7. In the middle section, add a Value just to the right of the words Current Value:

   On the Value screen form, enter VAR01**.** for the expression; UNKNOWN for the type.
   Enter G999999999999999999999999 for the format.
   Select the Value then invoke the command: ADD CONDITION. Set the initial behavior to WHITE NO BLINK NO REV FULL. No other entries are required.
8. Use the Add Target command to place an invisible target over the top section. For the Action section, enter the following:

   S_VAR(VAR01, R_VAR(20,-5,25,"ENTER VARIABLE NAME",FALSE,0));
   S_STR(STRING01,IE_VAR (VAR01));

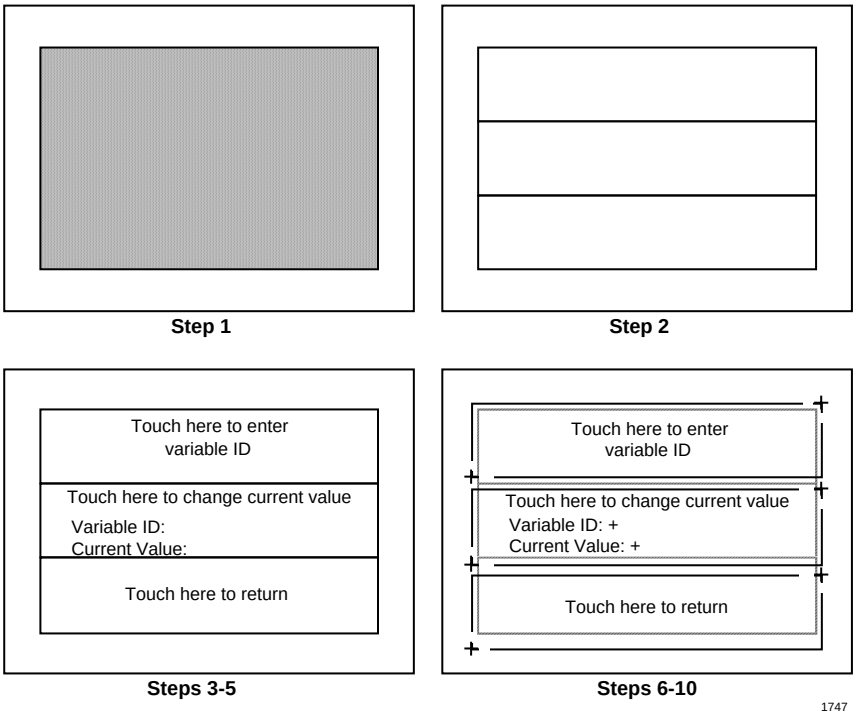


Figure 1-1 — Example 6

Explanation— The first actor, Read Variable (R_VAR) outputs the prompt ENTER THE VARIABLE NAME and presents a Text Input Port. Actor S_VAR stores operator input in DDB variable VAR01. UPDATE calls for an updated display.

9. Use the Add Target command to place an invisible target over the middle section. Avoid overlapping the upper target. For the Action section, enter the following:

   RS_SYS(VAR01.,20,-3,25,"ENTER NEW VALUE",FALSE,0);
   UPDATE(-1,0)

Explanation— Actor RS_SYS outputs the prompt ENTER NEW VALUE, and presents a Text Input Port. Whatever the operator types in is stored in system variable VAR01. UPDATE calls for an updated display.

10. Use the Add Target command to place an invisible target over the lower section. Avoid overlapping the middle target. For the Action section, enter the following:

    SCHEM("SCHEM")

Use any legitimate display name for "SCHEM" (e.g., use "SYS_MENU" to return to the System Menu). Alternatively, you can substitute the actor **QUE_KEY(PRR_DISP)** to call up the previous display.

Operation— After touching the upper target, the operator then sees the prompt **ENTER THE VARIABLE NAME** and enters the ID. The entry is stored in Display Database locations VAR01 and STRING01. Next, the display is updated. The two Values display the variable ID just entered, and its current value when the display is updated. Some objects were given special priorities and conditional behavior so that they would not be covered by the solid background.

By touching the middle target, the operator sees the prompt **ENTER NEW VALUE** and an entry area. The value entered is stored in the point's parameter and the change is logged through the Real-Time Journal.

When the lower target is touched, the named display or the prior display is called up depending on which actor you use.

The text input ports should appear in blank areas of the top and middle sections but, if necessary, you can adjust them. The parameters 20,-5,25 or 20,-3,25 in the action sequences specify the port's coordinates and length (25). In this case, the port is specified 20 character units to the right and either 5 or 3 character units below the target's upper left-hand corner.

1.5 ACTOR DATA ENTRY

Action procedures are typed into a screen form that is presented while working with the Picture Editor's Add Target command, or when configuring keyboard buttons (refer to the *Button Configuration Data Entry* manual).

As shown in Figure 1-2, the screen form specifies the target or button that is being configured and provides an action block where the action procedure can be entered. Other information may also appear on this screen form, depending on whether a target or button is being configured.

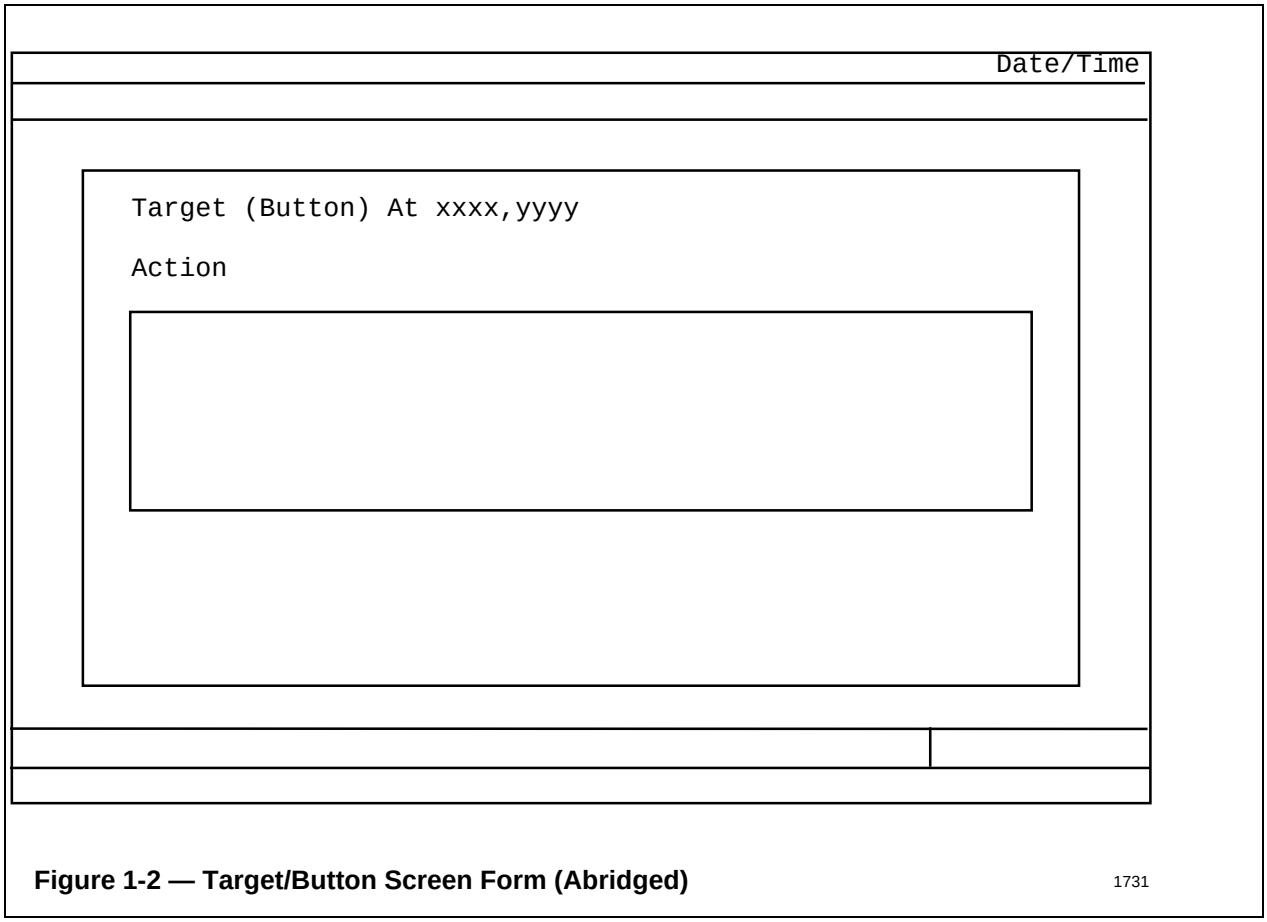


Figure 1-2 — Target/Button Screen Form (Abridged) 1731

1.5.1 Calling Sequence

Each actor has a unique calling sequence, typically of the form ACTOR NAME(parameters). For example, GROUP(1,6).

There are two ways to enter the calling sequence:

1. Refer to the detailed description for the actor. Type the calling sequence directly into the action block. For example,

   ```
   Action
   ```

   ```
   GROUP(1,6)
   ```

   After the calling sequence has been entered, press the ENTER key. The entry is then checked for correctness. If the CANCEL key is pressed before the ENTER key, the operation is canceled. If an error is detected in the syntax of the actor, that part of the statement that is in error is rewritten in red. An error message is presented on the screen to help clarify the error. The action sequence is also checked for completeness of parameter data.

2. If you do not wish to type all (or any part) of the parameter data in the action block, you will be asked for any parameters that were not specified. When the Enter key is pressed, a screen form is presented for each item that requires entries.

   Figure 1-3 illustrates a typical parameter screen form that is presented when only the actor name GROUP is typed into the action block. The actor reference is shown at the top of the form. This identifies the item that is being resolved.

   Next, the parameter is identified and an entry block, with the default value (if any), is provided.

   The second item is repeated for each parameter required.

   Finally, the action-sequence window is a snapshot of the action block. This window shows the correlation between the action sequence and the parameters that are being entered.

   Type the parameters into the entry block provided. When you press the Enter key, the entered parameter data are incorporated into the calling sequence in the proper syntax. If an error is detected in the parameter data, the original Target/Button screen form is displayed with any errors shown in red. If no error is detected, processing of actor sequence continues and additional parameter screen forms may appear to resolve the entire action sequence.

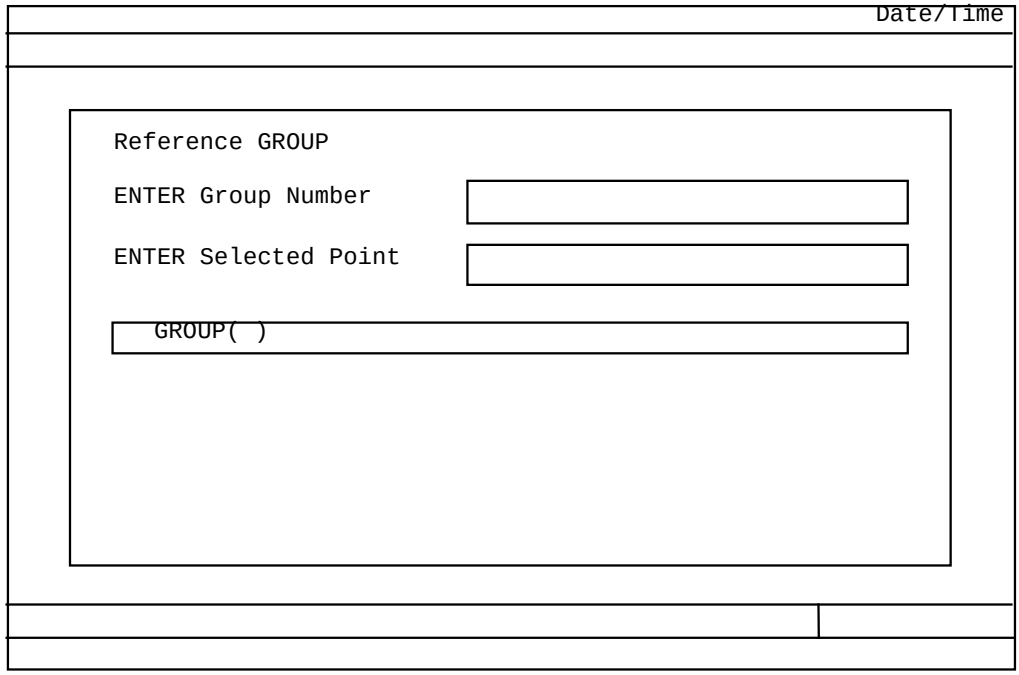


Figure 1-3 — Typical Parameter Screen Form for Actors

1733

Comments—Comments are useful to explain your sequences. You can enter comments before or after any action sequence (a semicolon is used to separate action sequences). Comments must be enclosed in curly braces { }.

Example—

```
IF(CMP_R(ENT01G.PV,GT,10.0)); {Set flag for variant action.}
CROSSCRN(01);MSGSUM;{Write Message Summary on screen 1}
ELSE....;
```

If you need additional room to enter action sequences or comments, press the PAGE FWD key and continue the entry. Do not allow an action sequence to be broken between two pages. For example, the actor S_ENT(ENT01G,R_ENT(10,20,10,"ENTITY?",TRUE,2)); cannot be broken except immediately after the semicolon delimiter. Do not break up an IF, THEN statement. Press PAGE BACK to display previous pages.

1.6 ACTOR ERROR HANDLING

When a target or button with a series of actors is touched, the actors for that target begin executing. Continued execution depends upon successful execution of the previous actor. As long as each actor continues to convey success to its successor, the next actor executes; however, when a failure condition is detected the remaining actors for that target essentially do nothing.

1.7 SCREEN REGIONS

For custom-graphic displays, the screen is configured (at run time) into two regions. The first region contains the standard top 2-line display and is used for interaction with the user (prompts, etc.). This region is referred to as the **prompt region**. The second region contains the picture, typically, a schematic. This region is referred to as the **schematic region**. For use with certain actors, each of these regions are identified by number. The prompt region is region number 1 and the schematic region is region number 2. Regions 3 and 4 are **overlay** regions (refer to the Multiple Overlay and Background Overlay actors). They are the same size as region 2. In each case, the origin is at the lower left corner of the region. Figure 1-4 illustrates the screen regions.

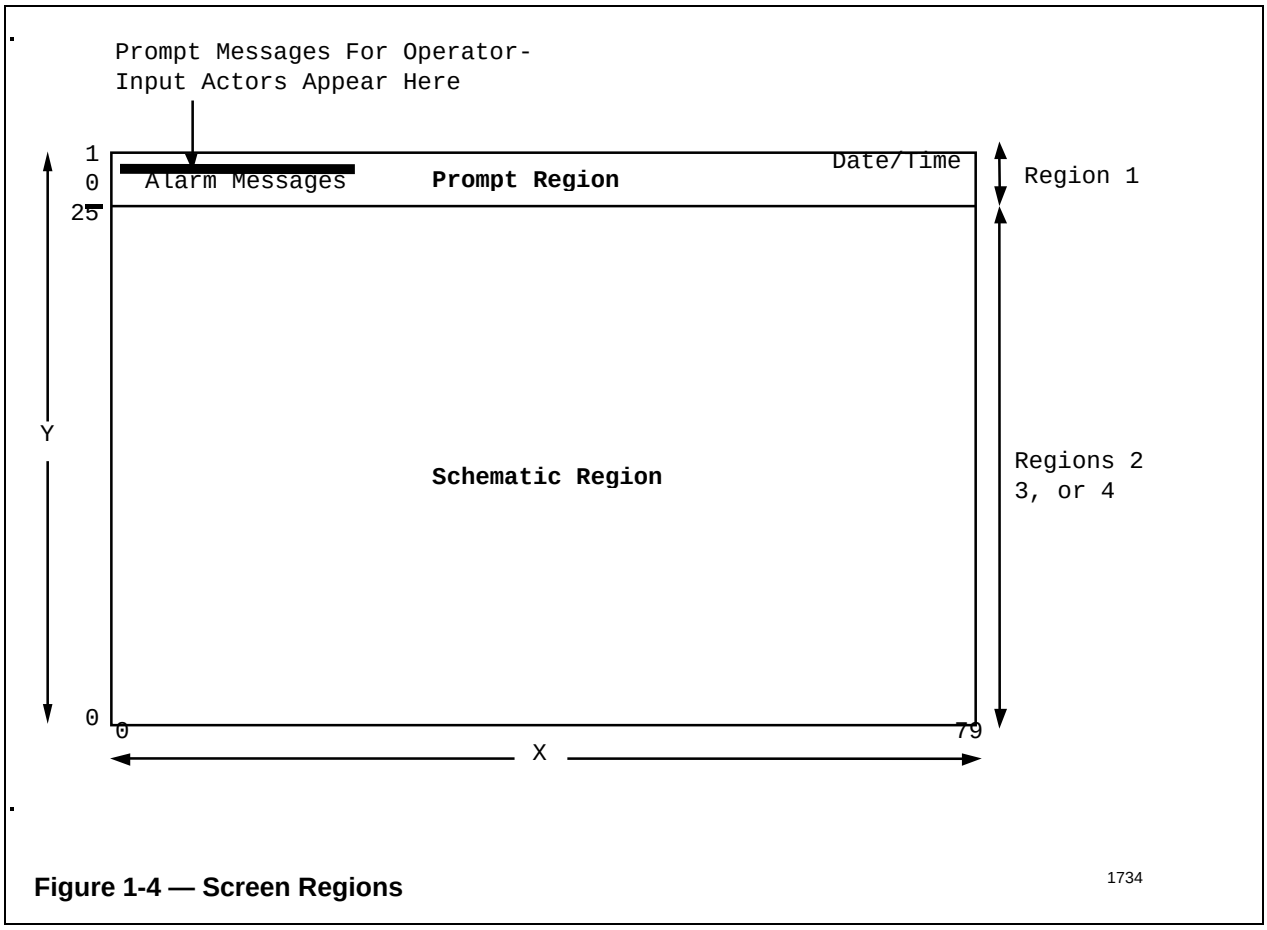


Figure 1-4 — Screen Regions

1.8 TESTING ACTOR OPERATION

Action procedures are built by using the Picture Editor or Button Configuration functions of the Engineering Personality or the Universal Personality. The button configuration file and custom-graphic display files are compiled and the file names are listed during Area Database configuration (refer to the *System Startup Guide*, tasks 23, 24, and 29). After the Area Database is installed and the operator functions are running (either in the Operator's Personality or the Universal Personality), you should activate each target or button and verify proper action. Also, check the execution time. Depending on the quantity and mix of actors used, an action procedure can execute in a few milliseconds or it can take up to several minutes, but once a target or button is activated, the operator cannot call up another display until the associated action procedure has run to completion.

If performance is unacceptable, you may want to rebuild with a different action sequence or use Control Language to implement the action. If you want to rebuild the action procedure, you can read the file back into the Picture Editor or (Button Configurator), make revisions, then recompile using the same file name.

1.9 REFERENCES

Title	Binder
Picture Editor Reference Manual	Implementation/Engineering Operations - 2
Picture Editor Form Instructions	Implementation/Engineering Operations - 3
Picture Editor Data Entry	Implementation/Engineering Operations - 3
Button Configuration Form Instructions	Implementation/Engineering Operations - 1
Button Configuration Data Entry	Implementation/Engineering Operations - 1
Keyboards	Implementation/Engineering Operations - 1

2 DISPLAY ACTORS

Display actors are used to call up other displays and menus, to move the cursor, output prompt messages, etc.

2.1 CALL UP THE ALARM ANNUNCIATOR DISPLAY

This actor is used to call up the Alarm Annunciator display.

Calling Sequence— ALARMANN

Optional Parameter Entry Form— none

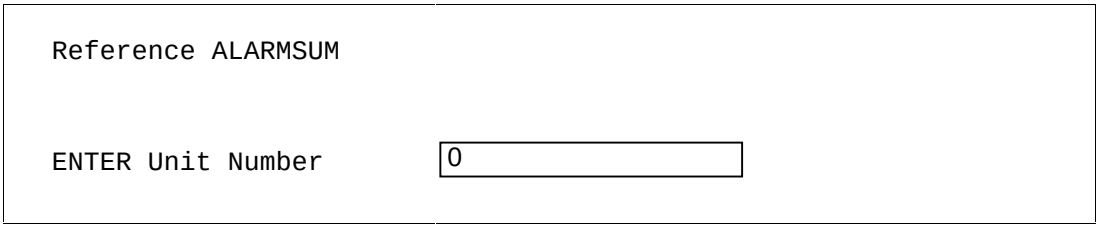
2.2 CALL UP THE ALARM SUMMARY DISPLAY

This actor is used to call up the Area Alarm Summary display or an individual Unit Alarm Summary display.

Calling Sequence— ALARMSUM(UNIT NUMBER)

Example— ALARMSUM(0) calls up the Area Alarm Summary Display
Example— ALARMSUM(01) calls up the Unit 1 Alarm Summary Display

Optional Parameter Entry Form



Parameter Explanation— Unit Number ranges from 1 through 36 and allows you to specify the particular Unit Summary display. A specification of zero (0) requests the Area Summary display.

2.3 CALL UP THE ATTRIBUTE POINT SUMMARY DISPLAY

This actor is used to call up the attribute Point Summary display.

Calling Sequence— ATTRIBPS

Optional Parameter Entry Form— none.

2.4 CALL UP BACKGROUND OVERLAY

This actor is used to call up a specified overlay over the currently displayed schematic. The overlay is displayed within the specified coordinates.

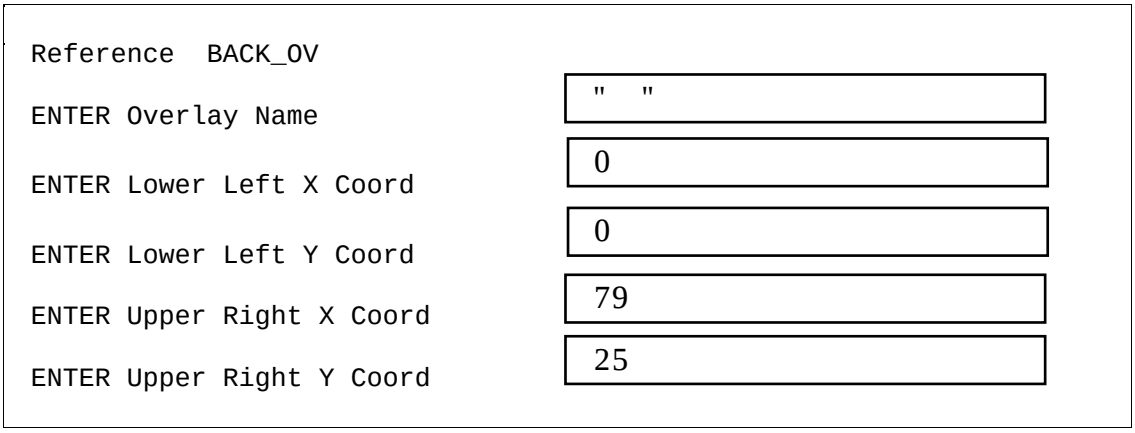
The Background Overlay actor can be invoked only if a schematic is currently displayed, otherwise an error message results. If the requested overlay is found, it is called up without clearing the schematic that is currently displayed. If the specified overlay is already being displayed, a demand update is done to the overlay. If the requested overlay is not found, an error display results. If the specified overlay is not the one currently being displayed, the screen area occupied by the current overlay is cleared and the new overlay is displayed. Only one Background Overlay can be invoked at a time, but the current Background Overlay can be replaced by using the actor to call a different Background Overlay. A Background Overlay may exist simultaneously with another overlay that was called by the Overlay actor or the Multiple Serial Overlay actor, but only one Background Overlay can exist at a time on the same schematic.

The Background Overlay update rate is slower than the schematic; therefore, it is most useful when data collection is not necessary at the regular four second rate. This is the major difference between the Background Overlay and the Multiple Serial Overlay. If data must be passed between the schematic and the overlay, you must use the global display database. The size of the overlay, plus the schematic, must not exceed the current maximum size for a schematic.

Calling Sequence—
BACK_OV("OVERLAY NAME",LOWER LEFT X,LOWER LEFT Y,UPPER RIGHT X,UPPER RIGHT Y)

Example— BACK_OV("BKOVLY1",0,0,79,25)

Optional Parameter Entry Form



Call Up Background Overlay (Contd.)
Parameter Explanation

- Overlay Name is the name of the schematic that is to be called up as an overlay. It is a 1-8 character string.

- Lower Left X and Lower Left Y Coordinates (in character units) are the coordinates of the lower left corner of the region being defined. The range for X is 0-79, and for Y is 0-25.

- Upper Right X and Upper Right Y Coordinates (in character units) are the coordinates of the upper right corner of the region being defined. The range for X is 0-79, and for Y is 0-25.

Notes and Precautions

The BACK_OV actor itself cannot be used in a Background Overlay. It must be implemented from a Multiple Overlay, through a target in a schematic, or by use of a configurable button.

When the first Background Overlay is called up, the underlying schematic is not cleared. If a subsequent Background Overlay is called up, the screen area occupied by the previous Background Overlay (including any schematic objects in that area) is cleared; therefore, schematic objects should not be placed in areas where you intend to use Background Overlays.

The lower left X-Y coordinates and upper right X-Y coordinates define two corners of a rectangle (similar to the rectangle formed when building a target). The rectangle specifies that part of the overlay picture that is used to overlay a schematic. In the example BACK_OV("BKOVLY1",0,0,79,5), the lower 6 lines, 80 columns wide of the picture named BKOVLY1 is used to overlay the corresponding area of a schematic. Refer to Figure 1-4 for screen coordinate information.

2.5 CALL UP THE BOX POINT SUMMARY DISPLAY

This actor is used to call up the Box Point Summary display.

Calling Sequence— BOX_PS

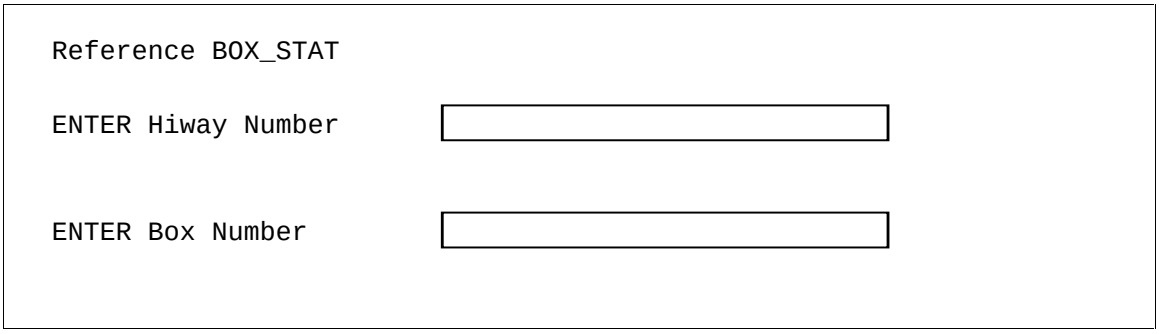
Optional Parameter Entry Form— none.

2.6 CALL UP THE BOX STATUS DISPLAY

This actor is used to call up the Box Status display.

Calling Sequence— BOX_STAT(HWY NUMBER,BOX NUMBER)

Optional Parameter Entry Form



Parameter Explanation—Hiway Number ranges from 0 through 100 and specifies the Data Hiway that contains the desired box. Box Number specifies the box for which status is requested, and ranges from 0 to 100.

2.7 CALL UP THE CUSTOM SAVE DISPLAY

This actor is used to call up the Custom Save display.

Calling Sequence— CUSTSAVE

Optional Parameter Entry Form— none.

2.8 CALL UP CHANGE ZONE (STANDARD)

The CHGZONE actor lets the operator call up a standard Change Zone. The Change Zone then allows the operator to change a subset of parameters; this subset depends on the point type specified in the target's action procedure.

NOTE

The CHG_ZONE actor is used to call up both the Standard Change Zone and Enhanced Change Zone. If the subpicture/overlay added to the schematic is ENHAN_CZ, then at run time when the schematic is invoked, Enhanced Change Zone will appear on the display. If the subpicture/overlay added to the schematic is CHG_ZONE, then at run time when the schematic is invoked, Standard Change Zone will appear on the display.

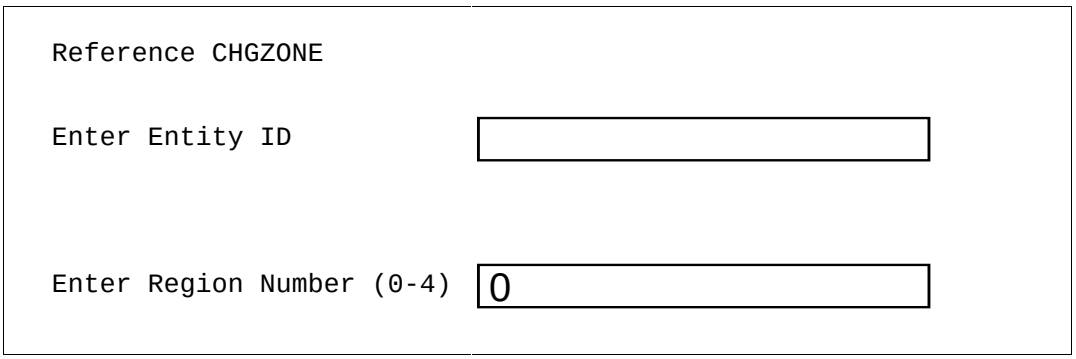
Source and object files for the Standard Change Zone subpictures are supplied on the volume &DSY.

Standard Change Zones are illustrated later in this section.

Calling Sequence— CHGZONE(Entity ID,REGION) or
CHG_ZONE(Entity ID,REGION)

Example— CHGZONE(A100,2) calls up a Change Zone for point A100

Optional Parameter Entry Form



Parameter Explanation

Entity ID is the point name

Region Number has the following meaning

Region = 0 Implies the subpicture is in the same region as the target.
Region = 2 Explicitly states that the subpicture is on the main schematic.
Region = 3 Explicitly states that subpicture is on the overlay.

If a change zone that references an illegal point type is activated, the message "INVALID POINT TYPE" is displayed in the Change Zone area.

Construction— The first step is to build the main display. Reserve a 3-line by 80-column area for the Change Zone, preferably at the bottom of the display. Then, add one or more targets to the display (each target typically calls the Change Zone for one point). The action sequence for the actor can take several forms.

If you want to add the Change Zone to the main schematic, use the form

CHGZONE(*point name*,2)

Then, add the subpicture called CHG_ZONE to the display (usually, the command is ADD SUBPICTURE NET>&DSY>CHG_ZONE). Typically, the subpicture's origin should be located at the bottom-left corner of the display.

At this point a dummy Change Zone subpicture appears in the display. The actual Change Zone image depends on the point type and is not visible until operating time. See Figures 2-1 through 2-8 for examples of typical Standard Change Zone images. Note that if the Longer Tag Names option is enabled, the Plant Information Network (PIN) Tag Name appears as two lines of half-height text. Otherwise, this information appears as full-height text.

Faster Operation— Faster updates are possible if the Change Zone is in an overlay. The simplest way is to use the standard Change Zone subpicture (i.e., its object file) as the overlay. To do so, build the main schematic, again reserving space for the Change Zone. Then, add a target with the action sequence

OVERLAY("CHG_ZONE");CHGZONE(*point name,region #*)

Faster updates are also possible when the FAST key on the Operator's Keyboard is pressed; refer to the Set Collection command for configuration information.

Operation— At operating time, when the target (or button) is activated, the overlay actor calls up the Change Zone image and displays a Change Zone for the named point. The overlay actor is described elsewhere in this manual.

Area Pathname Catalog considerations— For fastest updates, the overlay should be kept resident in the Universal Station's memory. To do this, enter the overlay's filename (CHG_ZONE in this case) as one of the first items on the Memory Resident Schematic File name list (refer to the pathname catalog configuration task in the *System Startup Guide* for configuration procedures.)

The system must be able to find the schematics and overlays; therefore, you must either copy the object file (CHG_ZONE.DO in this case) to a volume that is listed in the Schematic Pathnames search list, or add the pathname NET>&DSY to the search list.

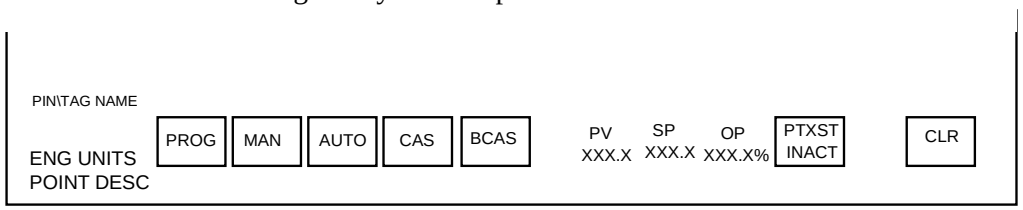
Adding custom features— Because both the source and object files for the change zone subpicture are provided, an alternative to using the standard subpicture directly as an overlay is to add it to a display that contains other objects of your choice and use that display as an overlay. In such cases, you would substitute whatever filename you give that display for CHG_ZONE in the overlay example just described.

CAUTION

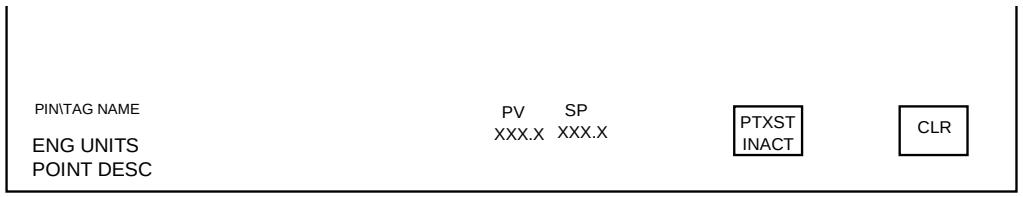
You should never modify the standard subpicture's source file (CHG_ZONE.DS).

$CZ_ACT— $CZ_ACT is a boolean parameter that is set true when either the CHG_ZONE actor or the USER_CZ actor executes. It is set false when the CLEAR_CZ actor executes.

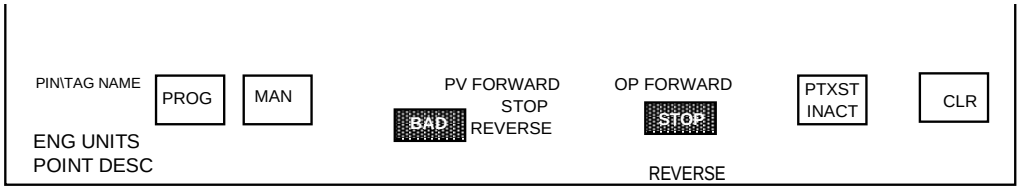
For a PM/APM/HPM Regulatory Control point:



For a PM/APM/HPM Regulatory PV point:



For a PM/APM/HPM Digital Composite point:



For a PM/APM/HPM Analog Output point:

For a PM/APM/HPM Analog Input point:

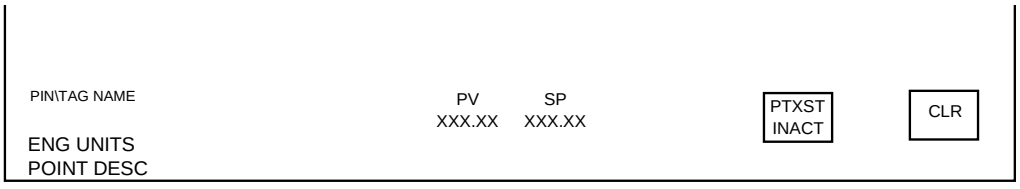
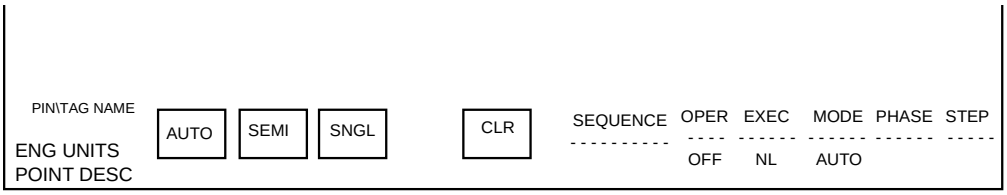


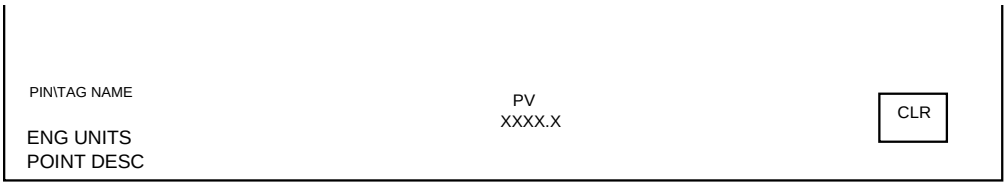
Figure 2-1 — Change Zone Images

Chg 11289

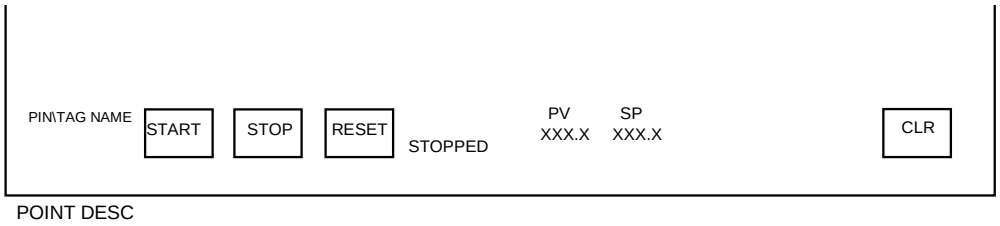
For a PM/APM/HPM Process Module point:



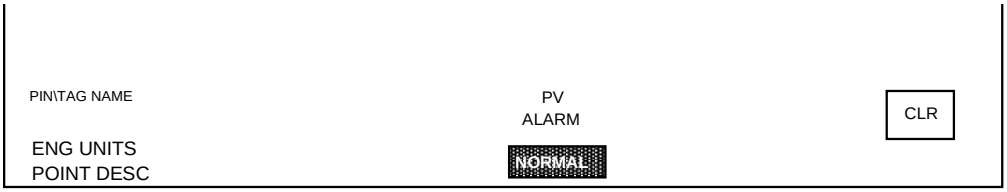
For a PM/APM/HPM Numeric Point:



For a PM/APM/HPM Timer point:



For a PM/APM/HPM Flag point:

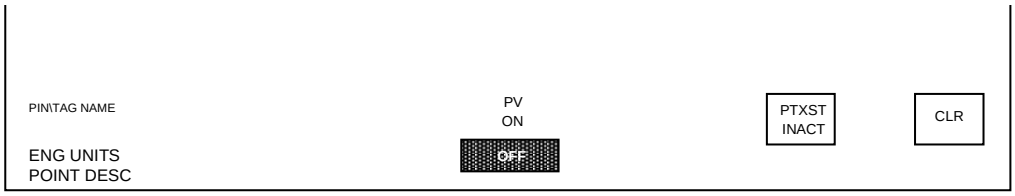


For a PM/APM/HPM Logic point:

Figure 2-2 — Change Zone Images

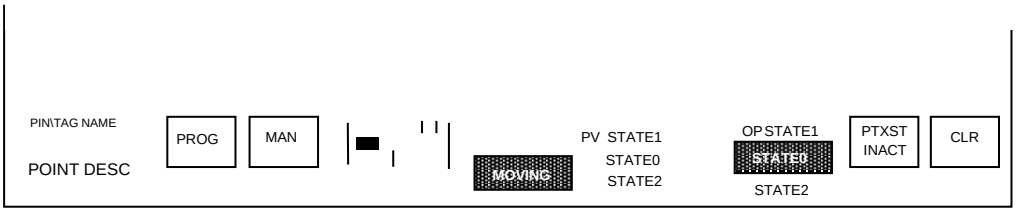
chg11289

For a PM/APM/HPM Digital Input point:

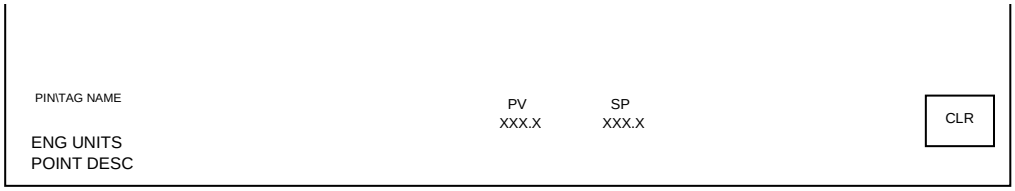


For a PM/APM/HPM Digital Output point:

For an APM or HPM Device Control point:



For an HG Analog Input point:

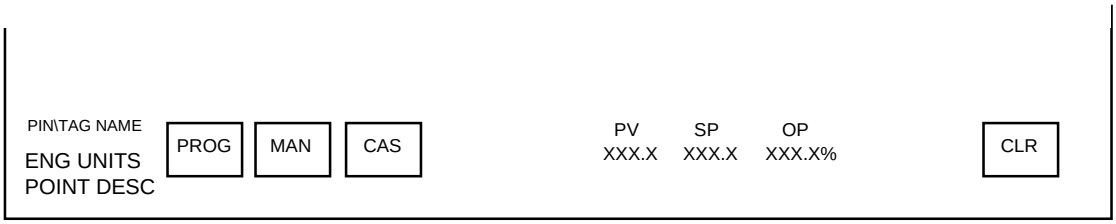


For an HG Analog Output point:

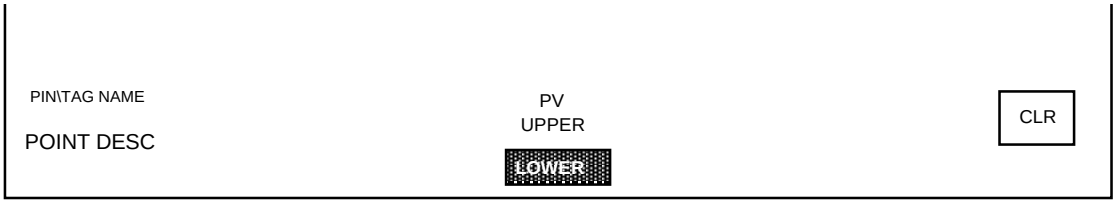
Figure 2-3 — Change Zones Images

Chg 11289

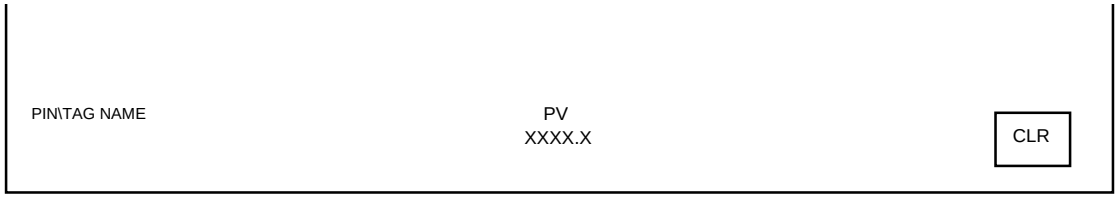
For an HG Analog Composite point:



For an HG Flag point:



For an HG Numeric point:



For an HG Process Module point:

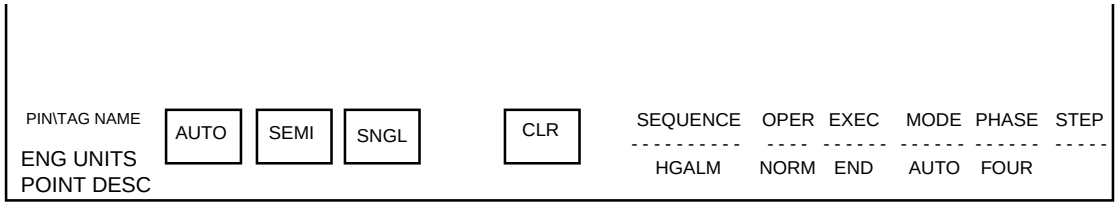
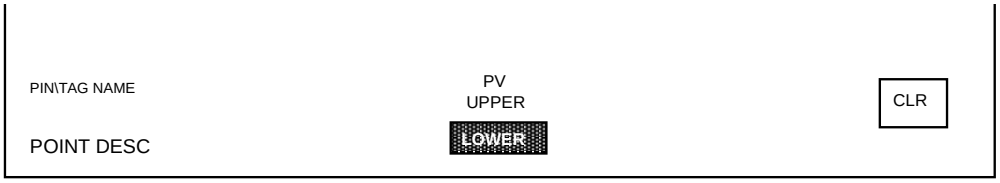


Figure 2-4 — Change Zone Images

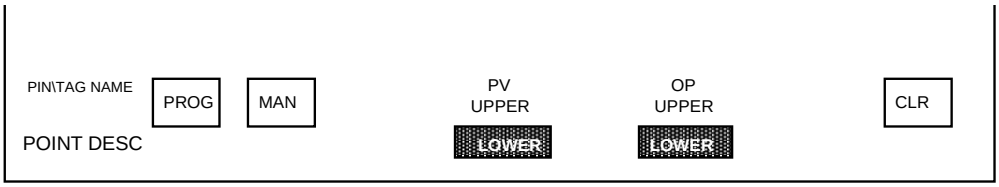
11289

For an HG Logic Block point:

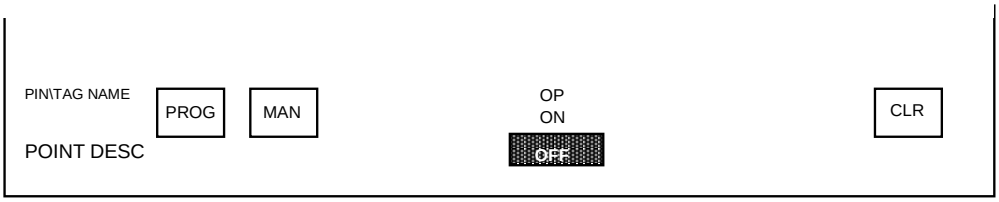


For an HG Control Counter point:

For an HG Digital Composite point:



For an HG Digital Output point:



For an HG Digital Input point:

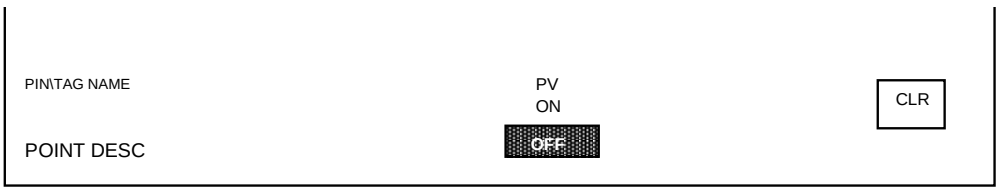
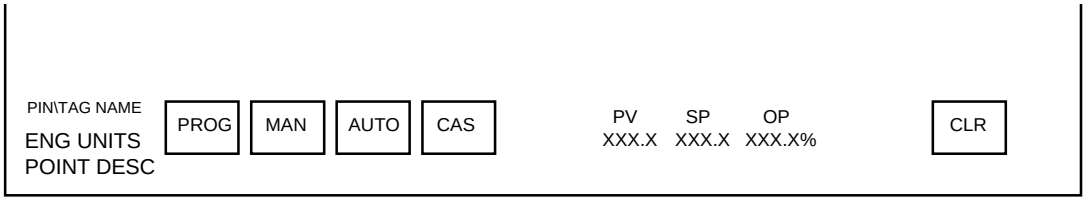


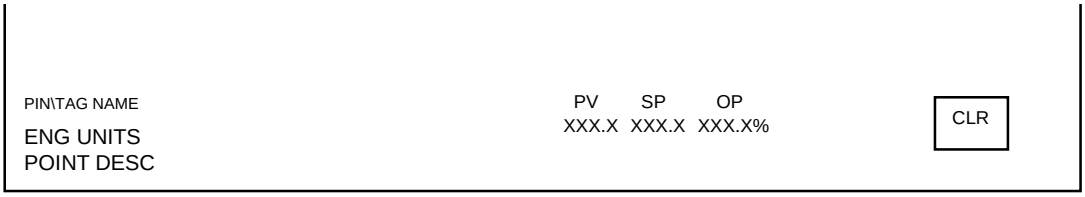
Figure 2-5 — Change Zone Images

11289

For an HG Regulatory Control point:



For an HG Counter point:



For an HG Timer point:

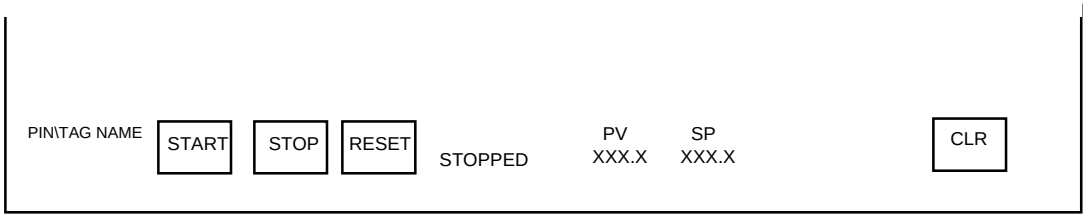
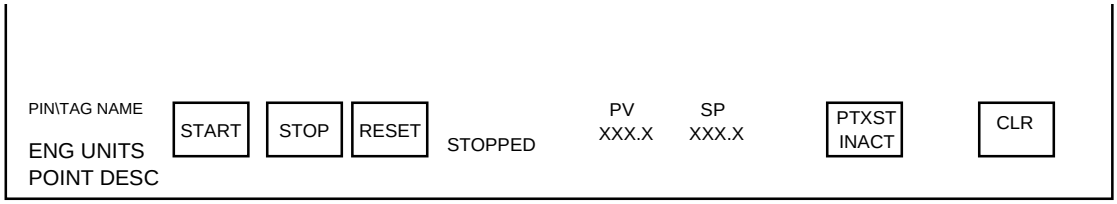


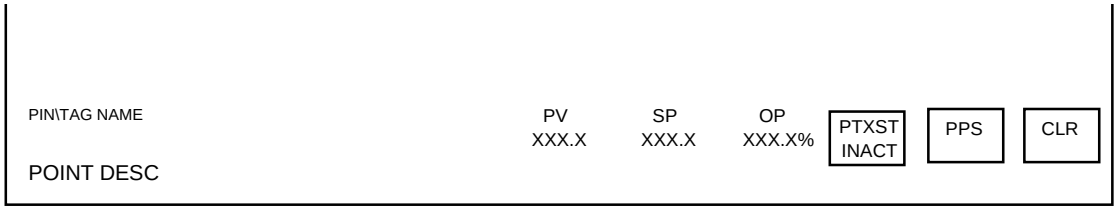
Figure 2-6 — Change Zone Images

11289

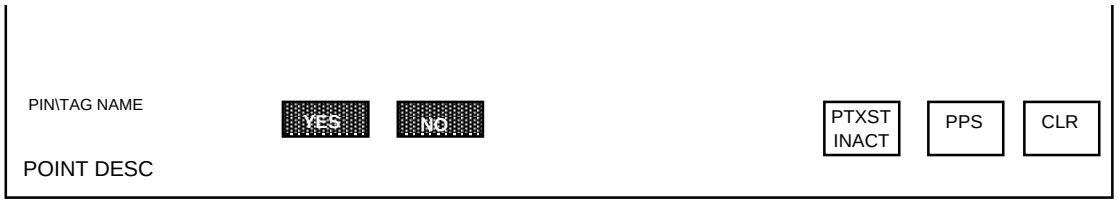
For an AM Timer point:



For an AM Counter point:



For an AM Switch point:

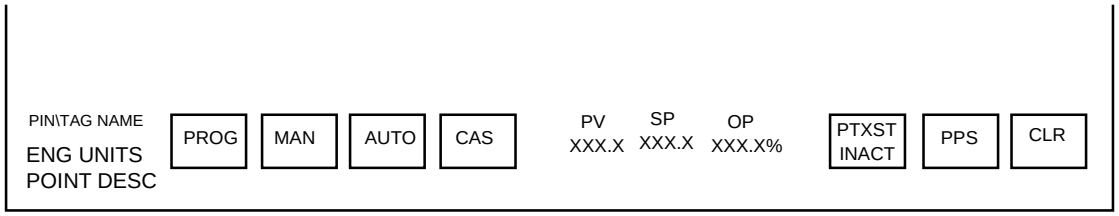


For an AM Custom point:

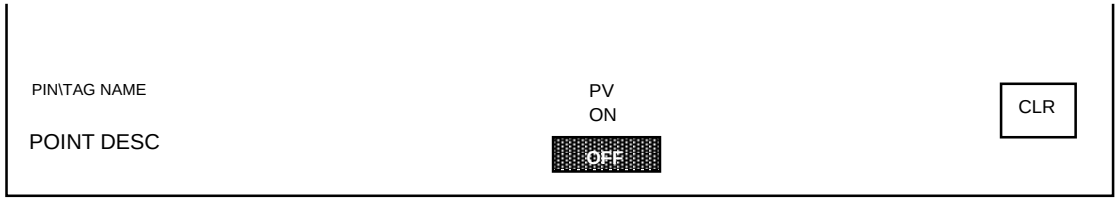
Figure 2-7 — Change Zone Images

11289

For an AM Regulatory Control point:



For an AM Flag point:



For an AM Numeric point:

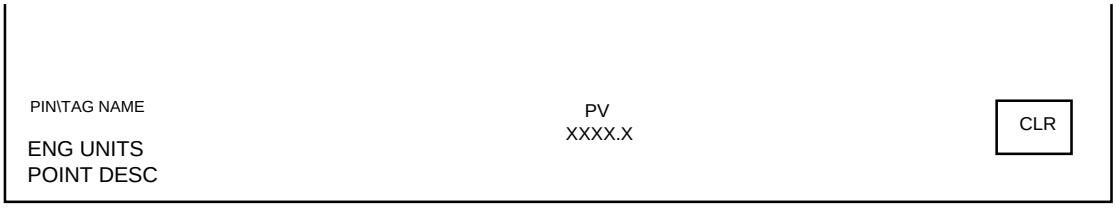


Figure 2-8 — Change Zone Images 11289

2.9 CALL UP CHANGE ZONE (ENHANCED)

The Enhanced Change Zone is similar to the Standard Change Zone, except that the Enhanced Change Zone contains additional point status information and a mini-trend graph.

The Enhanced Change Zone is available in the Picture Editor as a sub-picture or overlay and can be used in a custom schematic.

NOTE

The CHG_ZONE actor is used to call up both the Standard Change Zone and Enhanced Change Zone. If the subpicture/overlay added to the schematic is ENHAN_CZ, then at run time when the schematic is invoked, Enhanced Change Zone will appear on the display. If the subpicture/overlay added to the schematic is CHG_ZONE, then at run time when the schematic is invoked, Standard Change Zone will appear on the display.

Enhanced Change Zone is applicable only for the 18 point types that are referred in this section. For all the other point types, Enhanced Change Zone and Standard Change Zone displays are identical.

The CHG_ZONE actor lets the operator call up the Enhanced Change Zone. The Enhanced Change Zone then allows the operator to change a subset of parameters. This subset depends on the point type specified in the target's action procedure.

The Enhanced Change Zone contains the following information in addition to the data present in the Standard Change Zone.

- Indicators for the System Alarm Status, Point Alarm Status, PV Status, OP Status, SP Status and Point Status (same as the Detail Display).
- A Trend Graph of the SP, PV and OP parameters with targets to change time base. The trend in the Enhanced Change Zone takes the data from HM if the point is historized and the timebase is greater than or equal to 5 MIN, otherwise it displays the real time values.
- A bar to the left of the trend graph representing the OVERVAL percentage value of full PV EU or SPEU range, depending on the selected window mode. A target on the OVERVAL bar toggles between the two modes.
- Color-coded tic marks reflecting the alarm priorities, beside the trend graph representing the PV alarm limits, the Deviation alarm limits and the SP value.
- State indicators with color-coded boxes for the Digital Composite points. (same as Detail Display)
- SP TV Ramping - If SPOPT = TV, then a target appears which allows swapping the PV, SP and OP parameters with TVPROC, SPTV and RAMPTIME parameters. These SP target value ramping parameters displays current values and can be changed by the operator.
- In addition, the display format for the PTEXECST parameter is changed as compared to the Standard Change Zone. The PTEXECST parameter cannot be changed during run time from Enhanced Change Zone.

NOTE

A schematic user can add either the Enhanced Change Zone or the Standard Change Zone, but not both. However, if the user combines two schematics using the MULT_OV actor, with one schematic having the Standard Change Zone and the other having the Enhanced Change Zone, then both are displayed at the same time.

Source and object files for the Enhanced Change Zone subpictures are supplied on volume &DSY.

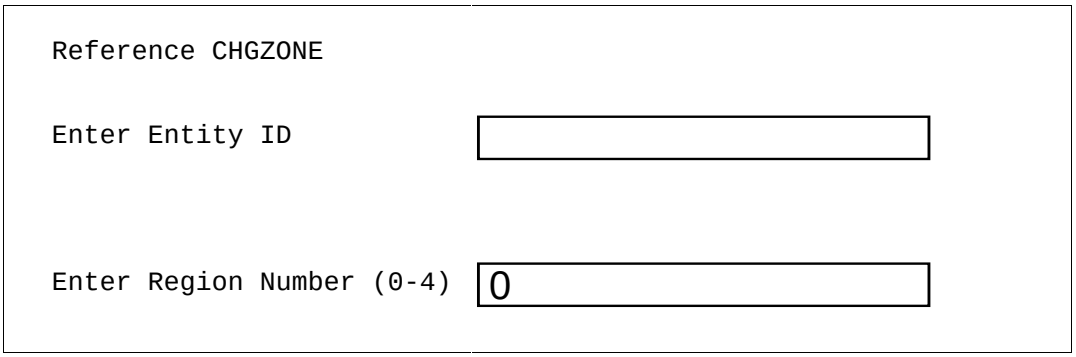
Calling Sequence

The existing Change Zone actor is used to invoke the Enhanced Change Zone also.

CHGZONE(Entity ID,REGION) or
CHG_ZONE(Entity ID,REGION)

Example— CHGZONE(A100,2) calls up a Change Zone for point A100

Optional Parameter Entry Form



Parameter Explanation

Entity ID is the point name.

Region Number has the following meaning.

Region = 0 Implies the subpicture is in the same region as the target.
Region = 2 Explicitly states that the subpicture is on the main schematic.
Region = 3 Explicitly states that subpicture is on the overlay.

If a Change Zone that references an illegal point type is activated, the message INVALID POINT TYPE is displayed in the Change Zone area.

Construction— The first step is to build the main display. Reserve a 3-line by 80-column area for the Change Zone, preferably at the bottom of the display. Then, add one or more targets to the display (each target typically calls the Change Zone for one point). The action sequence for the actor can take several forms.

If you want to add the Change Zone to the main schematic, use the form

CHGZONE(*point name*,2)

Then, add the ENHAN_CZ subpicture at the specific coordinates of (0,0) using the following command:

ADD SUB NET>&DSY>ENHAN_CZ 0 0

Typically, the subpicture's origin is located at the bottom-left corner of the display. At this point a dummy Enhanced Change Zone subpicture appears in the display. The actual Enhanced Change Zone image depends on the point type and is not visible until the operating time.

See the illustrations in the section Examples of Enhanced Change Zone Displays for the operating time displays.

NOTE

If the Longer Tag Names option is enabled (i.e., has been selected in the NCF), the Plant Information Network (PIN) Tag Name appears in two lines of half-height text. Otherwise, this information appears as full-height text.

Faster Operation— Faster updates are possible if the Enhanced Change Zone is in an overlay. The simplest way is to use the Enhanced Change Zone subpicture (i.e., its object file) as the overlay. To do so, build the main schematic, again reserving space for the Enhanced Change Zone. Then, add a target with the action sequence

OVERLAY("ENHAN_CZ");CHGZONE(*point name,region #*)

Faster updates are also possible when the FAST key on the Operator's Keyboard is pressed. Refer to the Set Collection command for configuration information.

Operation— At operating time, when the target (or button) is activated, the overlay actor calls up the Enhanced Change Zone image and displays the Enhanced Change Zone for the named point. The overlay actor is described elsewhere in this manual.

Area Pathname Catalog considerations— For fastest updates, the overlay should be kept resident in the Universal Station's memory. To do this, enter the overlay's filename (ENHAN_CZ in this case) as one of the first items on the Memory Resident Schematic File name list (refer to the pathname catalog configuration task for the Area Database in the *System Startup Guide*).

The system must be able to find the schematics and overlays, therefore, you must either copy the object file (ENHAN_CZ.DO) to a volume that is listed in the Schematic Pathnames search list, or add the pathname NET>&DSY to the search list.

Adding custom features— Because both the source and object files for the Enhanced Change Zone subpicture are provided, an alternative to using the standard subpicture directly as an overlay is to add it to a display that contains other objects of your choice and use that display as an overlay. In such cases, you would substitute whatever filename you give that display for ENHAN_CZ in the overlay example just described.

CAUTION

Never modify the Enhanced Change Zone subpicture's source file (ENHAN_CZ.DS).

Examples of Enhanced Change Zone Displays

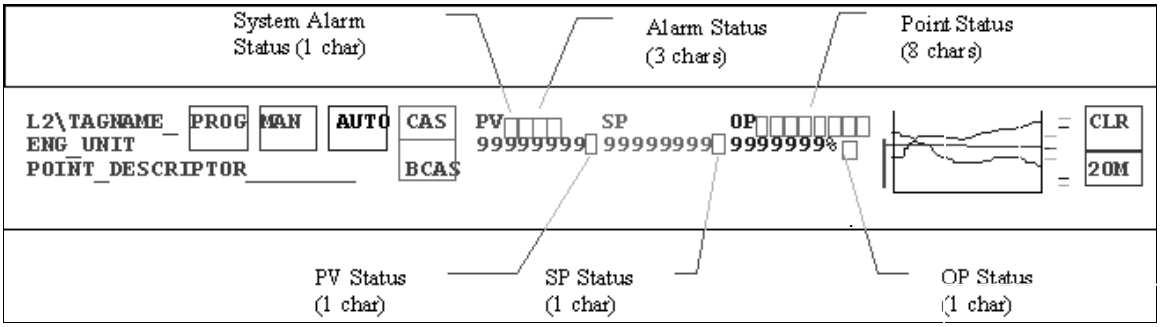
The Enhanced Change Zone display is different from the Standard Change Zone display only for the following point types,

1. PM/APM/HPM Regulatory Control Point
2. PM/APM/HPM Regulatory PV Point
3. PM/APM/HPM Digital Composite Point
4. PM/APM/HPM Analog Output Point
5. PM/APM/HPM Analog Input Point
6. PM/APM/HPM Digital Input Point
7. PM/APM/HPM Digital Output Point
8. APM/HPM Device Control Point
9. HG Analog Input Point
10. HG Analog Output Point
11. HG Analog Composite Point
12. HG Digital Composite Point
13. HG Digital Output Point
14. HG Digital Input Point
15. HG Regulatory Control Point
16. AM Regulatory Control Point
17. Fieldbus PID Point
18. Fieldbus Analog Input Point
19. Fieldbus Analog Output Point

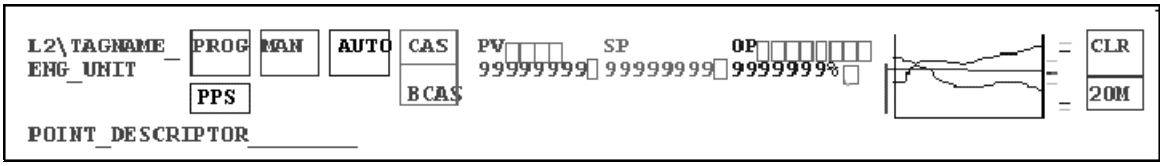
NOTE

For all other point types (which do not belong to the above listed 19 point types), the Standard Change Zone and Enhanced Change Zone displays are the same.

The different indicators are briefly explained using the example for a PM/APM/HPM Regulatory Control Point. For the description of the indicators used refer Table 3.3 to 3.8 in the Process Operations Manual.



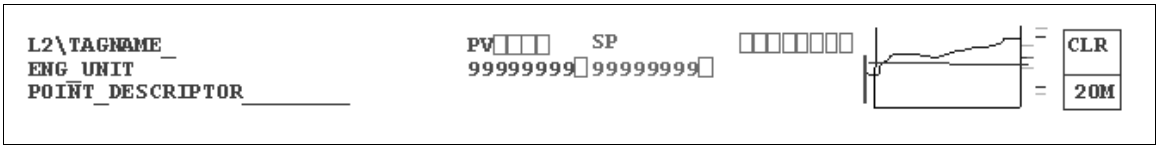
For a PM/APM/HPM Regulatory Control Point



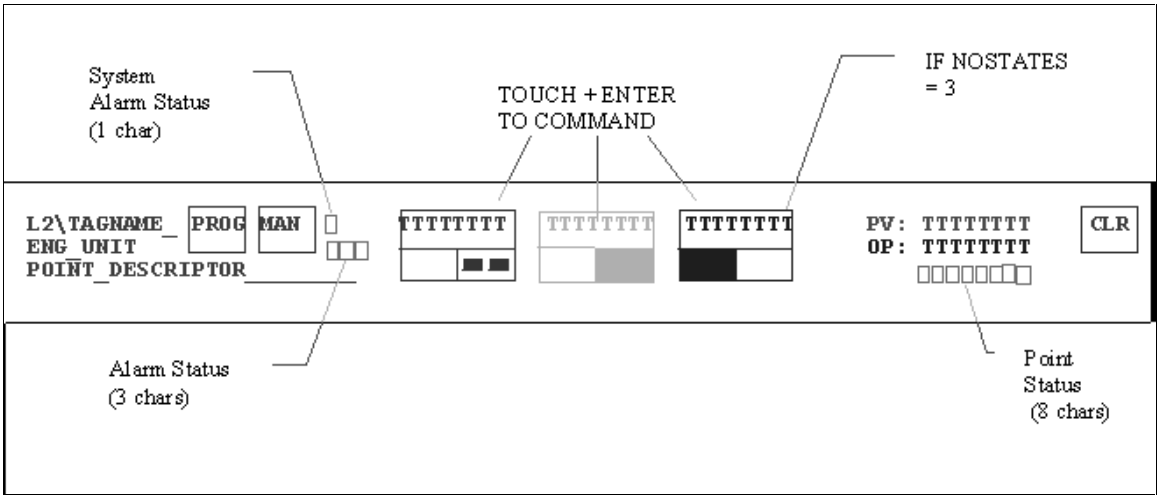
NOTE

The PPS target is located below the PROG target in the Enhanced Change Zone whereas in the Standard Change Zone it is displayed near the CLR target.

For a PM/APM/HPM Regulatory PV Point



For a PM/APM/HPM Digital Composite Point

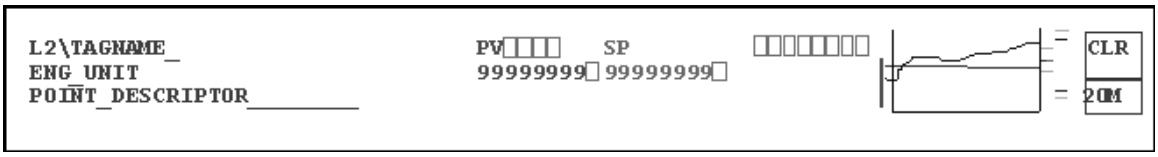


NOTE

If a state is configured as momentary state, MOM will be displayed in the State box.

For a PM/APM/HPM Analog Output Point

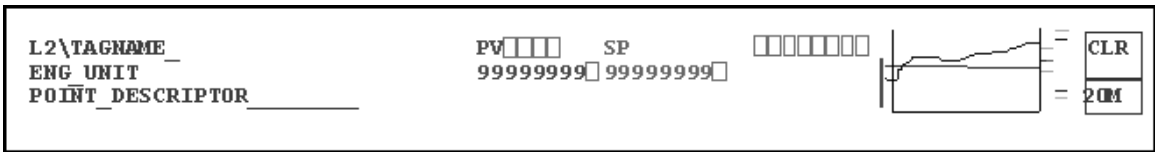
For a PM/APM/HPM Analog Input Point



For a PM/APM/HPM Digital Input Point

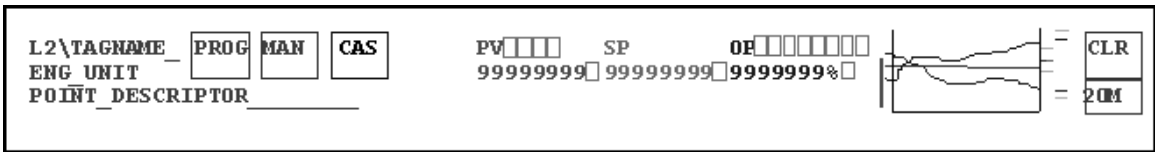
For a PM/APM/HPM Digital Output Point

For a HG Analog Input Point

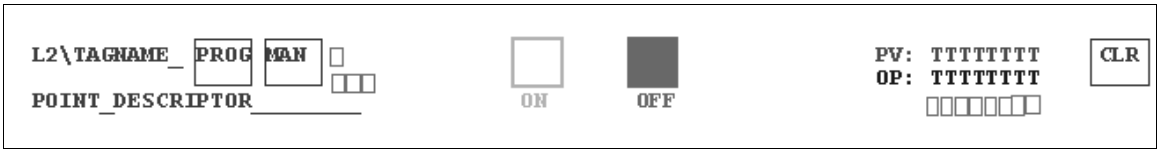


For a HG Analog Output Point

For a HG Analog Composite Point



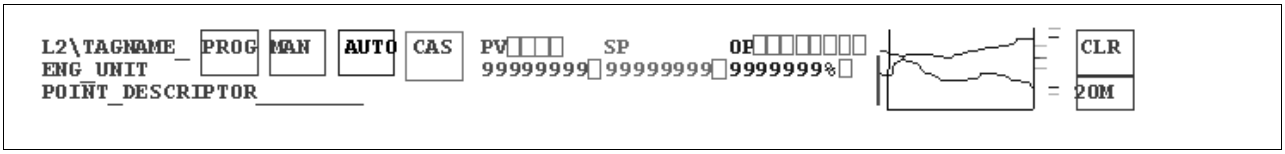
For a HG Digital Composite Point



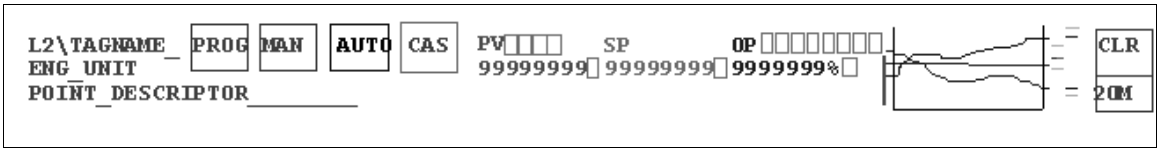
For a HG Digital Output Point

For a HG Digital Input Point

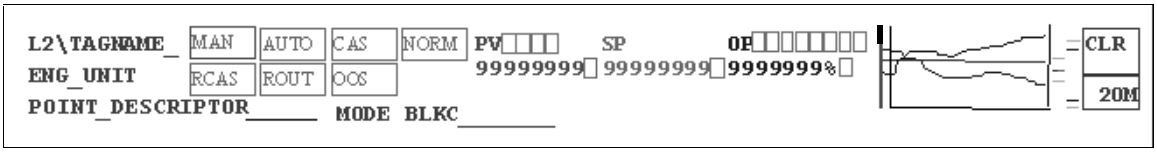
For a HG Regulatory Control Point



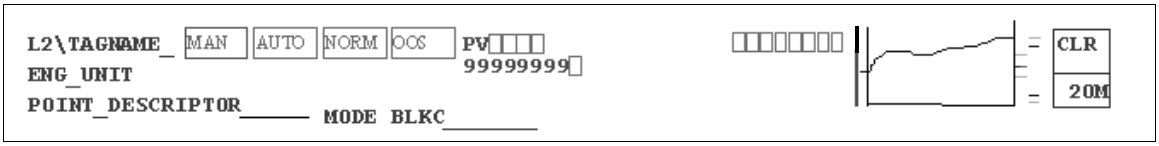
For a AM Regulatory Control Point



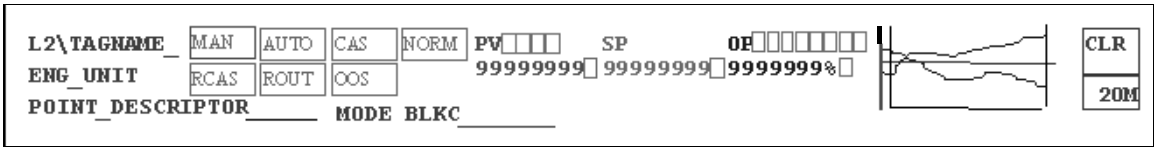
For a Fieldbus PID Point



For a Fieldbus Analog Input Point



For a Fieldbus Analog Output Point



2.9.1 Mini-Trend Graph

The trend graph consists of:

- traces of the relevant parameters PV, SP and OP values depending on the point type,
- color-coded tick marks that indicate the values of the respective alarm limits,
- a target to change the time base for the trend graph,
- an OVERVAL percentage bar representing the scaling of the trend graph for the PV and SP traces, and
- invisible scroll targets.

NOTE

The mini-trend can be used only to get an overview of the trend, to view the detailed trend the operator should invoke the Trend Display by pressing the TREND button.

Each feature in the mini-trend graph is explained in detail in the following sections.

OPERATING MODES

- The Trend Graph can be operated in two modes, i.e. SP window mode and the PV window mode. The user can specify the default Window mode (PV/SP) for the mini-trend in the Enhanced Change Zone, i.e. the default mode in which the Enhanced Change Zone should be displayed every time it is invoked. This can be done by setting the global display database variable TRWINECZ to ON/OFF for PV/SP modes.

  The user can set this in the initial target in the Custom System Status Display. Setting this to ON will select the PV as the default window mode and setting it to OFF will select SP as the default window mode. If the user does not set the global display database variable TRWINECZ, the SP window mode will be used as the default mode.
- The user can toggle between the two window modes by selecting the target on the OVERVAL bar. After each toggle the trend is re-invoked with the new scaling positions.

NOTE

The display database variable used to toggle the modes using the OVERVAL bar target is independent of the global display database variable used to select the default window mode.

OP TRACE SCALING

- The OP trace is always scaled from 0% –100 %.

PV/SP TRACE SCALING (OVERVAL BAR)

- The scaling of the PV/SP traces is depicted by the OVERVAL bar. In a scale from 0% to 100 %, the bar depicts the part that is scaled in the trend graph. The bar is displayed from the low_range% to the high_range%. The low range% to the high range% values are based on the algorithm for scaling. In the SP window mode, the OVERVAL bar is displayed in GREEN color and in the PV window mode the OVERVAL bar is displayed in CYAN color. The traces that are not within the display range (depicted by the OVERVAL bar) are not displayed in the trend graph.

An OVERVAL (percentage) bar appears on the left side of the trend graph. A brief description of this bar is given below. (Refer to the figure in the Mini-Trend Graph Section)

For example, consider a point having
Window Mode = SP Window
SPEUHI = 150
SPEULO = 50
OVERVAL = 10
SP (when Enhanced Change Zone is invoked) = 75

Since OVERVAL value is 10 %, the OVERVAL bar length is 20% (i.e., 2 * overval value).

The OVERVAL bar starts at **SP % - OVERVAL**, (25% - 10%) which is 15% above the origin in the Y-axis and ends at **SP % + OVERVAL**, (25% + 10%) which is 35% above the origin in the Y-axis.

Converting it to the Engineering units, the graph is scaled from 65 to 85, i.e.

SPEULO + 15 %* (150-50) = 65
SPEULO + 35 % * (150-50) = 85

The trend starts from the right side of the trend graph. The PV/SP trends are clipped below 65 and above 85 i.e., the PV/SP trends are zoomed between 65 to 85.

TIME BASE

- Selecting the Time Base target changes the time base value cyclically. The user selects a particular time base value by repeatedly touching this target and stopping it when the desired value is reached. The default timebase is 20M and clicking the target changes the timebase in the cyclic order as 1H, 2H, 8H, 16H, 24H, 48H, 96H, 1M, 2M, 5M and 20M.

SCROLL TARGETS

- There are invisible scroll targets provided to scroll the trend graph backward and forward. Scroll is enabled only when the time base selected is 5 Min or above, and additionally if at least one trace exists, whose data source is HM.

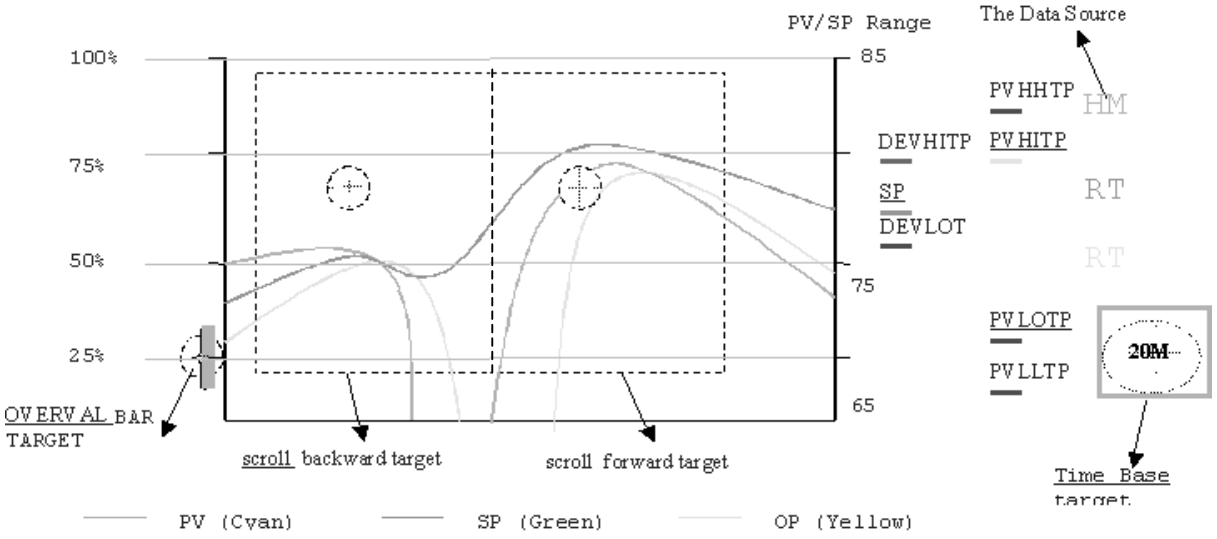
DATA SOURCE

- The Data Source for each of the trace is displayed on the right side of the Trend Graph in the trace color. When the time base selected is greater than or equal to 5 Minutes, the Data Source for all the historized traces will be HM. If the time base selected is less than 5 minutes all the traces will show real time trends (RT).

COLOR CODED TICK MARKS

- Color-coded tick marks, reflecting the alarm priorities are also shown on the right side of the trend graph, representing the deviation alarm limits (DEVHITP, DEVLOTP) and the SP value in the first vertical and the PV alarm limits (PVHHTP, PVLLTP, PVHITP, PVLOTP) in the second vertical. The tick marks are also displayed as per the scaling selected by the OVERVAL bar. The tick marks are not displayed for values outside the display range.
- The color of the tick mark is determined by the alarm priority set for the parameter and the color specified in the NCF for that priority. When the priority for the point is not Low, High or Emergency, the tick mark will not be displayed.
- The Tick marks will not displayed if the trip limits are NAN, or if they are not configured.
- When any values of PVHHTP, PVLLTP, PVHITP, PVLOTP are equal, the tick mark with the highest priority will be displayed at that position.
- The deviation high tick mark is displayed at position equal to SP + DEVHITP.
- The deviation low tick mark is displayed at position equal to SP - DEVLOTP.
- The DEVHITP and DEVLOTP tick marks are not displayed if the values are zero and if the values are very close to SP, the SP tick will be overlapped by the Deviation tick marks (Refer to Case 5 in Special Cases of Trend Graph).

As shown in the following illustration, the trend graph provides a graphical representation of the point's trend in detail.



2.9.2 Mini-Trend Graph Scaling

Trend graph is scaled automatically and is applicable for the PV and SP traces. Scaling is determined by the

- Value of PV/SP (PV for PV window mode, SP for SP window mode)
- OVERVAL value
- PVEUHI and PVEULO for PV window mode, and SPEUHI and SPEULO for the SP window mode. However, if the point does not have SPEUHI and SPEULO, PVEUHI and PVEULO will be used instead.

Algorithm for scaling the Trend in the Enhanced Change Zone for the PV and SP traces is explained with examples:

Case 1: If overval = 0 then
low_range% := 0
high_range% := 100

Case 2: elseif (winval% - overval) < 0 then
low_range% := 0
high_range% := MIN (100, 2 * overval)

Case 3: elsif (winval% + overval) > 100 then
high_range% := 100
low_range% := MAX (0, 100 - (2 * overval))

Case 4: else
low_range% := MAX (0, winval% - overval)
high_range% := MIN (100, winval% + overval)
endif

where winval% is the window variable that can be SPP or PVP depending on whether it is the SP window or the PV window. The SPP and PVP are defined as follows:

$$SPP = \frac{(SP - SPEULO) \times 100}{(SPEUHI - SPEULO)}$$

$$PVP = \frac{(PV - PVEULO) \times 100}{(PVEUHI - PVEULO)}$$

The SPP and the PVP are values of PV and SP while selecting the window mode using the OVERVAL bar target.

NOTE

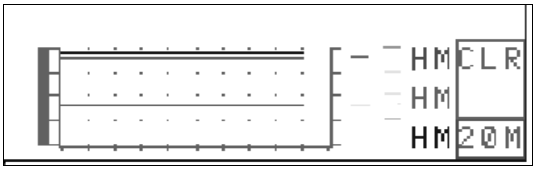
In R620, the OVERVAL parameter is not supported for the Fieldbus points and the full range of the OVERVAL will be displayed for these point types.

2.9.3 Special Cases of Mini-Trend Graph

For each of the special cases in the algorithm, an illustration is given below. The values of the parameters for the illustrations are given in the table below, and the corresponding trend display for each of the cases is also given in the subsequent subsections.

Parameter Name	**Case 1**	**Case 2**	**Case 3**	**Case 4**	**Case 5**
Window Mode	SP	PV	SP	PV	SP
PV	40.0	25.0	25.0	100.0	40.0
SP	90.0	90.0	90.0	25.0	90.0
OP %	95.0	95.0	95.0	46.0	95.0
PVEUHI	100.0	100.0	100.0	150.0	100.0
PVEULO	0.0	0.0	0.0	50.0	0.0
SPEUHI	100.0	100.0	100.0	100.0	100.0
SPEULO	0.0	0.0	0.0	0.0	0.0
OVERVAL	0	30	40	25	0
PVHHTP	100.0	100.0	100.0	75.0	100.0
PVLLTP	25.0	25.0	25.0	25.0	25.0
PVLOTP	50.0	50.0	50.0	60.0	25.0
PVHITP	75.0	75.0	75.0	75.0	100.0
DEVHITP	10.0	10.0	10.0	25.0	2.0
DEVLOTP	50.0	50.0	50.0	10.0	2.0
PVHHPR	Emergency	Emergency	Emergency	High	Emergency
PVLLPR	High	High	High	Low	High
PVLOPR	Low	Low	Low	No Action	Low
PVHIPR	Low	Low	Low	Low	Low
DEVHIPR	No Action	No Action	No Action	High	Emergency
DEVLOPR	Low	Low	Low	High	Low
PVP	---	25	---	50	---
SPP	90.0	---	90.0	---	90.0
Winval	90.0	25.0	90.0	50	90.0
Low_range%	0	0	20	25	0
High_range%	100	60	100	75	100
Low RNG in EU	0	0	20	75	0
High RNG in EU	100	60	100	125	100

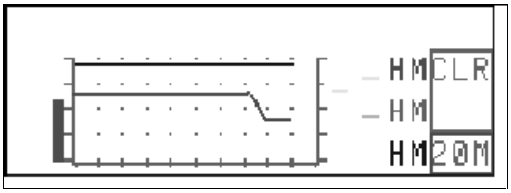
Case 1 - Consider a case where the overval = 0, low_range% := 0 and high_range% := 100. The trend graph for this case is shown below. Refer to the table in this section for more details on the parameters and their values for this case.



Observations on the resultant display:

- DEVHITP tick mark is not displayed, as the Priority is No Action.

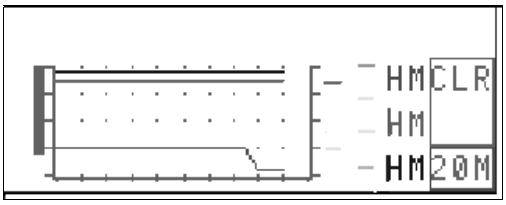
Case 2 - Consider a case where (winval% - overval) less than 0, then the low_range% := 0 and high_range% := MIN (100, 2 * overval). The trend graph for this case is shown below. Refer to the table in this section for more details on the parameters and their values for this case.



Observations on the resultant display:

- DEVHITP tick mark is not displayed as the Priority is No Action.
- SP trace is not within the display range and therefore not displayed.
- PVHHTP, PVHITP and SP tick marks are outside the display range.

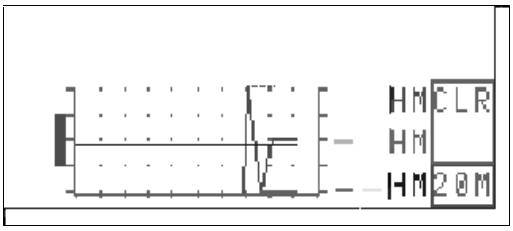
Case 3 - Consider a case where (winval% + overval) is greater than 100, then the high_range% := 100 and low_range% := MAX (0, 100 - (2 * overval)). The trend graph for this case is shown below. Refer to the table in this section for more details on the parameters and their values for this case.



Observations on the resultant display:

- DEVHITP tick mark is not displayed as the Priority is No Action.

Case 4 - Consider a case where low_range% := MAX (0, winval% - overval) and high_range% := MIN (100, winval% + overval). The trend graph for this case is shown below. Refer to the table in this section for more details on the parameters and their values for this case.



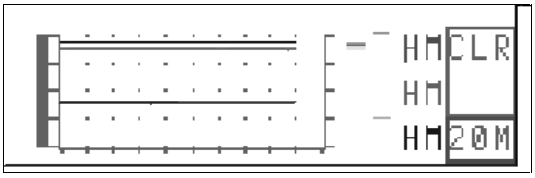
Observations on the resultant display:

- PVLLTP, PVLOTP and DEVLOTP are not within the display range.
- As PVHHTP, PVHITP values are equal and PVHHPR is greater than PVHIPR, the PVHHTP is displayed.

Case 5 – This is an additional case related to Case 1 of the algorithm, and has been added to illustrate that

- When any of the values of PVHHTP, PVLLTP, PVHITP and PVLOTP are equal, the tick mark with the highest priority will be displayed at that position.
- The DEVHITP and DEVLOTP tick marks are not displayed if the values are zero and if the values are very close to SP, the SP tick will be overlapped by the Deviation tick marks.

The trend graph for this case is shown below. Refer to the table in this section for more details on the parameters and their values for this case.



Observations on the resultant display:

- In the first vertical, the PVHHTP is displayed as PVHHPR is greater than PVHIPR, and PVLLTP is displayed as PVLLPR is greater than PVLOPR.
- The DEVLOTP and DEVHITP tick marks are very close to SP value and overlap a greater portion of the SP tick mark

2.9.4 Scenarios of PVEUHI/PVEULO and SPEUHI/SPEULO Values

The various scenarios that the user may encounter while working with the Enhanced Change Zone are given below.

- If the PVEUHI and PVEULO values are changed when the Enhanced Change Zone is displayed, the OVERVAL bar target has to be toggled or the Enhanced Change Zone re-invoked to revise the scaling based on the current values. Otherwise, the PVP/SVP (PV/SP value at the time of Enhanced Change Zone invocation) will be displayed with respect to the previous PVEUHI /SPEUHI and PVEULO/SPEULO values.
- If the PV format of the display is changed when the Enhanced Change Zone is displayed, the Enhanced Change Zone has to be re-invoked for the SP and PV values to be displayed in the new format.
- If the PVEULO and PVEUHI does not exist for a point, then the PV WINDOW scaling will be similar to the case when the OVERVAL is ZERO, i.e., the full range will be displayed. If PVEUHI is not valid for the point, then PVEUHI will be taken as 100, and PVEULO is not valid for the point and PVEULO is taken as 0.
- If PVEUHI and PVEULO are equal, the OVERVAL bar is not displayed.
- If the SPEULO and SPEUHI does not exist for a point, then the SP WINDOW scaling will be similar to the case when the OVERVAL is ZERO, i.e., the full range will be displayed (in this case PVEULO and PVEUHI will be used for scaling instead of SPEULO and SPEUHI).
- For analog input points and regulatory PV points which have their PVTV parameter mapped to the SP parameter, the mini-trend displays the PVTV parameter trace for SP. The standard trend does not display the trace.

2.10 REPLACING STANDARD CHANGE ZONE WITH ENHANCED CHANGE ZONE

To replace an existing Standard Change Zone with the Enhanced Change Zone, the appropriate method can be used:

- If the Standard Change Zone (to be replaced with Enhanced Change Zone) has been added as a sub-picture, the schematic needs to be edited using the Replace Sub-picture command and then recompiled.
- If the Standard Change Zone (to be replaced with the Enhanced Change Zone) has been added as an overlay, the schematic is edited to modify the OVERLAY(CHG_ZONE, n) to OVERLAY(ENHAN_CZ, n) and then recompiled.
- To replace the Standard Change Zone with Enhanced Change Zone in all the schematics, rename ENHAN_CZ.DO as CHG_ZONE.DO in &DSY directory of HM. The existing customer schematics, which are using Standard Change Zone as sub-picture, will not be affected.

2.11 CALL UP THE DETAIL DISPLAY

This actor is used to call up the Standard Detail display for a particular point.

With R620, the detail display actor can be used in the following scenario. R620 allows the user to specify a custom detail display for a particular point that is to be used instead of the standard detail display. This is accomplished by using the $CDETAIL parameter which has been added to each point type (HG/AM/NIM points). The $CDETAIL parameter allows the user to enter the filename (8 character) of the custom detail display (such as a custom schematic) that is to be used instead of the standard detail display. If a filename is not entered, then the standard detail display will be displayed whenever a request is made for the detail display of that particular point type. Additionally, if the filename is not a valid schematic filename, an error message appears when an attempt is made to call up the custom detail display.

When the custom detail display is invoked, a target on the custom detail display (that has been previously configured using the detail display page actor (DET_PAGE ("Point Name, Page #")) can be used to invoke a specific page of the standard detail display. The page menu on the standard detail display has a new target (CDETAIL) which allows the user to call up the custom detail display from the standard detail display. If the filename entered for the $CDETAIL parameter is not a valid schematic filename, an error message appears when an attempt is made to call up the custom detail display from the page menu.

The CURRDSP parameter will display the custom schematic name, if a custom detail display is invoked. Otherwise, CURRDSP parameter will display the same values as it does in R610.

The custom detail display parameter is included in the Exception Build files. This parameter is not supported by Find Names functionality, however it is supported by Documentation Tool.

On a GUS, pressing the DETAIL key displays the Standard/Custom detail display (on the Native window).

When Cross-Screen is active and invocation of the custom detail display fails, the error messages appear on the originating station and the standard detail appears on the destination station.

When Cross-Keyboard is active and invocation of the custom detail display fails, the error messages appear on the destination station.

CAUTION

Depending on the user's configuration of the schematic pathname in the Area Database – pathname catalog and the complexity of the custom display, the invocation of the Custom Detail Display could take considerably longer time than the invocation of the Standard Detail Display. (i. e, the Custom Detail Display is not automatically memory resident like the standard detail display). To quicken the process of invocation, it is recommended that you place the pathname of the custom detail schematic in the beginning of the pathname catalog in Area Database.

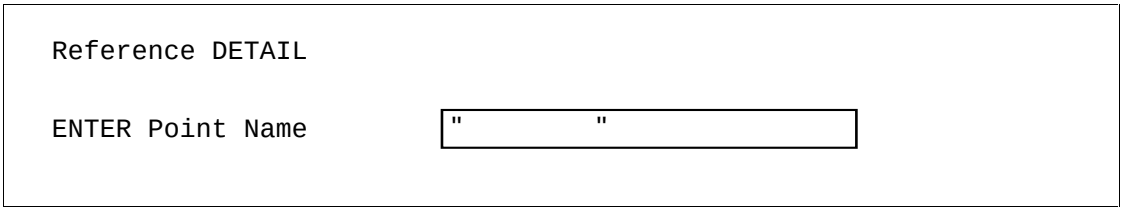
NOTE

To enable users to access the standard detail display for points that have the custom detail display configured, a button can be configured to prompt the user for the tag name and using the DET_PAGE actor, the standard detail display can be invoked.

Calling Sequence— DETAIL("POINT NAME")

Example— DETAIL("HG0501") calls up the Detail display for point HG0501.

Optional Parameter Entry Form



Parameter Explanation— POINT Name is any valid point name.

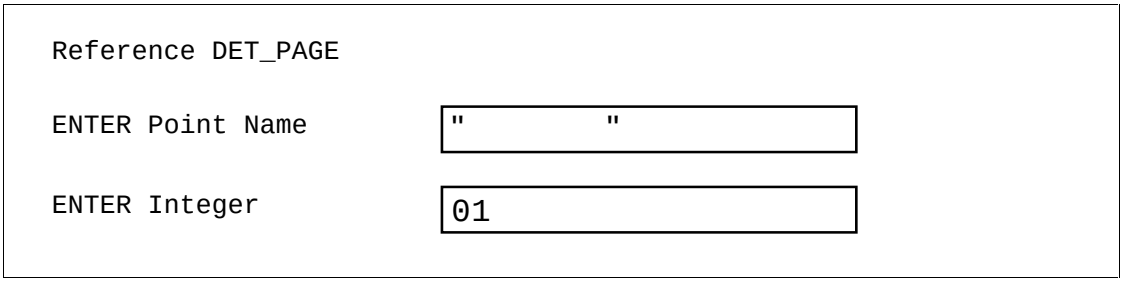
2.12 CALL UP THE DETAIL DISPLAY, PAGE N

This actor is used to call up a specified page (n) of the Detail display for a particular point.

Calling Sequence— DET_PAGE("POINT NAME",PAGE NUMBER)

Example— DET_PAGE("HG0501",2) calls up page 2 of the Detail display for point HG0501.

Optional Parameter Entry Form



Parameter Explanation— POINT Name is any valid point name. For Process Module Data Points or Array points, only the first page of the Detail Display can be called up.

2.13 CALL UP THE EVENT HISTORY DISPLAY

This actor is used to call up the Event History display.

Calling Sequence— EHR

Optional Parameter Entry Form— none

2.14 CALL UP THE FLOPPY INITIALIZATION DISPLAY

This actor is used to call up the Floppy Initialization display.

Calling Sequence— INITFLOP

Optional Parameter Entry Form— none.

2.15 CALL UP THE GROUP DISPLAY

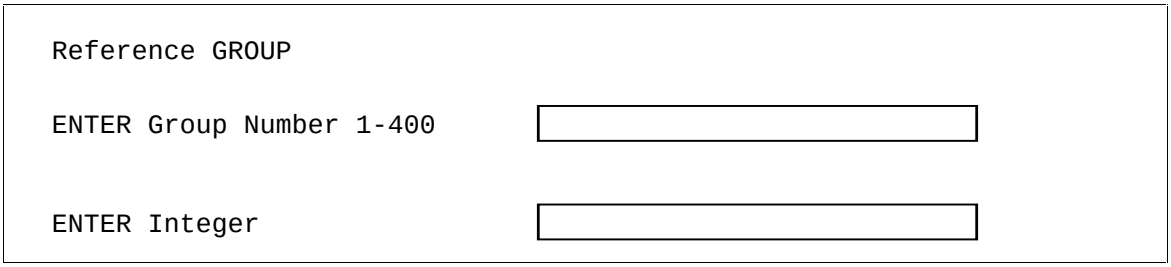
This actor is used to call up a Group display. You can specify the following items:

- Group number
- Selected point.

Calling Sequence— GROUP(GROUP NUMBER,POINT)

Examples— GROUP(01,0) calls up the Group 1 display
GROUP(02,1) calls up the Group 2 display with point 1 selected.

Optional Parameter Entry Form



Parameter Explanation— Group Number can range from 1 through 400 and refers to a configured group.

Selected Point— This entry can range from 1 through 8 and refers to one of eight points that can be configured in the group. Entering one of the eight point numbers causes that point to be selected when the Group display is called up. If this parameter = zero (0), the group is called up without a point selected.

2.16 CALL UP THE GROUP EDIT DISPLAY

This actor is used to call up the Group Edit display.

Calling Sequence— GRP_EDIT

Optional Parameter Entry Form— none

2.17 CALL UP GROUP MODULE DISPLAY

This actor is used to call up the Module Group display on a screen in the Universal Station's console. You can specify the module group number.

Calling Sequence— MODGROUP(MODULE GROUP NUMBER)

Example— MODGROUP(405) calls up the Module Group display for group 405.

Optional Parameter Entry Form

```
Reference MODGROUP


                                  
ENTER Module Group Number
```

Parameter Explanation— Module Group Number refers to a configured Module Group and can range from 401 through 450.

2.18 CALL UP GROUP TREND DISPLAY

This actor is used to call up the Group Trend display.

Calling Sequence— GRPTREND(GROUP NUMBER:1..400,SELECTED POINT:0..8)

Example— GRPTREND(3,7) Calls up the Group 3 Trend display with point 7 selected.

Optional Parameter Entry Form

```
Reference GRPTREND

ENTER Group Number

ENTER Selected Point      0
```

Parameter Explanation— Group Number can range from 1 through 400 and refers to a configured group.

Selected Point— This entry can range from 1 through 8 and refers to one of eight points that can be configured in the group. Entering one of the eight point numbers causes that point to be selected when the display is called up. If this parameter is zero (0), the display is called up without a point selected.

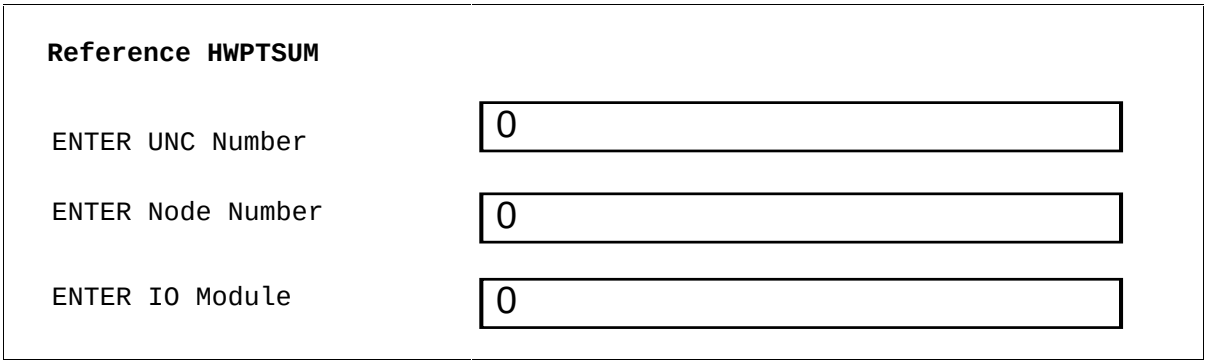
2.19 CALL UP THE HARDWARE POINT SUMMARY DISPLAY

This actor is used to call up the Hardware Point Summary display.

Calling Sequence— HWPTSUM(UCN Number,UCN Node Number, I/O Module Number)

Example— HWPTSUM(10,03,11)

Optional Parameter Entry Form



Parameter Explanation— The UCN number can range from 1 to 20

The Node number can range from 1 to 64

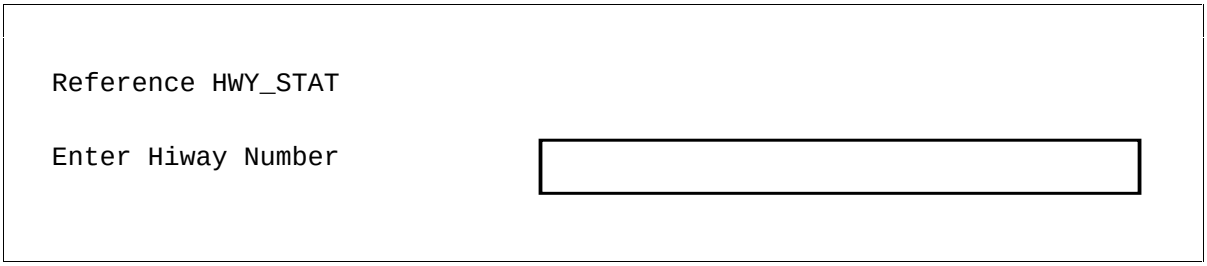
The IO Module number can range from 0 to 255

2.20 CALL UP THE HIWAY STATUS DISPLAY

This actor is used to call up the Hiway Status display.

Calling Sequence— HWY_STAT(HIWAY NUMBER)

Optional Parameter Entry Form



Parameter Explanation— The Data Hiway number can range from 0 to 100.

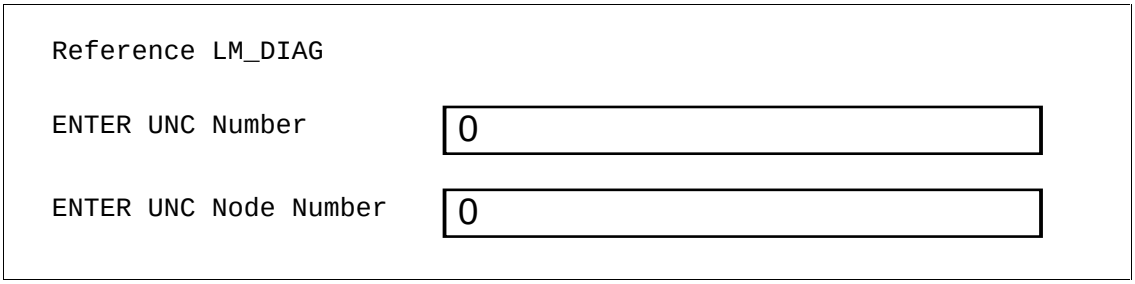
2.21 CALL UP THE LOGIC MANAGER DIAGNOSTIC DISPLAY

This actor is used to call up the Logic Manager Diagnostic display.

Calling Sequence— LM_DIAG(UCN Number,UCN Node Number)

Example— LM_DIAG(10,03)

Optional Parameter Entry Form



Parameter Explanation— The UCN Number can range from 1 to 20

The UCN Node Number can range from 1 to 64

2.22 CALL UP THE MESSAGE SUMMARY DISPLAY

This actor is used to call up the Message Summary display on a screen in the Universal Station's console.

Calling Sequence— MSGSUM

Optional Parameter Entry Form— none

2.23 CALL UP MULTIPLE SERIAL OVERLAYS

This actor is used to call up specified overlays serially.

The Multiple Overlay actor can be invoked only if a schematic is currently displayed, otherwise an error message results. If the requested overlay is found, it is called up without clearing the schematic that is currently displayed. If the specified overlay is already being displayed, a demand update is done to the overlay. If the requested overlay is not found, an error display results. If the specified overlay is not the one currently being displayed and if all previous overlays called up with the currently displayed schematic were done using the MULT_OV actor, the screen area occupied by the current overlay is cleared and the new overlay is displayed. If the OVERLAY actor, was previously used to call up any overlay with the currently displayed schematic, an error message results.

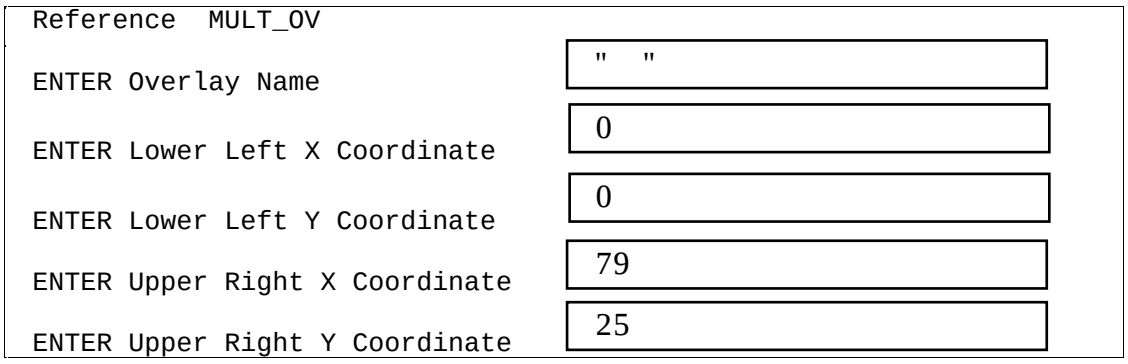
The overlay update rate is 4 seconds and the overlay has a higher update priority than the schematic. If data must be passed between the schematic and the overlay, you must use the global display database. The size of the overlay plus the schematic must not exceed the current maximum size for a schematic.

Calling Sequence—

MULT_OV("OVERLAY NAME",LOWER LEFT X,LOWER LEFT Y,UPPER RIGHT X,UPPER RIGHT Y)

Example— MULT_OV("OVERLAY1",0,0,79,5)

Optional Parameter Entry Form



Parameter Explanation

- Overlay Name is the name of the schematic that is to be called up as an overlay. It is a 1-8 character string.
- Lower Left X and Lower Left Y Coordinates (in character units) are the coordinates of the lower left corner of the region being defined. The range for X is 0-79, and for Y is 0-25.
- Upper Right X and Upper Right Y Coordinates (in character units) are the coordinates of the upper right corner of the region being defined. The range for X is 0-79, and for Y is 0-25.

Call Up Multiple Serial Overlays (Contd.)

Notes And Precautions

The MULT_OV actor itself cannot be used in a Multiple Overlay. It can be implemented from a Background Overlay, through a target in a schematic, or by use of a configurable button.

When the first Multiple Overlay is called up, the underlying schematic is not cleared. If a subsequent Multiple Overlay is called up, the screen area occupied by the previous Multiple Overlay (including any schematic objects in that area) is cleared; therefore, schematic objects should not be placed in areas where you intend to use Multiple Overlays.

The lower-left X-Y coordinates and upper-right X-Y coordinates define two corners of a rectangle (similar to the rectangle formed when building a target). The rectangle specifies that part of the overlay picture that is used to overlay a schematic. In the example MULT_OV("OVERLAY1",0,0,79,5), the lower 6 lines, 80 columns wide of the picture named OVERLAY1 is used to overlay the corresponding area of a schematic. Refer to Figure 1-4 in Section 1 for screen coordinate information.

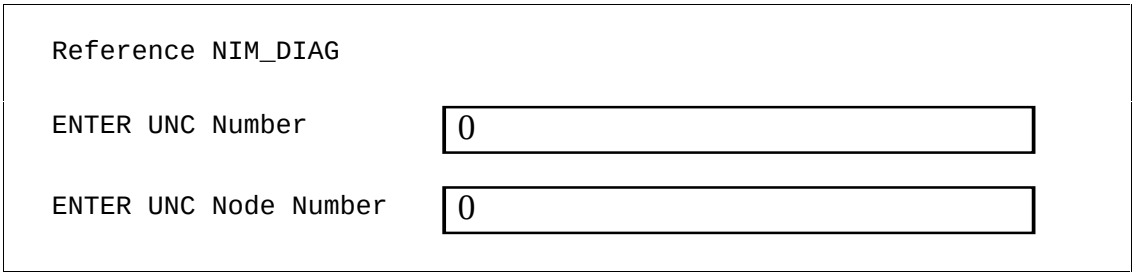
2.24 CALL UP THE NIM DIAGNOSTIC DISPLAY

This actor is used to call up the Network Interface Module Diagnostic display.

Calling Sequence— NIM_DIAG(UCN Number,UCN Node Number)

Example— NIM_DIAG(10,17)

Optional Parameter Entry Form



Parameter Explanation— The UCN Number can range from 1 to 20

The UCN Node Number can range from 1 to 64

2.25 CALL UP THE NODE POINT SUMMARY DISPLAY

This actor is used to call up the Node Point Summary display.

Calling Sequence— NODE_PS

Optional Parameter Entry Form— none

2.26 CALL UP THE ORGANIZATIONAL SUMMARY MENU

This actor is used to call up the Organizational Summary Menu.

Calling Sequence— OSUMMENU

Optional Parameter Entry Form— none

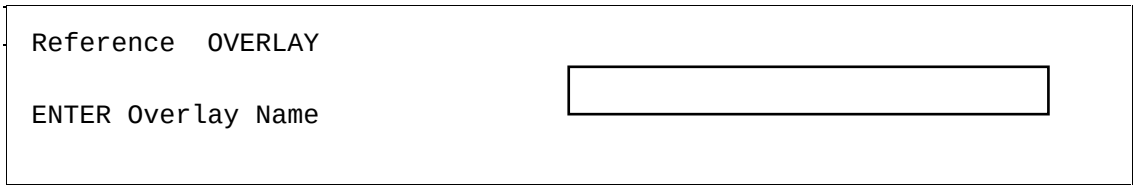
2.27 CALL UP OVERLAY

This actor is used to call up a named Overlay display. This was the original overlay actor. For new schematics, the Background Overlay (BACK_OV) or Multiple Serial Overlay (MULT_OV) are preferred.

Calling Sequence— OVERLAY("OVLYNAME")

Example 1— OVERLAY("Disp123")

Optional Parameter Entry Form



Parameter Explanation— OVLYNAME is the name of a drawing that you wish to use as an overlay. The Overlay name can be 1-to-8 characters long.

Functional Description— The Overlay actor can be invoked only if a custom graphic display is currently displayed. If found, the named overlay is displayed without clearing the display currently on the screen. Both the custom graphic display and the overlay display continue to be displayed and updated. An overlay has an update rate of four seconds and a higher update priority than the custom graphics display.

Only one overlay is allowed on the screen at a time. An error message results if the called overlay cannot be found, or if a custom graphic display (schematic) is not currently displayed. If the overlay actor is invoked when an overlay is already on the screen, an error message results unless it is the same overlay. In that case the actor causes the overlay to update. Note that the overlay cannot be independently cleared, i.e., you must call up a custom graphic display (the same or a different one) before another overlay can be used. Refer to the MULT_OV actor if you need to use different overlays with the same custom graphic display.

The overlay occupies region 3 (see Figure 1-4 in Section 1 and the Change Zone Actor).

Precautions— If you wish to pass data between the custom graphic display and the overlay, use the Global Display Database.

Example 2— OVERLAY(R_STR(35,1,8,"Enter Overlay Name",TRUE,1))

In this example, the operator can type in an overlay name through a text input port.

2.28 CALL UP OVERVIEW DISPLAY

This actor is used to call up the Overview display. Note that the Overview display is configured through the Area Database/Area Configuration Menu.

Calling Sequence— OVERVIEW

Optional Parameter Entry Form— none

2.29 CALL UP THE OVERVIEW EDIT DISPLAY

This actor is used to call up the Overview Edit display.

Calling Sequence— OVW_EDIT

Optional Parameter Entry Form— none

2.30 CALL UP THE PROCESS MANAGER MODULE/NETWORK INTERFACE MODULE DIAGNOSTIC DISPLAY

This actor is used to call up the Process Manager Module (PMM) Diagnostic display or the Network Interface Module (NIM) diagnostic display. If the Universal Control Network (UCN) Node number specified is for a PMM, the PMM Diagnostic display is called up. If the UCN Node number specified is for a NIM, the NIM Diagnostic display is called up.

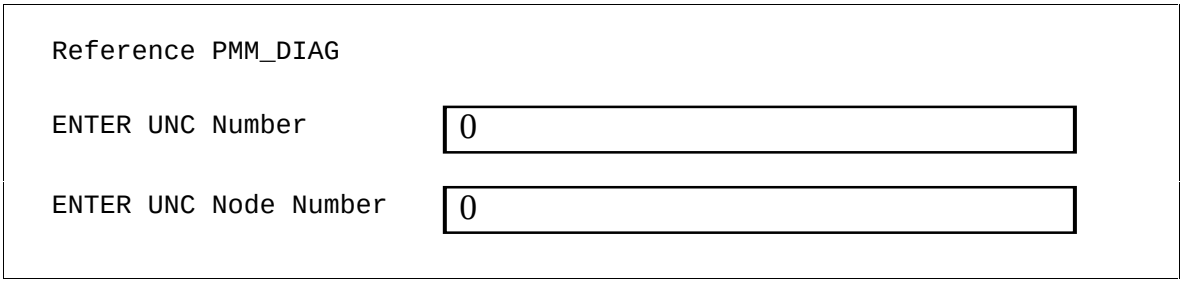
You can specify the following items:

- The UCN number
- The (UCN) Node number

Calling Sequence— PMM_DIAG(UCN NUMBER,UCN NODE NUMBER)

Example— PMM_DIAG(01,03) calls up the diagnostic display for Network 1, Node 3

Optional Parameter Entry Form



Parameter Explanation

- The UCN Number can range from 1 to 20 and specifies the Universal Control Network that contains the PMM or NIM.
- The UCN Node Number can range from 1 to 64. This parameter specifies the PMM or NIM for which diagnostics are requested.

2.31 CALL UP THE PROCESS MANAGER STATUS DISPLAY

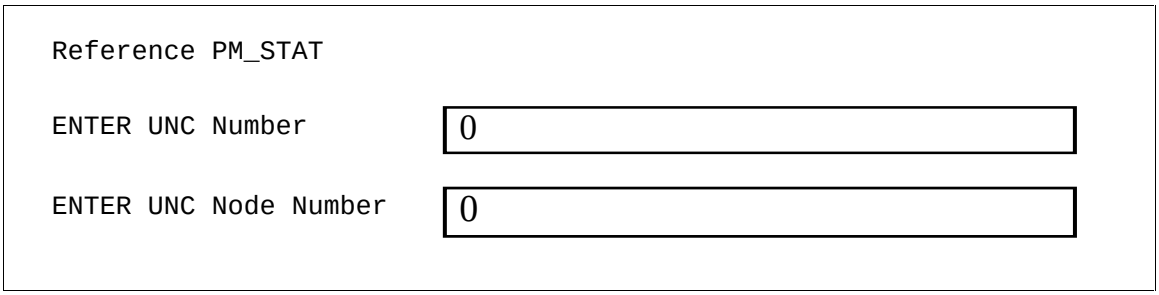
This actor is used to call up the Process Manager (PM) Status display. You can specify the following items:

- The Universal Control Network (UCN) number
- The (UCN) Node number

Calling Sequence— PM_STAT(UCN NUMBER,UCN NODE NUMBER)

Example— PM_STAT(1,3) calls up the status display for Network 1, Node 3

Optional Parameter Entry Form



Parameter Explanation

- The UCN Number can range from 1 to 20 and specifies the Universal Control Network that connects to the desired Process Manager.
- The UCN Node Number can range from 1 to 64. This parameter specifies the Process Manager for which status is requested.

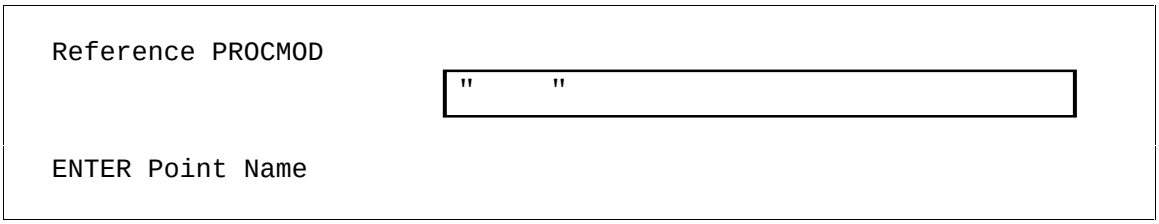
2.32 CALL UP THE PROCESS MODULE DISPLAY

This actor is used to call up the Process Module display for sequence points.

Calling Sequence— PROCMOD("POINT NAME")

Example—PROCMOD("HG0501") calls up the Process Module display for point HG0501

Optional Parameter Entry Form



Parameter Explanation— Point Name refers to any valid Process Module point. If the point name entered is not a Process Module point, the Detail display is called up.

2.33 CALL UP THE PROCESS VARIABLE RETRIEVAL DISPLAY

This actor is used to call up the Process Variable Retrieval display.

Calling Sequence— PVR

Optional Parameter Entry Form— none

2.34 CALL UP THE REAL TIME JOURNAL DISPLAY

This actor calls up the Real Time Journal

Calling Sequence— RTJ

Optional Parameter Entry Form— none

2.35 CALL UP THE REPORT MENU

This actor calls up the Report/Log/Trend/Journal Menu.

Calling Sequence— REP_MENU

Optional Parameter Entry Form— none

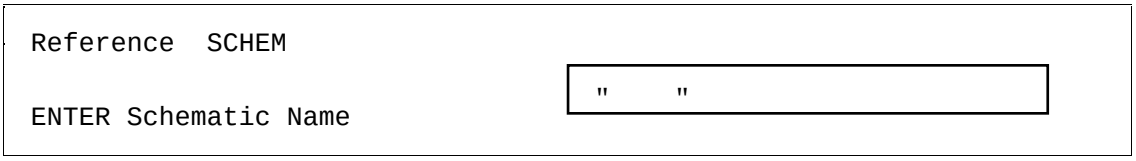
2.36 CALL UP SCHEMATIC

This type of actor allows the user to call up a specified custom graphic display.

Calling Sequence— SCHEM("SCHEMATIC")

Example— SCHEM("FRED") calls up the custom display with the filename FRED.

Optional Parameter Entry Form



Parameter Explanation— SCHEMATIC is the name of the desired display; it can range from one to eight characters.

Functional Description— When the actor executes, the named display is called up. If the display was not configured in the Area Database, or does not exist, the following error message is presented: Abstract missing or Network file error. The update rate is 4 seconds.

2.37 CALL UP THE TITLE SUMMARY DISPLAY

This actor is used to call up the Title Summary display.

Calling Sequence— TITLESUM

Optional Parameter Entry Form— none

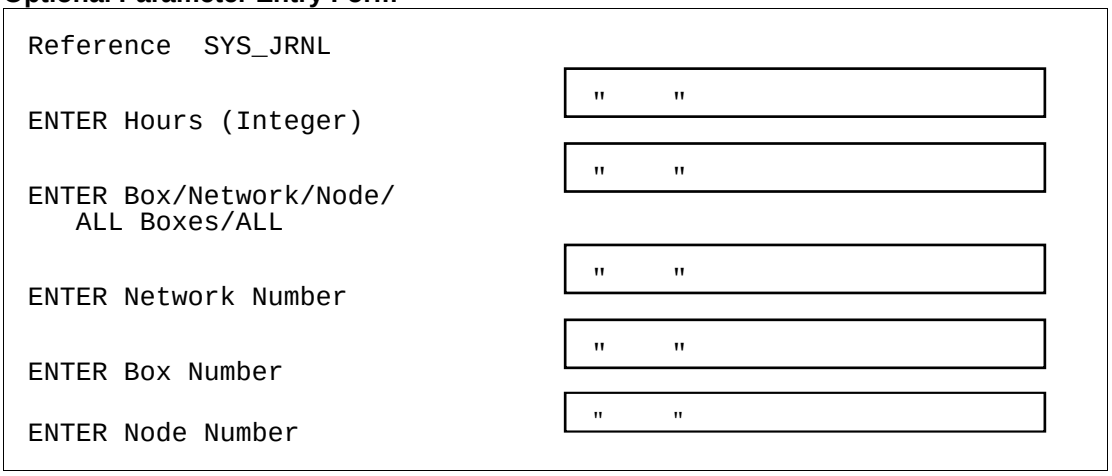
2.38 CALL UP SYSTEM STATUS JOURNAL (R600)

The new R600 actor, SYS_JRNL, can invoke different displays of the System Status journal. This actor can be added to a target of a Custom System Status Display, custom schematic, or any button in the Button Configurator file.

Calling Sequence— SYS_JRNL

Example— SYS_JRNL (t, a, b, c, d)

Optional Parameter Entry Form



Parameter Explanation—

where:

t integer (default value = 1)
represents the number of hours of data to retrieve (subtracted from current time to get start time)

a 1 - 5 (default value = 5)
1- Network Boxes, 2– Network, 3– ALL Boxes, 4– Node, 5– ALL

b 1 - 20 (default value = n/a)
Network Number (applicable when a = 1 or 2)

c 1 – 64 (default value = n/a)
Box Number (applicable when a = 1)

d 1-96 (default value = n/a)
Node Number (applicable when a = 4)

Functional Description— The parameters for this actor are optional. If no parameters are specified, then the default parameter settings, as specified above, will be used.

NOTE

When a custom schematic target or configured button invokes the System Status journal actor, execution of the PRIOR DISPLAY button will return to the display previously being viewed when the System Status journal was invoked, NOT the Event History Retrieval Menu.

2.39 CALL UP THE TREND OVERVIEW DISPLAY

This actor is used to call up the configured Trend Overview display on a screen in the Universal Station's console.

Calling Sequence— TRENDOVW

Optional Parameter Entry Form— none

2.40 CALL UP THE UCN IO MODULE DIAGNOSTIC DISPLAY

This actor calls up a Universal Control Network's IO Module Diagnostic Display
You can specify the following items:

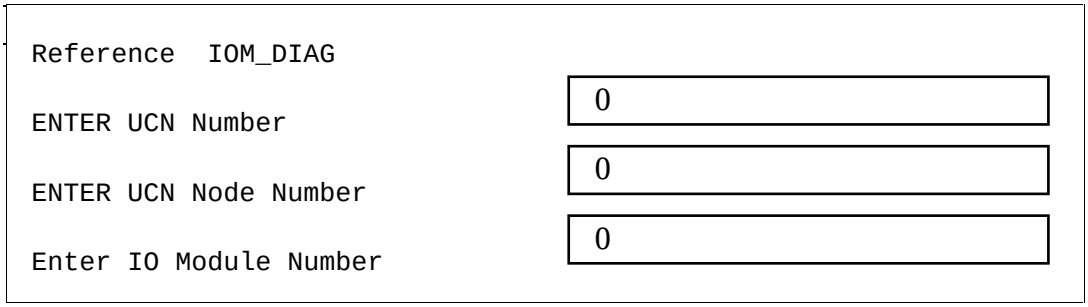
- A Universal Control Network (UCN) number
- A UCN Node number
- A IO Module number

Calling Sequence—

IOM_DIAG(UCN NUMBER,UCN NODE NUMBER,IO MODULE NUMBER)

Example—IOM_DIAG(01,03,06) calls up the IO Module Diagnostic Display for Network 1, Node 3, IO Module Number 6.

Optional Parameter Entry Form



Parameter Explanation

- The UCN Number can range from 1 to 20 and specifies the Universal Control Network that connects to the Process Manager containing the desired IO Module.
- The UCN Node Number can range from 1 to 64. This parameter specifies the node number of the Process Manager that contains the desired IO Module.
- The IO Module Number can range from 1 to 40. This parameter specifies the IO Module for which the diagnostic display is requested.

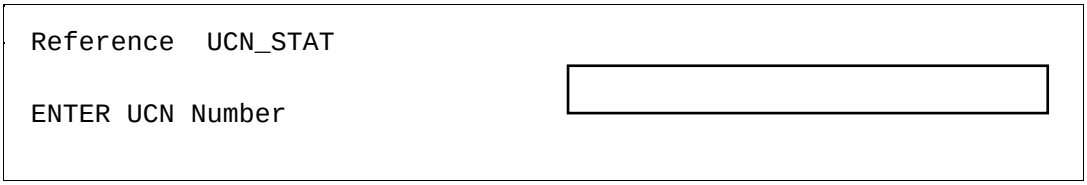
2.41 CALL UP THE UCN STATUS DISPLAY

This actor is used to call up the Universal Control Network's Status display.

Calling Sequence— UCN_STAT(UCN NUMBER)

Example— UCN_STAT(01) Calls up the UCN Status display for Network 1.

Optional Parameter Entry Form



Parameter Explanation— The UCN_Number specifies the number of the Universal Control Network for which status is requested.

2.42 CALL UP THE UNIT POINT SUMMARY DISPLAY

This actor is used to call up the Unit Point Summary display.

Calling Sequence— UNIT_PS

Optional Parameter Entry Form— none

2.43 CALL UP UNIT (MODULE) SUMMARY DISPLAY

This actor is used to call up the Sequence Unit Summary (Module Summary) display on a screen in the Universal Station's console. Note that the Unit Summary display is configured through the Area Database/Area Configuration Menu.

Calling Sequence— UNITSUM

Optional Parameter Entry Form— none

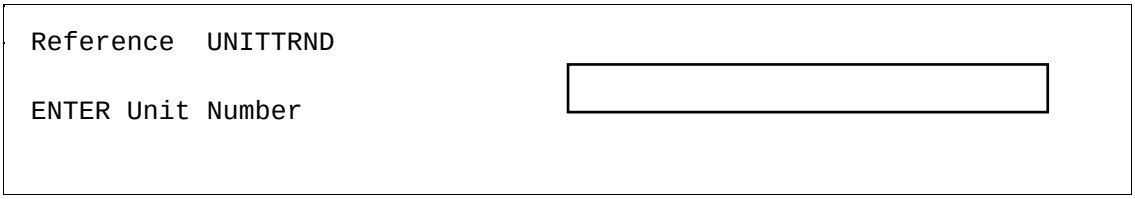
2.44 CALL UP UNIT TREND DISPLAY

This actor is used to call up the Unit Trend display.

Calling Sequence— UNITTRND(Unit Number:1..36)

Example— UNITTRND(01)

Optional Parameter Entry Form



Parameter Explanation— Unit Number is the unit that contains the points to be trended.

2.45 CALL UP THE USAGE POINT SUMMARY DISPLAY

This actor is used to call up the Usage Point Summary display.

Calling Sequence— USAGE_PS

Optional Parameter Entry Form— none

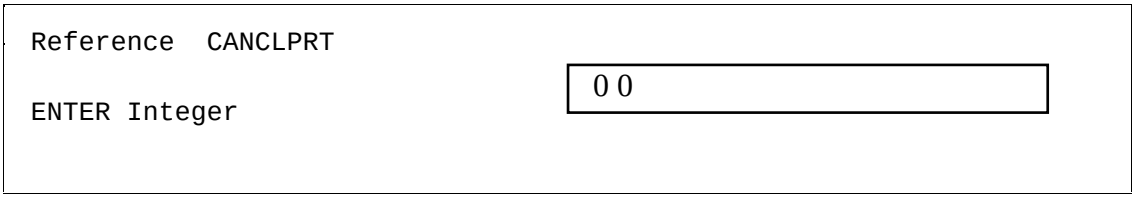
2.46 CANCEL PRINT

This actor is used to cancel a printout.

Calling Sequence— CANCLPRT(Printer Number)

Example— CANCLPRT(01)

Optional Parameter Entry Form



Parameter Explanation— Printer Number is an integer that specifies the number of the printer on which printing is to be canceled.

2.47 CLEAR THE CHANGE ZONE

This actor can be used to clear the Standard and Enhanced Change Zones. It has the same effect as the CLR target on the change zone (see CHGZONE actor).

Calling Sequence —CLEAR_CZ

Optional Parameter Entry Form —None

2.48 CLEAR PROMPT REGION

This actor allows you to clear the prompt region of the display.

Calling Sequence— PROMPT_C

Optional Parameter Entry Form— none

2.49 CLEAR THE SCREEN

This actor clears the screen. It also clears the schematic object files in US memory. Refer to Section 4.1 in the *Picture Editor Data Entry* manual for more information.

Calling Sequence— CLR_SCRN

Optional Parameter Entry Form— none

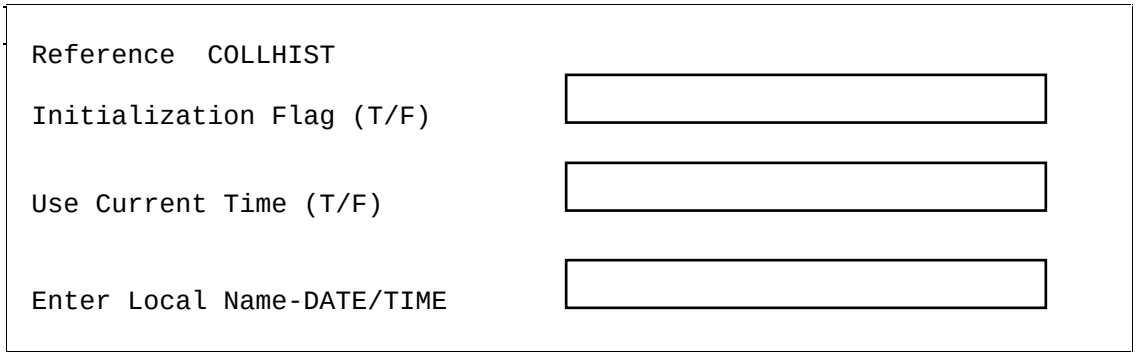
2.50 COLLECT HISTORY

This actor initiates processing of a pictures history collectors. Typically it is triggered by an operator after entering all entities and parameters needed by the history collectors.

Calling Sequence— COLLHIST (Init_Flag, Curr_Flag, Local_Name)

Examples— COLLHIST(xx,yy,zz)

Optional Parameter Entry Form



Parameter Explanation

- Initialization Flag —True (T) causes old history values to be replaced with *** while the new value is being collected. False (F) allows the old values to be displayed until they are replaced with newly retrieved values.

- Current Time — True (T) causes the current time to be used as the reference time. False (F) means to get the starting date and time from the Local Name variable (below).

- Local Name— Name of the Local DDB variable that contains the reference Date/Time. Note that the Date/Time is typically obtained through operator input actors and stored in the DDB variable using the Combine Date/Time actor (C_DATTIM). There must be an entry even if the "Use Current Time " parameter = True.

Reference time establishes a date/time for the history collector. Arguments to the collector can offset the collection period from reference time (for example, second hour of the day).

CAUTION

Do not use the COLLHIST actor in an Initial Action. If you do, the error message: `Invalid COLLHIST Usage` results and the actor does not execute. Then the picture is displayed.

2.51 CROSS SCREEN (CONDITIONING)

This actor is used in cross station requests. The actor conditions the screen such that the next call up request is directed to that screen.

Calling Sequence— CROSSCRN(SCREEN NO)

Example— Crosscrn(02);UNITSUM calls up the Unit Summary display on screen 2.

Optional Parameter Entry Form

```
Reference  CROSSCRN

ENTER the Screen Number
```

Parameter Explanation— Screen Number ranges from 1 through 10 and represents the screen numbers for a given operator's console.

Functional Description— The Cross Screen actor conditions the Universal Station to direct its next keystroke or target display-invocation action to the requested screen. The specified screen number must exist for the console in use or an error will occur when the target is activated. At

execution time, both stations must be running the Operator Personality or the Universal Personality (UP). If the UP is running, both stations must be executing Operator functions (i.e., Operator Personality displays must be selected).

Note that although it is possible to construct a target with just the Cross Screen actor, it is not intended to operate that way. For targets in a custom graphic display, a Cross Screen actor should always be followed by a display invocation. Assignment for configurable buttons can contain just the Cross Screen actor.

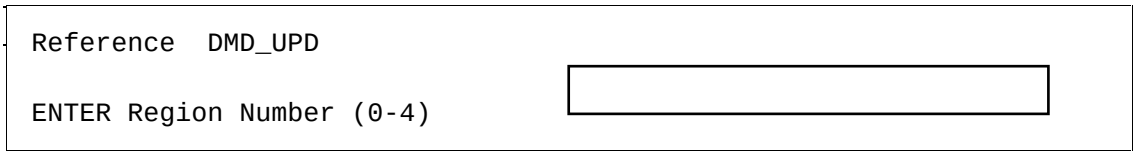
2.52 DEMAND UPDATE DISPLAY

This actor permits the user to demand a display update.

Calling Sequence—DMD_UPD(REGION:0..4)

Example—DMD_UPD(2) updates the schematic region of the display (see Figure 1-4 in Section 1).

Optional Parameter Entry Form



Parameter Explanation— Region Number refers to the region number on the screen and can range from 0 through 4 (see Figure 1-4 in Section 1). A region number = 0 means use the region where the target is located (at operating time).

For a Configurable button to work properly, you must specify a region number other than zero (0).

2.53 DISPLAY FILE

This actor allows you to display the contents of a text file (up to 80 characters long), such as a report that was written to a file by way of a virtual printer. See also the related actor: Print File.

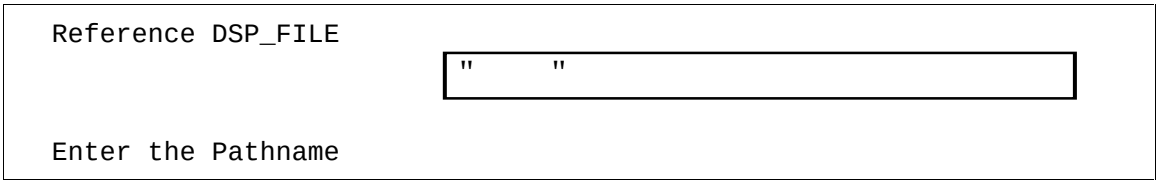
NOTE

This actor will not access files on remote LCNs via Network gateways. It does work locally. The system returns the error “INVALID PATHNAME SYNTAX” when a remote system file is entered.

Calling Sequence— DSP_FILE("Pathname")

Examples—DSP_FILE("NET>VIRT>VFIL20.61")
DSP_FILE(R_STR(1,10,21,"ENTER FILE NAME",TRUE,0))

Optional Parameter Entry Form



Parameter Explanation—Pathname is the path to a text file, especially those created by Report Output Functions, (i.e., created by printing by means of any of the virtual printers 11 through 30). Virtual printer numbers are associated with specific files by Virtual Printer Configuration files on the &ASY volume. Refer to the *Engineer's Reference Manual* for additional information on Report Output functions.

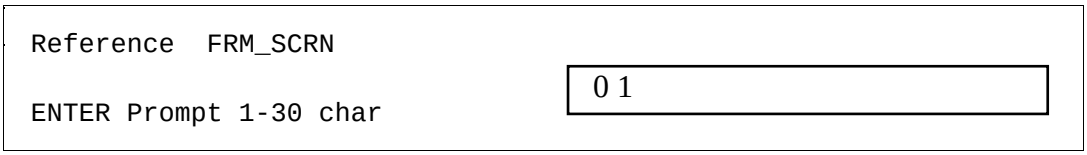
2.54 FROM SCREEN (DISPLAY TRANSFER)

This actor is used in cross-station requests. The From-Screen actor requests that a display from another screen in the same Universal Station console be called up on this screen.

Calling Sequence— FRM_SCRN(SCREEN NO)

Example— FRM_SCRN(01) calls up the same display that is on screen 1.

Optional Parameter Entry Form



Parameter Explanation— Screen Number ranges from 1 through 10 and represents the station numbers for a given Universal Station console.

Functional Description— The From-Screen actor causes the display on another screen in the same console to appear on the local screen. The display is presented in its initial invocation form. That is, selected points will not appear selected. Exceptions to this are documented with that particular display. The requested screen number must exist for the console.

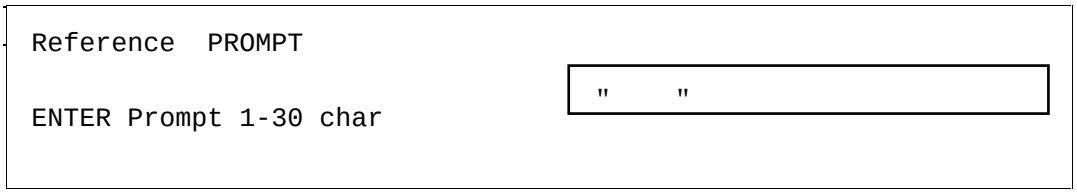
2.55 OUTPUT A PROMPT MESSAGE

This actor allows you to specify a message that will be presented in the prompt region of the display.

Calling Sequence— PROMPT("PROMPT STRING")

Example— PROMPT("CALL SUPERVISOR")

Optional Parameter Entry Form



Parameter Explanation— Prompt represents a string of 1 to 30 characters that will be sent to region 1. Also refer to the actor, PROMPT_C (Clear The Prompt Region).

2.56 PALETTE

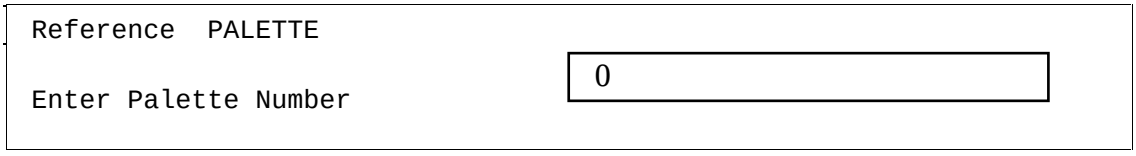
The actor causes the custom graphic display (e.g., schematic) to be displayed with a specified color palette (at operating time). You can use this actor to call different palettes without changing the schematic. A palette is a combination of a specific background and 16 colors available for use with that background. Note that there is also a Set Palette command.

Calling Sequence—PALETTE(Palette Number)
where palette number ranges from 0–16 (0 = default)

Examples—PALETTE(14)

S_INT(INT01,14);**PALETTE**(G_INT(INT01))
S_INT(INT01,R_INT(0,0,30,"Enter the Palette Number",TRUE,0));**PALETTE**(G_INT(INT01))

Optional Parameter Entry Form



Parameter Explanation—The palette number ranges from 0–16. Palette number 0 is the default. The traditional default palette consists of a black background with eight displayable colors in two intensities. Refer to the Palette command for more information.

2.57 PRINT FILE

This actor allows you to print the contents of a text file (up to 132 characters long), such as a report that was written to a file by way of a virtual printer. See also the related actor: Display File.

NOTE

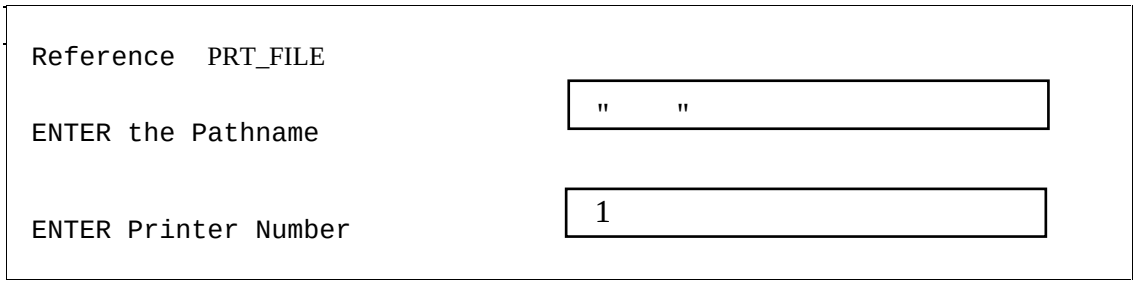
This actor will not access files on remote LCNs via Network gateways. It does work locally. The system returns the error “INVALID PATHNAME SYNTAX” when a remote system file is entered.

Calling Sequence— PRT_FILE("Pathname",Printer Number)

Examples— PRT_FILE("NET>VIRT>VFIL33.DR", 01)

PRT_FILE(R_STR(1,10,21,"ENTER FILE NAME",TRUE,0),R_INT(1,2,2,"ENTER PRINTER NUMBER",TRUE,0))

Optional Parameter Entry Form



Parameter Explanation

- Pathname is the path to a text file, especially those created by Report Output Functions (see DSP_FILE)
- Printer Number is the printer to be used, 1-30. It can be a virtual printer.

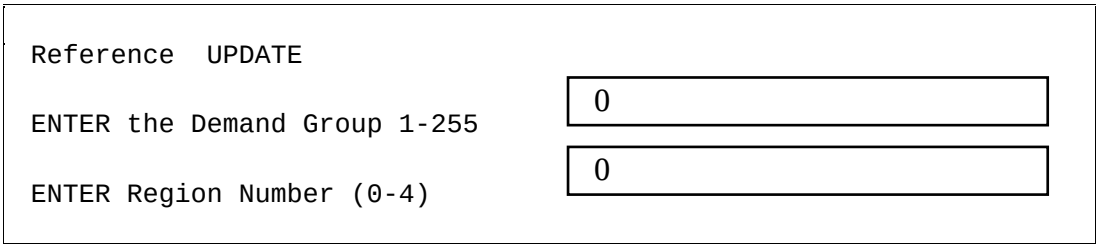
2.58 UPDATE THE DISPLAY

This actor allows you to demand a display update.

Calling Sequence— UPDATE(DEMAND GROUP,REGION NO.)

Example— UPDATE(0,0) updates data in the region where the target is located.

Optional Parameter Entry Form



Parameter Explanation

- Demand Group defines specific categories of points that are to be updated (refer to the Picture Editor Command, Set Collection).

  -1 means collect only local variables
  0 means collect all data
  1 through 255 means update only that specific collection group.

- Region number ranges from 0-4 and refers to the region number in a display (see Figure 1-4 in Section 1). A Region number = 0 indicates that the region where the touch target is located will be used (at run time).

  For a configurable button to work properly, you must specify a region number other than zero (0).

2.59 USER CHANGE ZONE

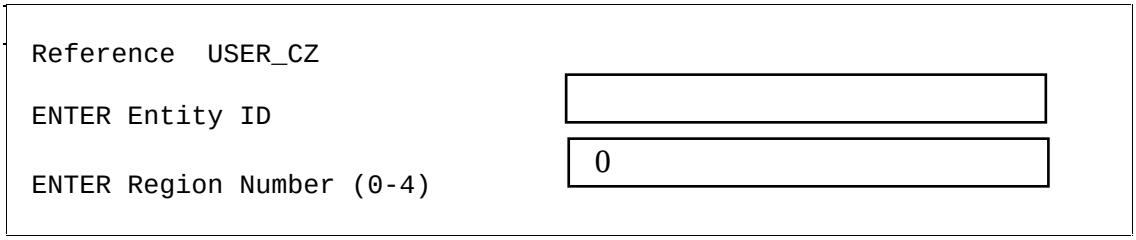
This actor allows use of the MAN, AUTO, NORM, SP, OP, and RAMP buttons with the specified point. When this actor is activated, i.e., with the Operating Personality running, a Text Input Port (TIP) for data entry appears in the top 2-line screen region. You can use the SP or OP keys to select the point's parameter and then either enter a value in the TIP or use the RAMP buttons to change values. You can select modes by pressing the mode buttons.

The actor stores the point and its associated data into the standard change zone database and allows standard change zone data to be used with custom change zones. You can display data using parameters with the $CZ_ENTY entity id.

Calling Sequence— USER_CZ(POINT NAME, REGION)

Example— USER_CZ(A100,3)

Optional Parameter Entry Form



Parameter Explanation

- Entity ID is any valid point name.
- Region is the region number where the user change variables are displayed. See Figure 1-4 in Section 1 for region definition.

Example— A target in the display contains the actor USER_CZ(A100,3). The operator activates this target, and selects one of the mode or parameter buttons. The point name and mode or parameter value is displayed (on an overlay in this case). The operator can then change the mode/parameter value with the mode/ramp keys.

$CZ_ACT— Refer to the CHGZONE actor for an explanation of this parameter.

3 KEYBOARD EVENT ACTIONS

This section describes actors that send keystrokes from one station to another and that allow you to create targets that act the same as a key on the Universal Station's keyboard.

3.1 CROSS KEYBOARD

This actor allows keystrokes to be sent from one station to another as if the sending keyboard was connected to the receiving station. Both stations must be in the same console. The receiving station can have a keyboard or it can be a station that has no keyboard. The actor was specifically developed to support stations that do not have a keyboard, such as an upper tier station.

The keystroke connection between stations is established when the actor is invoked and broken either by a time-out period or by pressing the DISCONNECT key at the sending console, whichever occurs first. The time-out period and disconnect key are specified in arguments to the CROSSKBD actor. Cross-Screen requests to the sending and receiving stations are rejected until the connection is terminated.

The connection is one way (no status or other information is returned to the sending keyboard). While a keystroke bridge is established, if stations are in different areas and a Configurable button is pressed on the sending keyboard, the action configured for the receiving station's button is attempted at the receiving station.

The Cross Keyboard actor is intended to provide on-process Operator Keyboard input and is not intended for off-process Engineering Keyboard input.

CAUTION

Once a connection is established, the sending station cannot be used for control until the connection is broken. This also means that the touchscreen does not function on the sending station until the connection is broken. There is one exception: if "PICK" has been selected for the Keyboard Key To Disconnect parameter, selecting any screen target at the sending station or pressing the SELECT key (regardless of cursor position) at the sending station will break the connection.

Calling Sequence— CROSSKBD(STN_NO,DISC_KEY,TIME,ALLOW_CON)

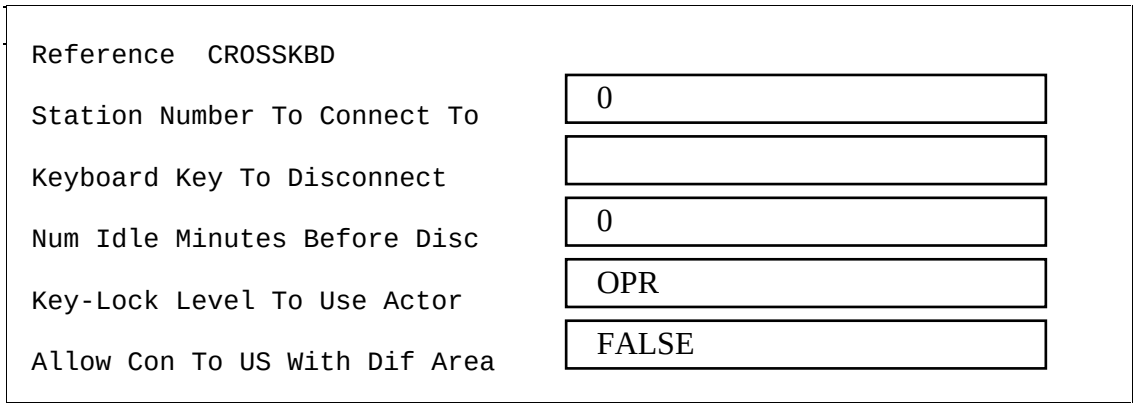
Example 1— CROSSKBD(8,CNF_7,2,OPR,FALSE)

In this example, a keyboard bridge is opened from the station where the actor is invoked to Station 8 for 2 minutes. Station 8 is in the same area. Pressing configurable button 7 will disconnect the bridge prior to timeout. Configurable button 7 may or may not be configured for another use when the Cross Keyboard function is not active.

Example 2— CROSSKBD(8,PICK,2,OPR,FALSE)

This example is the same as the previous one except selecting any sending screen target either by touch or with the cursor and SELECT key will disconnect the bridge prior to timeout.

Optional Parameter Entry Form



Parameter Explanation:

Station Number to Connect To— The station number that is to receive the keystrokes. The range is 1-10. It cannot be the same number as the sending station.

Keyboard Key To Disconnect— The key at the sending station that breaks the keystroke bridge between stations. Appendix C contains a list of standard key IDs.

CAUTION

Because some keys produce the same internal code as other seemingly unrelated keys especially when more than one keyboard panel is present, we suggest that configurable buttons 7 - 37 be used for the disconnect action (these buttons produce unique codes). Refer to the Button Configuration publications, if necessary, to identify these buttons.

For example, the BATCH key on the classic operator keyboard produces the same internal code as produced by the lower case d on the associated engineer's keyboard; therefore, if BATCH is configured as the Cross Keyboard disconnect key, typing a lower case d during a Cross Keyboard session will cause the connection to terminate.

In addition to the standard keys listed in Appendix C, the ID PICK can be used. PICK indicates that (at the sending station)—

- any screen target was selected, either by touching it or by placing the cursor on the target and pressing the SELECT key
- the SELECT key was pressed (regardless of cursor position).

NOTE

Using the DELETE key to disconnect the Cross Keyboard actor will not execute the disconnect. When DELETE is configured as the disconnect key, pressing SELECT will break the keystroke bridge.

Num Idle Minutes Before Disc— The time in minutes after establishing the keystroke bridge until it is automatically disconnected. The range is 0 - 60 minutes (0 = timer disabled).

Key-Lock Level To Use Actor— The minimum key lock level required at the sending station to invoke the CROSSKBD actor. The choices are

- OPR - Operator
- SUP - Supervisor
- ENGR - Engineer
- VIEW - View

Allow Con To US With Dif Area—

- TRUE allows connection to a receiving station that is running in a different area.
- FALSE means do not allow connection to a station that is running in another area. If this parameter is false, an area change at the receiving station breaks the cross keyboard connection (even if the change is to the same area). This protects against a change in area button configuration that might cause uncertain results.

Note that—

The Cross Keyboard Actor will not establish a connection if the receiving station is already connected to a different station by the Cross Keyboard Actor or the receiving or sending station is in a RULA session. RULA is a TDC3000^X option.

The Cross Keyboard connection will disconnect if any of the following conditions exist:

- The sending or receiving station attempts a RULA connection.
- The receiving station performs an area change (and the parameter `Allow Con To US With Dif Area` is set to FALSE).
- The sending or receiving station is shut down.

The disconnect key is processed by the sending station only and is not sent to the receiving station.

CAUTION

If you configure a disconnect key that does not exist on the sending station, the only way to break the connection is to wait for the timeout (if configured) or to reset the node. To provide maximum flexibility across various keyboard combinations, no keys are "blocked" from Cross Keyboard usage. This flexibility requires some caution on the part of the user.

For example, an engineer builds a schematic with a target that initiates a Cross Keyboard connection with no timeout and Function Key 1 (F1) as the disconnect key. If that target is selected at a station that has only an operator keyboard, there is no way to produce the F1 key code for disconnect and, because there is no timeout, the only way to break the connection is to reset the node.

Using a key as the Cross Keyboard disconnect key does not affect that key's action outside of the Cross Keyboard connection.

For example—

Suppose Configurable Button 7 (AN_CNF7) is used to invoke (point) Group_1. A schematic invokes the Cross Keyboard actor with AN_CNF7 as the disconnect key. Pressing Configurable Button 7 will cause the Cross Keyboard connection to terminate because the Cross Keyboard actor intercepts the key. After the disconnect, Button 7 will again invoke Group_1. This implies that Configurable Button 7 cannot be used to invoke Group_1 on the target node during a Cross Keyboard connection.

Changes to the keylock level at the sending station are executed on the receiving station as if the Keylevel Change actor had been called. This means the key level on the receiving station is changed only temporarily and is reset to the hardware key level in the same way as the $KEYCHG actor performs. The software key level at the sending station is reset to its hardware key level when the connection is broken.

Using DDB Variables with the CROSSKBD actor —

CAUTION

If Picture Editor DDB enumeration variables are used to supply the value for the CROSSKBD parameters, the DDB variables' default values may produce unexpected results If the actor is invoked before intended values have been stored into the DDB variables.

The picture editor provides a set of local DDB variables of type "enumeration" and of type "integer" (among others). The default value for the enumeration variables is "NULL_SET", which has a value of zero, and the default value for the integers is zero. These DDB variables can be used to supply the values for the parameters of the CROSSKBD actor as shown in the following example:

```
CROSSKBD(G_INT(INT01), G_ENM(ENM01), G_INT(INT02), OPR, FALSE);
```

In the example, INT01 holds the value of the station number. INT02 holds the number of idle minutes before disconnect, and ENM01 holds the value of the disconnect key. If the

actor is invoked before values have been stored to INT02 and ENM01, the default values will be used. The “idle minutes” will be zero, meaning that the connection will never time out. A zero inadvertently selects Configurable Button number 1 as the disconnect key (which provides a way to break the connection).

NOTE

For Release 520 and later systems, journaling of keylevel changes initiated by this actor can be suppressed by a System Wide Values option in the Network Configuration File.

3.2 Enter

Simulates the keyboard's ENTER key.

Calling Sequence— ENTER

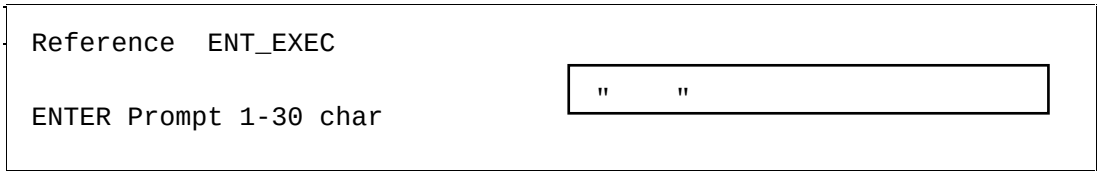
3.3 ENTER TO EXECUTE

This actor forces the operator to press the ENTER key to continue execution of an action sequence. A prompt message of up to 30 characters appears in region 1 (see Fig. 1-4 in Section 1). Use the Clear Prompt actor to remove the message later.

Calling Sequence— ENT_EXEC("PROMPT MESSAGE")

Example— ENT_EXEC("Press ENTER To Continue")

Optional Parameter Entry Form



Parameter Explanation—The Prompt Message is any string of 1 to 30 characters.

3.4 GET PRIMMOD

This actor retrieves the Primmod value assigned to the lamp-specific data field of a configurable button on the Universal Station where this actor executes. For this discussion, an $MPROD string is equivalent to a Primmod name.

The configurable button must be one of the 40 keyboard buttons with LED Alarm indicators and the Primmod value must have been previously stored.

The Primmod value returned is a string of 1 - 16 characters, null characters, or spaces.
If a Primmod value was not configured to drive the Lamp Specific Data section of the specified button, blanks are returned.

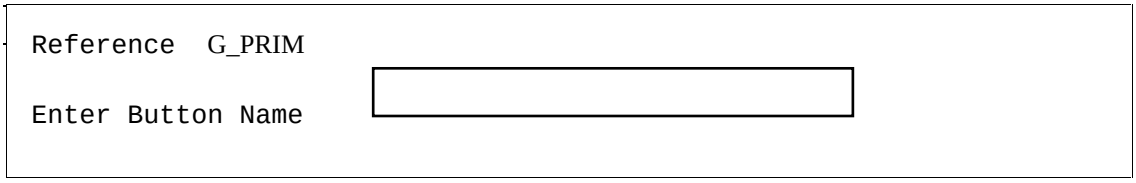
This actor is valid for Release 520 and later systems.

Calling Sequence— G_PRIM("button")
A string value is returned.

Example— S_STR(STRING01,**G_PRIM**("AN_CNF46"))

In this example, the primmod value associated with configurable button 46 is fetched and stored in the DDB String variable STRING01.

Optional Parameter Entry Form



Parameter Explanation—

Button is a symbolic button name expressed as a 1 - 8 character string. If an invalid symbolic name is specified, the actor will fail to execute and the message INVALID BUTTON NAME is displayed.

Standard default symbolic button names are listed in Appendix C.

NOTE

The Default Symbolic Button names may have been changed by a user-created Button Name File.

3.5 KEYLEVEL CHANGE

This actor sets the internal keylevel status at the Universal Station to a specified level. The permissible levels are: ENGR, SUP, OPR, and VIEW.

The $KEYCHG actor can be used to allow an operator to make temporary changes to an engineering or supervisory level parameter while the actual keyswitch remains in Operator position. Or, it can permit an actor string to have a different access level than that specified by the keyswitch. All actors following execution of $KEYCHG have the access specified by level.

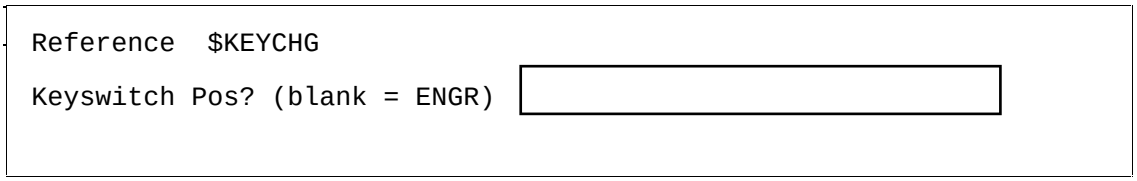
The internal keylevel is stored in the global DDB location $KEYLEVL. You can use the Keylevel Reset actor to return the keylevel status to match the physical key position.

Calling Sequence— $KEYCHG(level)

Example—

**$KEYCHG**(ENGR);SS_REAL(A100.PVHITP,90.0);SS_REAL(A100.PVEUHI,100.0);$KEYRST

Optional Parameter Entry Form



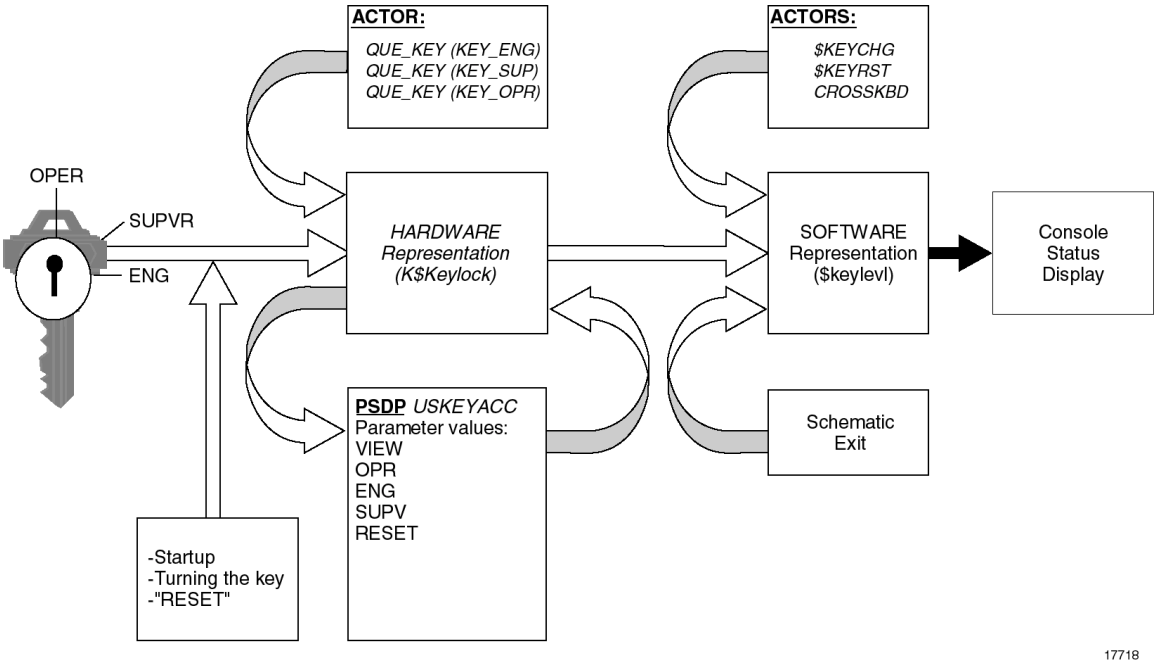
Parameter Explanation—

Level	Internal Keylevel Status
ENGR	Engineer
SUP	Supervisor
OPR	Operator
VIEW	View

NOTE

For Release 520 and later systems, journaling of keylevel changes initiated by this actor can be suppressed by a System Wide Values option in the Network Configuration File.

3.5.1 Hardware and Software Keylevels



There are two locations on Universal Stations (US, U^{x}S, and GUS) which represent keylevel: a hardware representation, and a software representation. These locations affect how the US responds to user requests to perform various functions.

"K$Keylock" is the hardware representation of keylevel. It exists whether or not there is a physical key on that station. K$Keylock is used by the station to determine how the US responds to user requests to perform station related functions (System Menu access, data changes from standard displays, etc.). When a Universal Station first comes up, the physical keylevel position value is stored to K$Keylock. If the station does not have a key switch, a value is still stored into K$Keylock. Anytime the physical hardware key is changed, K$Keylock is updated.

The actor "QUE_KEY" can also store a new value to K$Keylock. If the physical key is in the OPER position, and QUE_KEY sends a value of SUPVR to K$Keylock, then the system will think the keylevel is SUPVR. This will remain in effect until either the physical key is turned to a different position, or the QUE_KEY actor is executed which stores a new value to K$Keylock.

“$keylevl” is the software representation of keylevel. $keylevl is used within schematics to determine how the US responds to user requests to perform schematic related functions (actor execution, etc.). The actors $KEYCHG, KEYENG, and CROSSKBD are able to store a specific keylevel value to $keylevl. Additionally, the actor $KEYRST causes the value of K$Keylock to be stored to $keylevl. $KEYRST is executed from a target, or is executed automatically by the station whenever the user exits a schematic by invoking another schematic, or any other US display. Disconnecting a CROSSKBD also has the effect of restoring the value of $keylevl to the value held by K$Keylock.

The station always makes sure that any changes made to K$Keylock are automatically propagated to $keylevl. This means that when any physical key changes, QUE_KEY stores will update K$Keylock and then $keylevl.

In UIS Release 610 a new PSDP parameter (USKEYACC) will be added which will provide the ability to read, and write to the K$Keylock of any Universal Station (US, UxS, GUS) on an LCN. USKEYACC will always mirror the value of K$Keylock, at station start and throughout its operation. Whenever USKEYACC is written to by the local station or by a remote station, the modified value will be stored to K$Keylock.

The behavior of K$Keylock, $keylevl, and the actors remains the same (for example, a value change in K$Keylock is always propagated to $keylevl). The user has the ability to always restore K$Keylock to match the physical key position by storing a RESET value to USKEYACC, or by simply turning the key. If the Universal Station being restored does not have a physical key, the RESET will return the station’s keylevel values to their startup state. On a GUS without a physical key, RESET will return the station's keylevel value to the Emulated Key lock default value.

Any change to the nodes keylevel will be journalized to both the HM and the RTJ. The NCF key change journal suppression option will still suppress the CROSSKBD, $KEYCHG, $KEYRST, KEYENG, KEYRST and S_SENM actors. Changes made to the USKEYACC PSDP parameter will not be suppressed.

When schematics that use actors (KEYENG, $KEYCHG) to temporarily change the nodes keylevel access are exited, the nodes access level returns to the value existing in the PSDP parameter.

3.6 KEYLEVEL RESET

This actor sets the internal keylevel status to match the actual position of the keyswitch in the Universal Station. Refer to the $KEYCHG actor for typical uses.

The $KEYRST actor is executed even if an error occurs in previous actors (after changing the internal keylevel status). The internal keyswitch level is set to match the Universal Station keyswitch before the target completes execution.

Calling Sequence— $KEYRST

Example— See the Keylevel Change actor

Optional Parameter Entry Form–None

> **NOTE**
>
> For Release 520 and later systems, journaling of keylevel changes initiated by this actor can be suppressed by a System Wide Values option in the Network Configuration File.

3.7 KEY SIMULATION

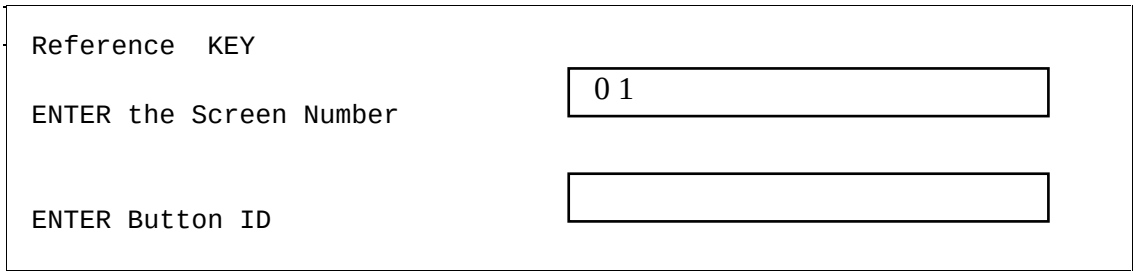
This actor is used to simulate a keystroke at a specified station. The sending and receiving stations must be located in the same console. Either the Universal or Operator Personality software can be running in the receiving station.

Calling Sequence— KEY(STATION,BUTTON ID)

Examples—KEY(02,PRR_DISP) simulates pressing the PRIOR DISPLAY key on station 2.

KEY(G_INT(INT01),SELECT) simulates pressing the SELECT key on the station number contained in INT01.

Optional Parameter Entry Form



Parameter Explanation

Screen— The station number that is to receive the simulated keystroke. The range is 1-10.

Button ID— The keystroke to simulate. Appendix C lists the Key IDs. Do not use this actor to simulate the Delete key.

CAUTION

The receiving station queue can hold up to 20 simulated keystrokes. Keystrokes received when the queue is full are discarded with no warning; therefore, you must restrict the speed of simulated keystrokes to a rate that does not cause the receiving station queue to overflow.

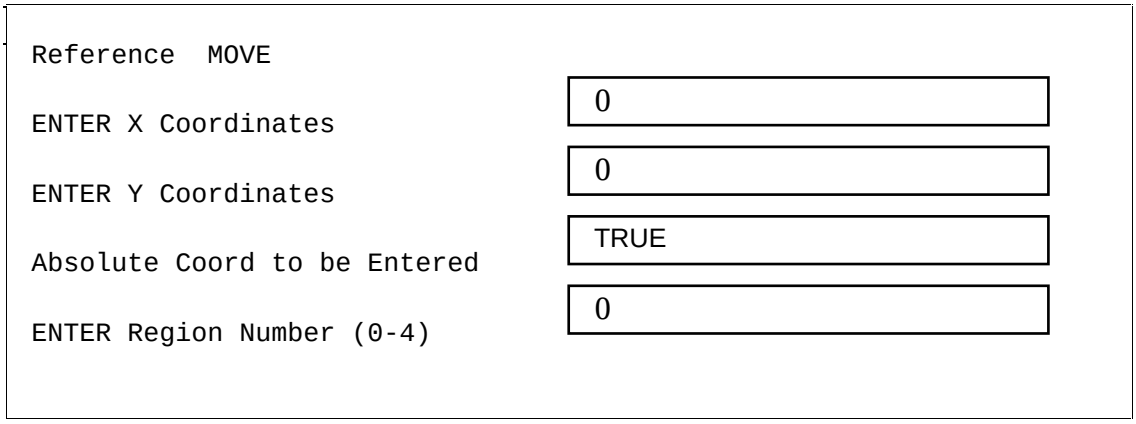
3.8 MOVE THE CURSOR

This actor allows the user to position the screen's visible cursor.

Calling Sequence— MOVE(X,Y,ABSOLUTE COORD,REGION NO.)

Example—MOVE(12,18,TRUE,2) moves the cursor to a point 12 columns from the left edge and 18 lines from the bottom of the screen.

Optional Parameter Entry Form



Parameter Explanation

- X-Y Coordinates (in character units)
  Specifies where you want the cursor to move (see Figure 1-4 in Section 1).

  X can range from 0-79
  Y can range from 0-25

- Absolute Coord
  Specifies whether the actor is to perform its function at the absolute coordinates or relative to the upper-left corner of the target.

  TRUE means move the cursor to the X,Y coordinate for the desired region.
  FALSE means move the cursor relative to the target's upper-left corner.

- Region refers to the region number in the display and can range from 0-4 (see Figure 1-4 in Section 1). A region number = 0 means use the region where the target is located (at run time).

  For a configurable button to work properly, you must specify a region number other than zero (0).

3.9 MULTIPLE REMOTE KEYSTROKES

This actor sends a string of characters to a remote station. The characters appear at the current cursor position on the remote screen.

The cursor on the remote screen must be positioned in either a text input port, or in a position that accepts character input such as on the command line of the Command Processor screen.

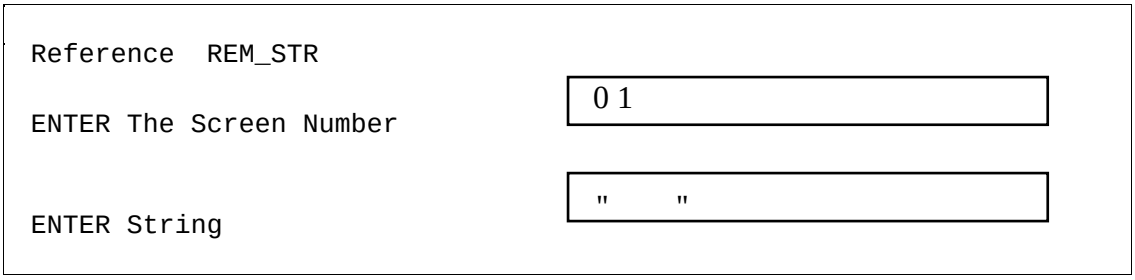
Calling Sequence— REM_STR(Screen Number,String)

Examples—

REM_STR(04,"Emergency Alarm on Unit 02")

REM_STR(R_INT(0,0,30,"Enter the screen number",TRUE,0),R_STR(0,0,80,"Enter the message to send",TRUE,0))

Optional Parameter Entry Form



Parameter Explanation

- Screen— The station number that is to receive the string of characters. The range is 1-10.
- String— 1 to 80 alphanumeric characters.

NOTE

Typically, the DELAY actor will need to be used following the REM_STRactor to allow the “remote” US to catch up prior to starting another actor that emulates keystrokes.

A delay of 1 second for every eight characters sent via REM_STR is usually sufficient.

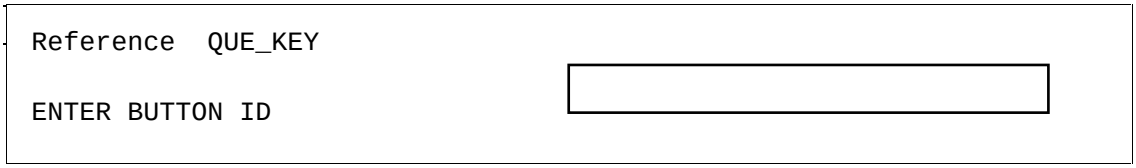
The need for a delay becomes more pronounced in R520 software.

3.10 QUEUE KEY

This actor allows a configurable button or schematic target to emulate any key on the Engineer's or Operator's keyboards. Refer to the Button Configuration publications for additional information (see References).

Calling Sequence— QUE_KEY(BUTTON ID)

Optional Parameter Entry Form



Parameter Explanation— Button ID is any key or configurable button identifier listed in Appendix C (except do not use this actor to emulate the Delete key).

CAUTION

It is possible to have a configurable button emulate itself, but doing so will put the station in a loop and should be avoided. Should this occur, it is possible to break out of the loop by pressing an invalid key (i.e., a key that is inappropriate for use in the current display).

NOTE

Target actions with multiple targets selected in sequence by QUE_KEY (TAB);QUE_KEY (SELECT) may experience occasional Select failures. The cause is related to the Universal Station's video card. Faster video card types are most likely to exhibit the problem.

To prevent Select action failures, use the following target actions:

QUE_KEY(TAB);DELAY(.....);QUE_KEY(SELECT)

The delay time should be a minimum of 100 mSeconds (The Delay Actor is described in Section 12).

A QUE_KEY (MOVE(.....)) action can be used instead of QUE_KEY (TAB) and does not require the Delay Actor, but this is more difficult to build because you must determine the coordinates for each target position.

3.11 REMOTE MOVE

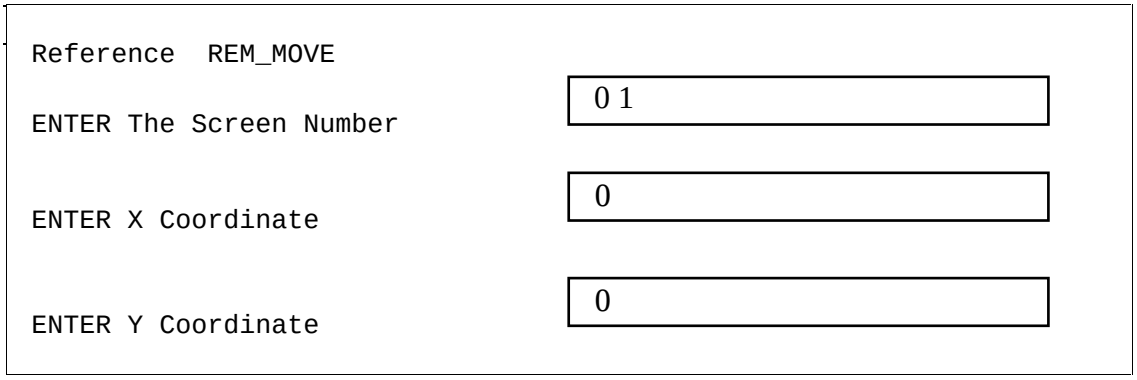
This actor moves the cursor at a remote station to a specified position.

Calling Sequence— REM_MOVE(Screen Number, X,Y)

Examples— REM_MOVE(03,0,25)

```
REM_MOVE(R_INT(0,0,30,"Enter the screen number",TRUE,0),
                (R_INT(0,0,30,"Enter the X Coordinate",TRUE,0),
                (R_INT(0,0,30,"Enter the Y Coordinate",TRUE,0));
```

Optional Parameter Entry Form



Parameter Explanation

- Screen— The station number that is to receive the string of characters. The range is 1-10.
- X Coordinate— The absolute “X” coordinate in characters. The range is 0 to 79. See Figure 1-4 in Section 1.
- Y Coordinate— The absolute “Y” coordinate in characters. The range is 0 to 25. See Figure 1-4 in Section 1.

3.12 STORE PRIMMOD

This actor assigns a Primmod value into the lamp-specific data field of a configurable button on the Universal Station where this actor executes. The configurable button must be one of the 40 keyboard buttons with LED Alarm indicators. For this discussion, an $MPROD string is equivalent to a Primmod name.

When a Store Primmod actor successfully executes, the event is sent to the Real Time Journal and recorded in Event History.

This actor is valid for Release 520 and later systems.

Calling Sequence— S_PRIM("button", "Primmod")

Examples—

S_PRIM("AN_CNF9","PM_GRP_AM101")
S_PRIM("AN_CNF9", " ")

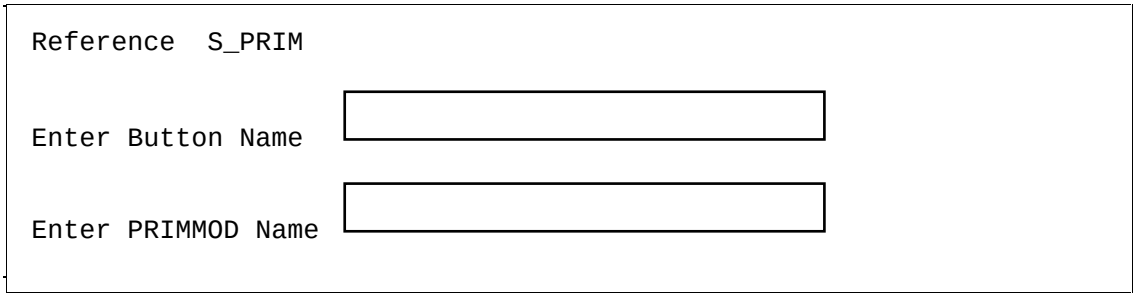
In the first example, the Primmod PM_GRP_AM101 is assigned to the lamp-specific data field for the configurable button named AN_CNF9.

In the second example, the actor clears any lamp-specific data assignment for the configurable button named AN_CNF9. The assignment is cleared if a Primmod, a Unit, or an Annunciator Group was assigned.

NOTE

If multiple actors attempt to update a configurable button at the same time, the last actor to execute determines the primmod value stored.

Optional Parameter Entry Form



Parameter Explanation—

Button is the symbolic name of a configurable button on the US keyboard. It consists of a 1- 8 character string. Standard Default Symbolic Button names are listed in Appendix C. Only configurable buttons with alarm indicators can be specified. Typically these are AN_CNF7 through AN_CNF46, (but see the note on the next page).

If an invalid symbolic button name is specified, the actor will fail to execute and the message INVALID BUTTON NAME is displayed.

NOTE

The Default Symbolic Button names may have been changed by a user-created Button Name File.

Primmod is a string of 1 - 16 characters, null characters, or spaces.

If an invalid primmod name is specified during configuration, the primmod name appears in red and an error message is displayed. If an invalid primmod name is specified at operating time, the actor will fail to execute and the error message INVALID PRIMMOD NAME is displayed.

4 OPERATOR INPUT ACTORS

This section describes actors that allow input from the operator. These actors output a prompt message and allow the operator to enter a value. The input value is converted to an internal form for use by other actors. Therefore, these actors cannot stand alone; they must be used to replace a parameter within some other actor.

Parameter Explanation— All of the actors allow you to specify the following:

- X, Y— The X and Y coordinates (in character units) of the Text Input Port (TIP) that is to be used to enter the data. These coordinates specify the left-most character of the Tip. The Tip must be only one character high. Be careful when creating or moving a target to choose coordinates for the Tip that are within the specified limits of the screen as illustrated in Figure 1-4 in Section 1 (X = 0 through 79; Y = 0 through 25).
- TIP LENGTH— The length of the port entry in character units. The range of this parameter is from 1 through 80 characters.
- PROMPT— A prompt message to request the value. The Length of the prompt string ranges from 1 through 30, and the string must be enclosed in quotes. The prompt is always written to region 1, at coordinate X=0, Y=1.
- ABSOLUTE COORD— Specifies the location of the Text Input Port in absolute coordinates, or relative to the target.

  FALSE means to display the TIP at the X-Y coordinate relative to the target's upper-left-hand corner. In this case, the X coordinates range from +79 to -79 and the Y coordinate ranges from +25 to -25.

  TRUE means to display the TIP at the X-Y coordinate relative to the origin of the region (see below).
- REGION NUMBER— The region that will contain the Text Input Port (see Figure 1-4 in Section 1). A region number equal to zero (0) means to use the region number where the target is located.
- CONFIGURABLE BUTTONS— The TIP location for an Operator Input actor that is activated by a configurable button is fixed at the top line of screen-region 1 (see Figure 1-4 in Section 1). **Only** the TIP LENGTH and PROMPT parameters are accepted.

Do not put a Text Input Port on the lower line of region 1.

NOTE

If a Text Input Port (TIP) is opened on top of a value that exists on the screen, the value that the TIP is on top of will overwrite an operator's entry and be used as the entered value. Always position text imports (ports) so they don't overlap any value objects.

NOTE

At operating time, if an Operator Input Actor is currently active (i.e., the text input port is open and awaiting input) and another target is chosen, the first actor is preempted and the text input port (TIP) for the second actor will appear; however, when a variant calls up a subpicture, the situation is different. If an Operator Input Actor in that subpicture is awaiting input, and the variant switches to another limb, the TIP is not closed (i.e., until you enter something). If nothing is entered into the port within two minutes, the remaining Actor sequence is aborted and the port is closed.

4.1 READ BOOLEAN DATA

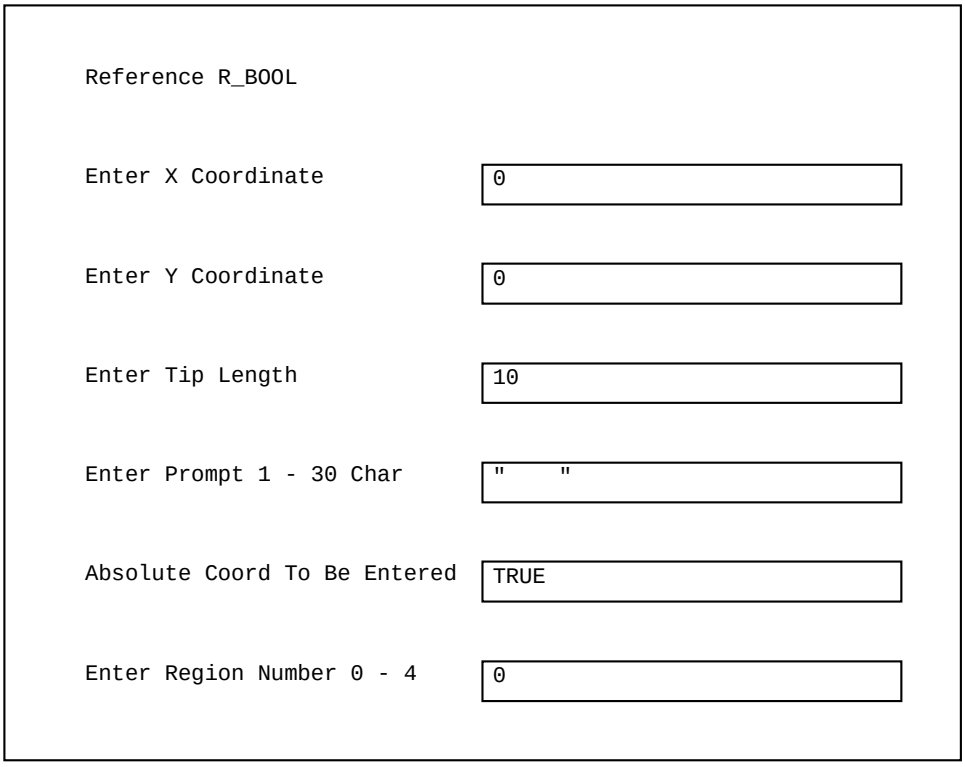
This actor allows Boolean data to be entered through a Text Input Port (TIP).

Calling Sequence— R_BOOL(X,Y,TIP LENGTH,"PROMPT",ABSOLUTE COORD,REGION NO)

Example—S_BOOL(BOOL01,**R_BOOL**(10,20,10,"ENTER BOOLEAN VALUE", TRUE,2))

A Boolean value is returned.

Optional Parameter Entry Form



4.2 READ DATE

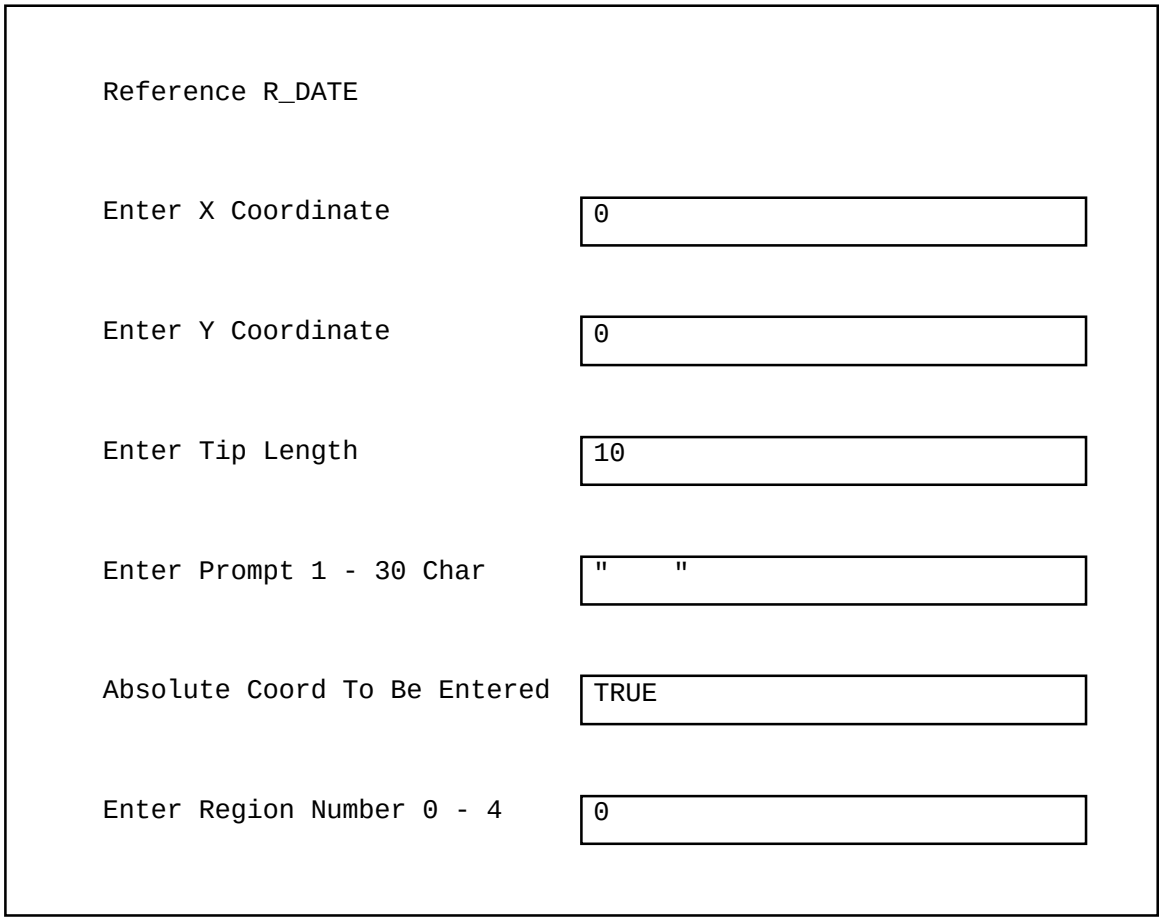
This actor allows a Date to be entered through a Text Input Port (TIP). Valid formats for the date are Numeric and European. Examples of these two formats to represent March 15, 1985 are 3 15 85 and 15 MAR 85.

Calling Sequence— R_DATE(X,Y, LENGTH,"PROMPT",ABSOLUTE COORD,REGION NO)

A date value is returned.

Example—S_DATE(DATIME1,**R_DATE**(10,20,10,"ENTER DATE VALUE", TRUE,2))

Optional Parameter Entry Form



4.3 READ DURATION

This actor reads duration data entered through a Text Input Port (TIP).
Valid formats are—

DDDDD HH MM SS or HH MM SS or HH MM

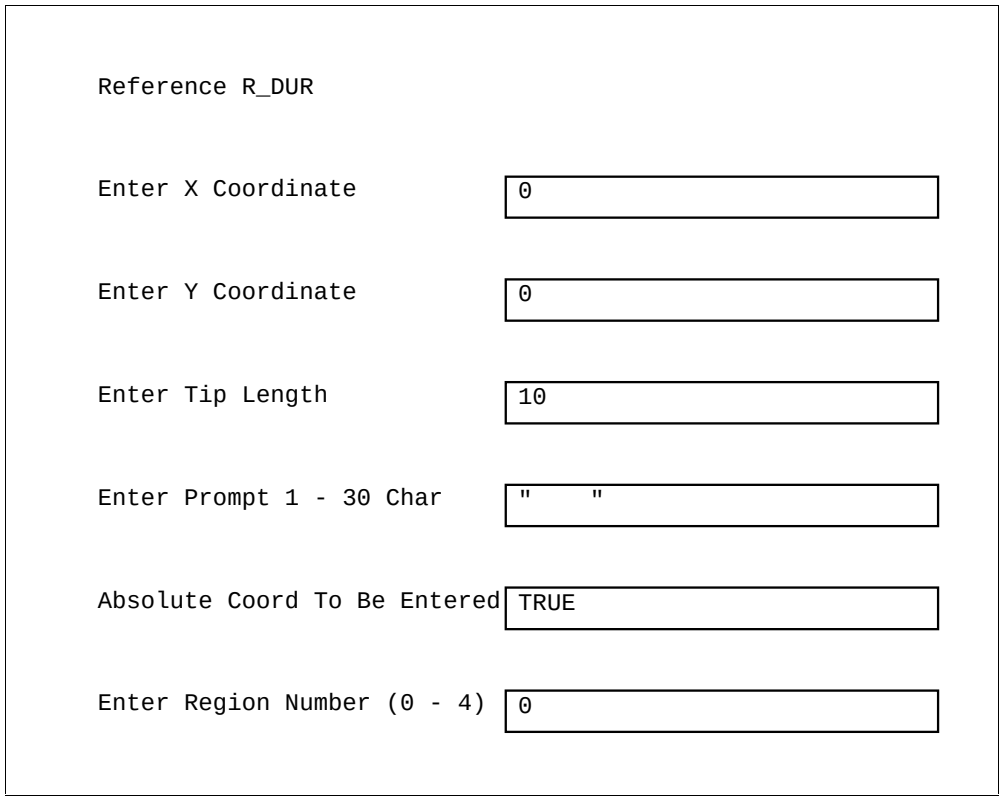
where DDDD = the number of days in the duration
HH = the number of hours in the duration
MM = the number of minutes in the duration
SS = the number of seconds in the duration

Calling Sequence— R_DUR(X,Y, LENGTH,"PROMPT",ABSOLUTE COORD, REGION NO)

A time value is returned.

Example—S_DUR(DATIME1,**R_DUR**(10,20,10,"ENTER DURATION VALUE", TRUE,2))

Optional Parameter Entry Form



4.4 READ TIME

This actor allows the time to be entered through a Text Input Port (TIP). The format is HH MM SS. HH = hours, MM = minutes, SS = seconds.

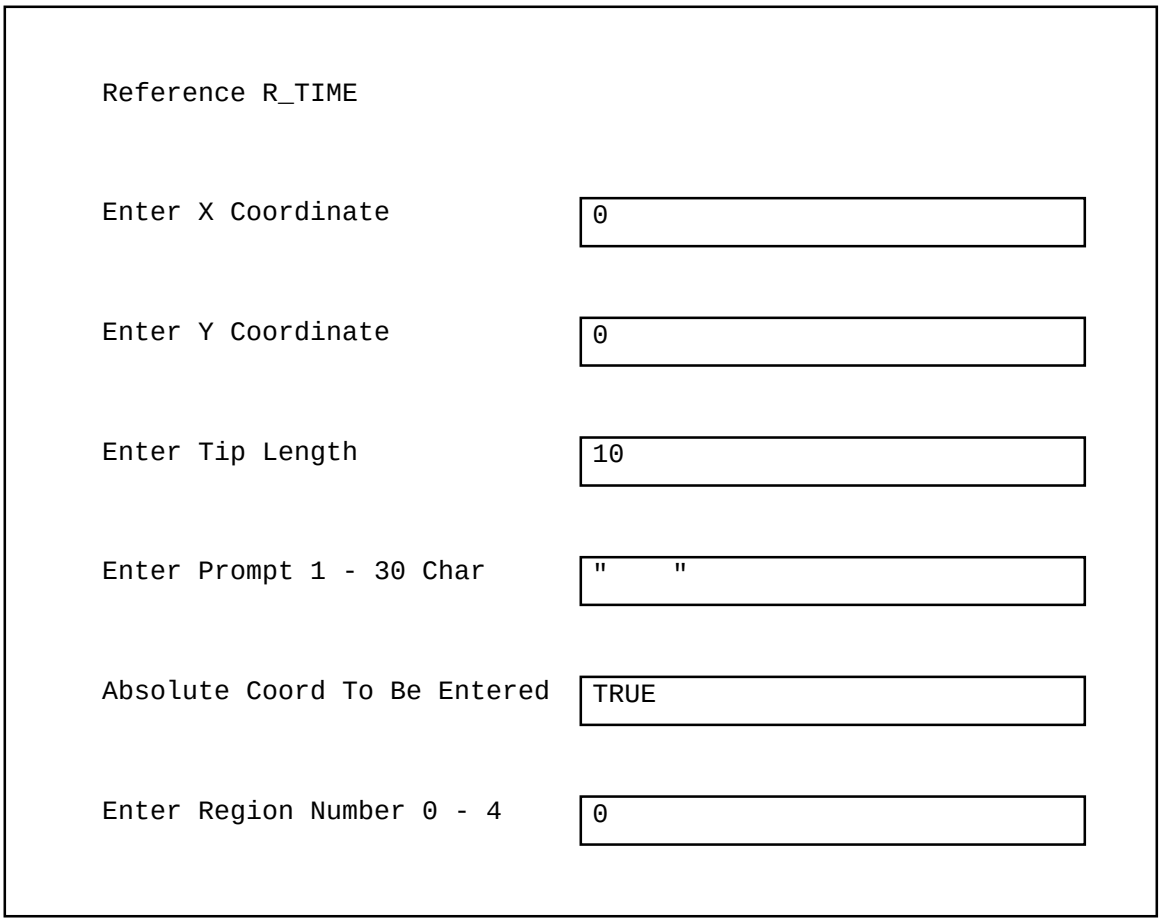
Note that when time is entered through the Picture Editor, the format is type checked. Only a time format is legal. If the time entry is nested with a date/time actor, the format allowed can be Date or Time.

Calling Sequence— R_TIME(X,Y,LENGTH,"PROMPT",ABSOLUTE COORD,REGION NO)

A time value is returned.

Example— S_TIME(DATIME1,**R_TIME**(10,20,10,"ENTER TIME VALUE",TRUE,2))

Optional Parameter Entry Form



4.5 READ ENTITY

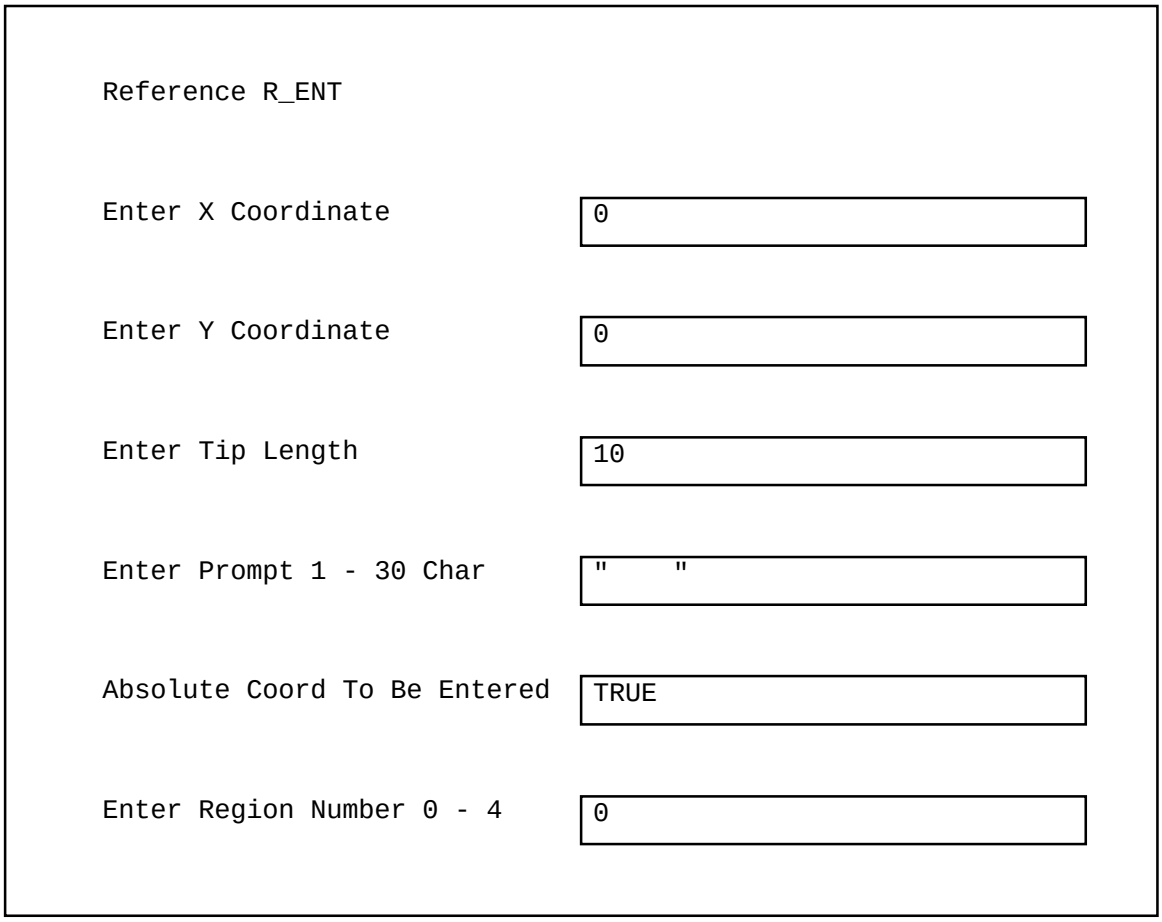
This actor allows a point name to be entered through a Text Input Port (TIP). If the point name doesn't exist, an error message is returned.

Calling Sequence— R_ENT(X,Y,LENGTH,"PROMPT",ABSOLUTE COORD,REGION NO)

An entity ID is returned.

Example— S_ENT(ENT01,**R_ENT**(10,20,10,"ENTER VALUE",TRUE,2))

Optional Parameter Entry Form



4.6 READ ENTITY NAME

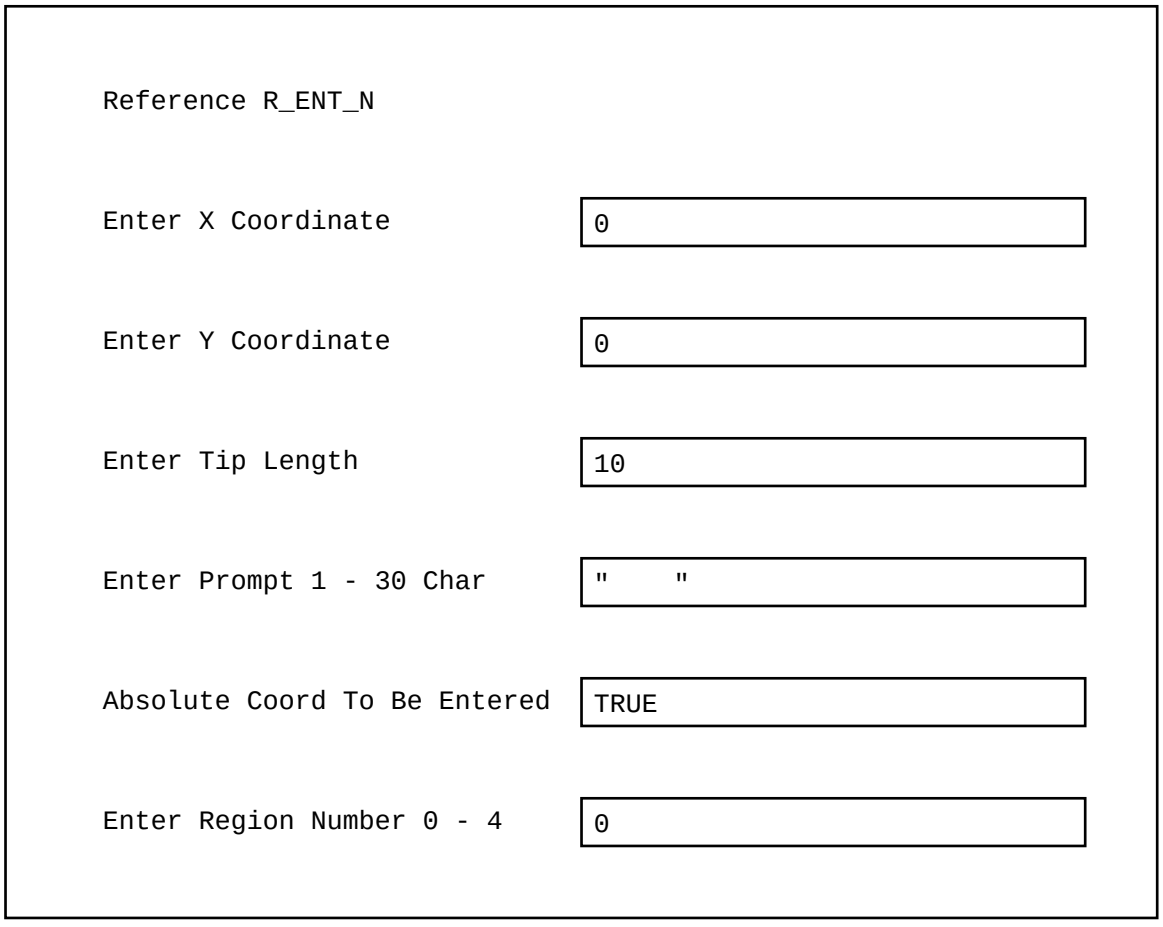
This actor allows string data that is considered to be a point name to be entered through a Text Input Port (TIP). Existence of the point name is verified through the network.

Calling Sequence— R_ENT_N(X,Y,LENGTH,PROMPT,ABSOLUTE COORD, REGION NO)

A string is returned that is equivalent to the point name.

Example—S_STR(STRING01,**R_ENT_N**(35,1,20,"ENTER ENTITY VALUE",TRUE,1))

Optional Parameter Entry Form



4.7 READ ENUMERATION

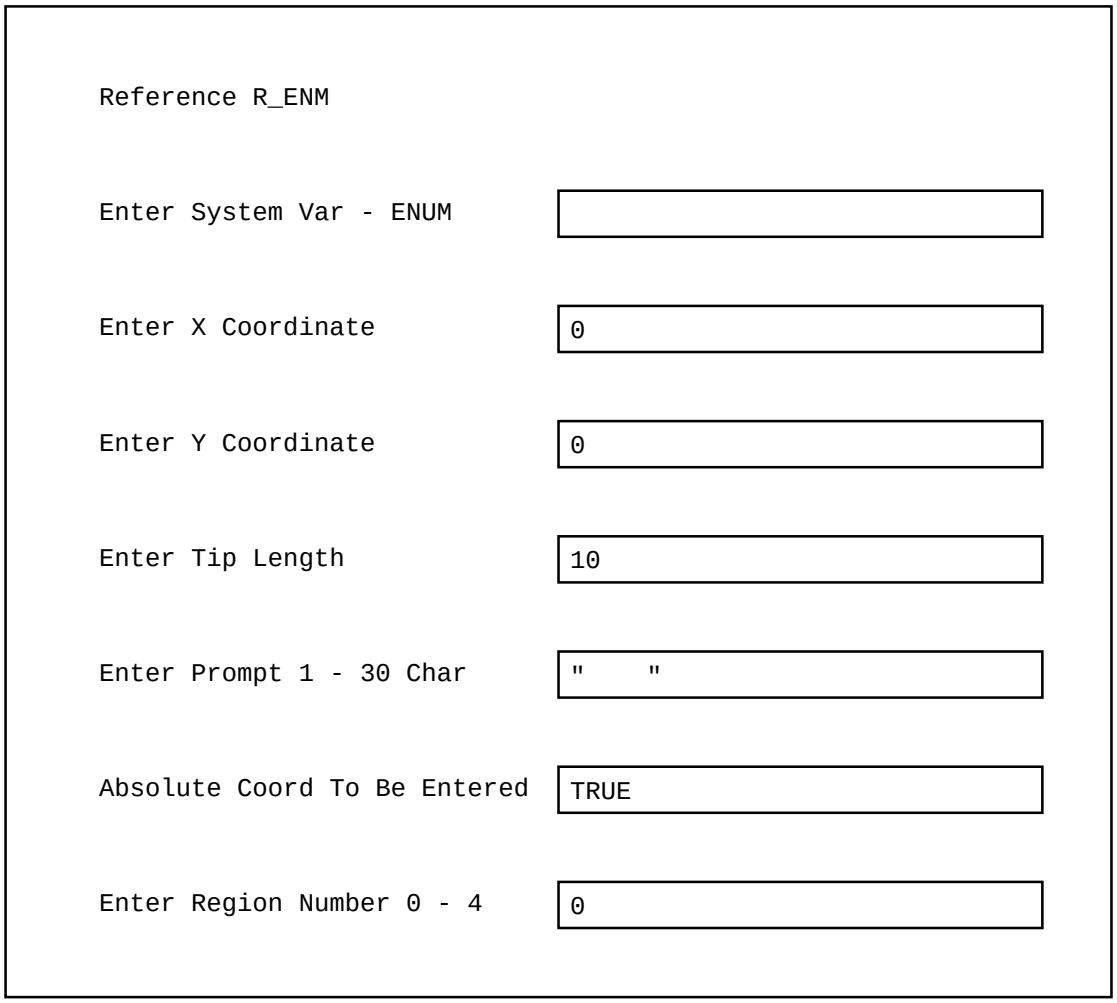
This actor allows enumerations for Entity types to be entered through a Text Input Port (TIP). The Variable ID is required for data-type verification at run time. Refer to the Parameter Reference Dictionaries for more information on enumerations.

Calling Sequence— R_ENM(VARIABLE ID,X,Y,TIP LENGTH, "PROMPT", ABSOLUTE COORD,REGION NO)

A string is returned; the string is a member of the enumerations set.

Example—S_STR(STRING01,**R_ENM**(A100.MODE,0,1,30,"ENTER: AUTO or CAS or MAN",FALSE,1))

Optional Parameter Entry Form



4.8 READ INTEGER DATA

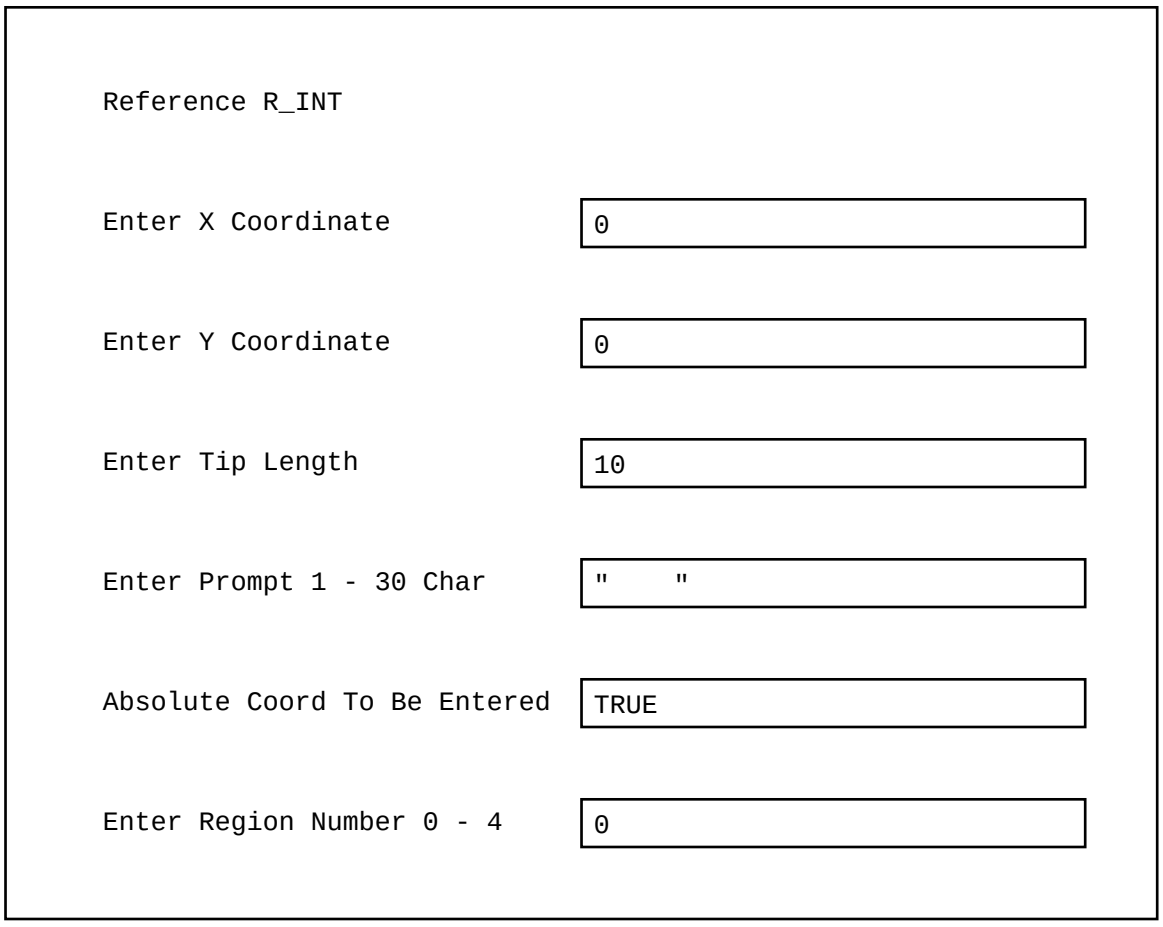
This actor allows Integer data to be entered through a Text Input Port (TIP).

Calling Sequence—R_INT(X,Y,TIP LENGTH,PROMPT,ABSOLUTE COORD, REGION NO.)

An integer value is returned.

Example— S_INT(INT01,**R_INT**(10,20,30,"ENTER VALUE",TRUE,2))

Optional Parameter Entry Form



4.9 READ PARAMETER ID

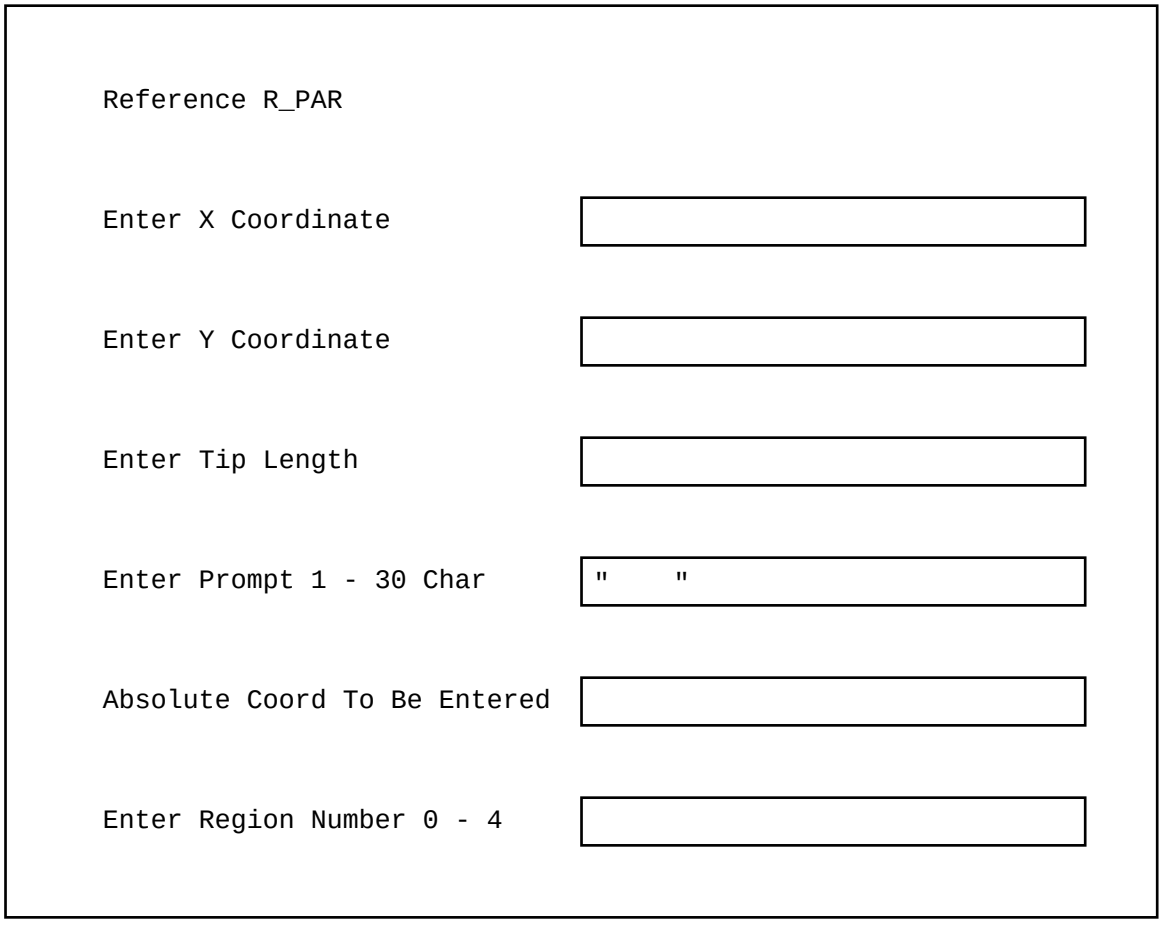
This actor is used to read any Parameter ID. A Parameter ID is defined as a legal, nonsubscripted parameter name. This actor must be used with another actor that requires a parameter such as C_VAR.

Calling Sequence— R_PAR(X,Y,LENGTH,"PROMPT",ABSOLUTE COORD, REGION NO)

A Parameter ID is returned.

Example— S_VAR(VAR01G,C_VAR(G_ENT10G),0,**R_PAR**(0,1,10,"ENTER PARAMETER",FALSE,0),-9999));S_STR(STR10G,IE_VAR(VAR10G))

Optional Parameter Entry Form



4.10 READ REAL DATA

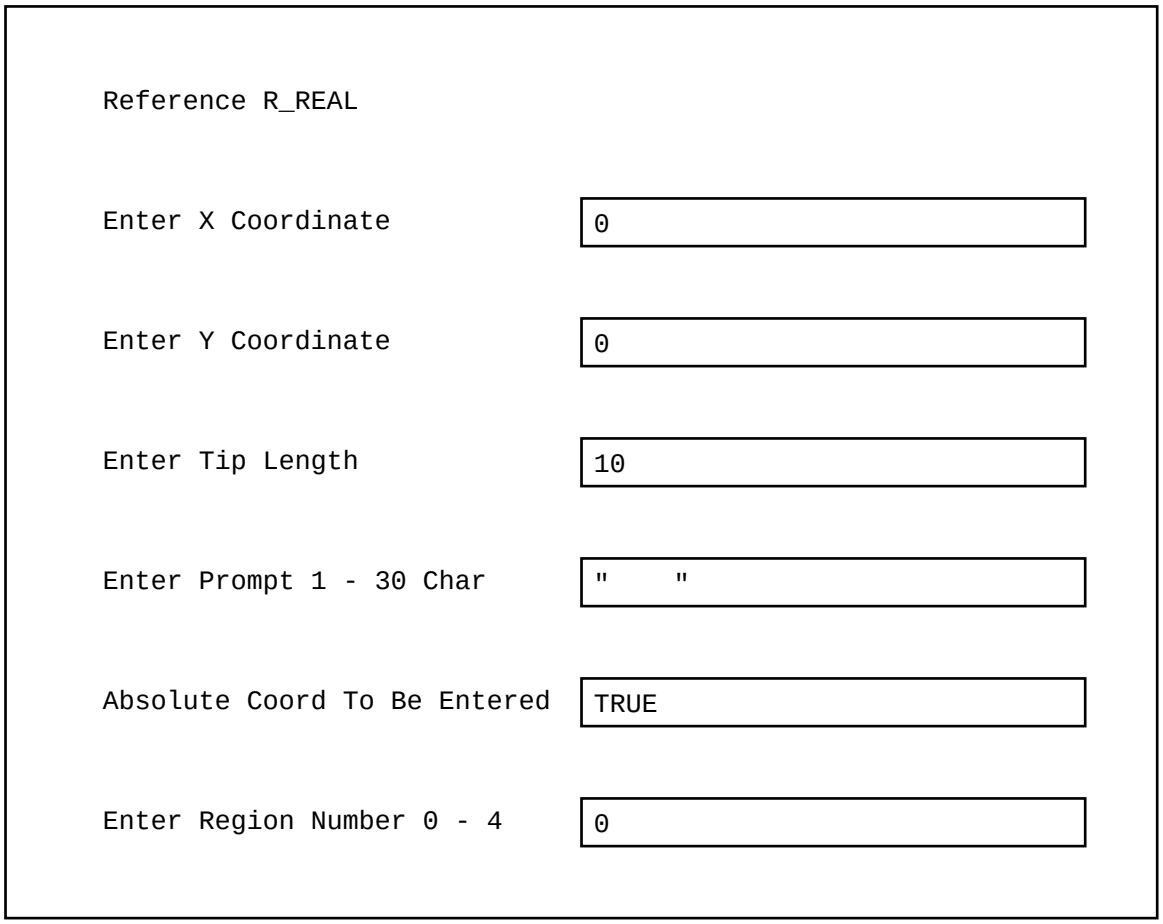
This actor allows Real numbers to be entered through a Text Input Port (TIP).

Calling Sequence— R_REAL(X,Y,TIP LENGTH,"PROMPT",ABSOLUTE COORD, REGION NO)

A Real value is returned.

Example— S_REAL(REAL01,**R_REAL**(35,1,20,"ENTER REAL VALUE",TRUE,1)

Optional Parameter Entry Form



4.11 READ STANDARD ENUMERATION DATA

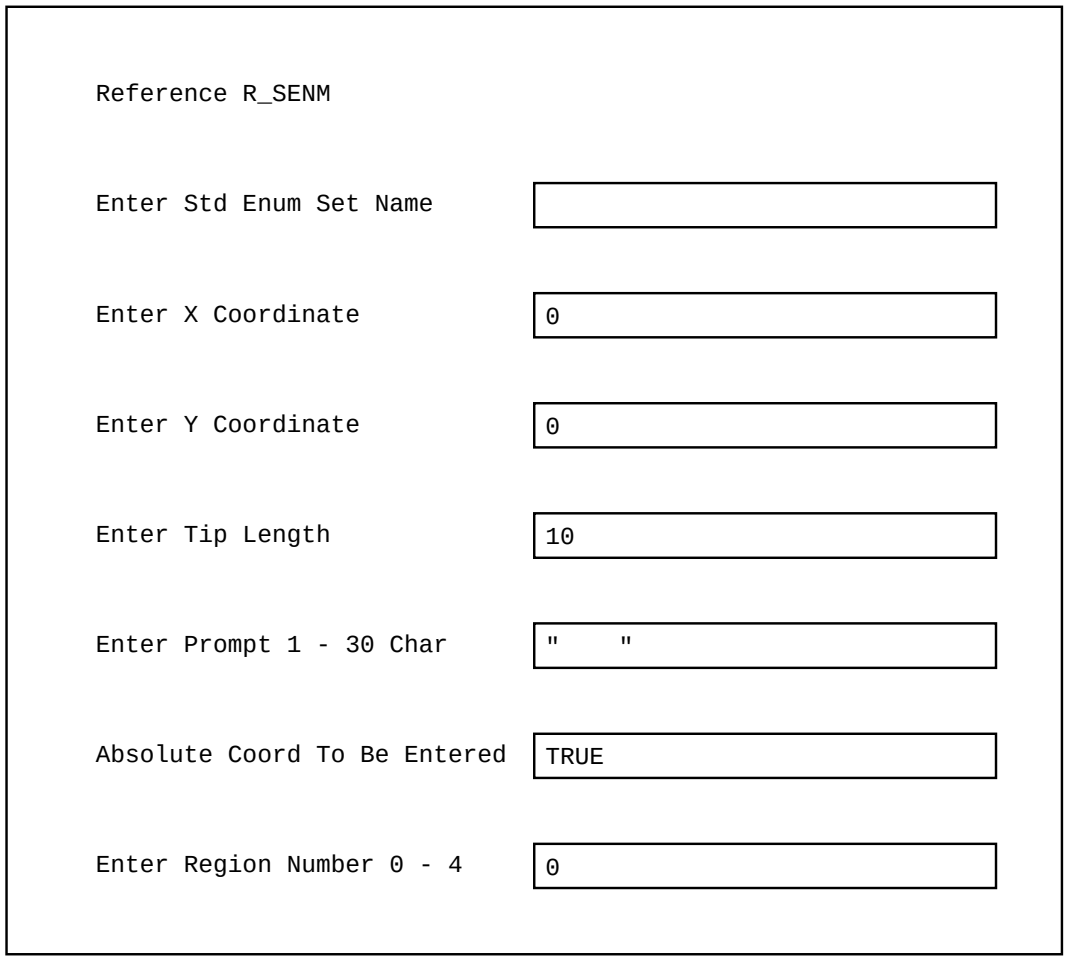
This actor allows enumeration data to be entered through a Text Input Port (TIP). It operates on only standard enumerations. The Text Input Port data-entry format is MEMBER. The Set Name is an identifier that names the particular set, such as UNIT. Refer to the Parameter Reference Dictionaries for more information on enumerations.

Calling Sequence—R_SENM("SET NAME",X,Y,TIP
LENGTH,"PROMPT",ABSOLUTE COORD,REGION NO)

An Enumeration is returned.

Example—S_SENM(ENM01,**R_SENM**("MODE",0,1,20,"ENTER:
AUTO or CAS or MAN",FALSE,1))

Optional Parameter Entry Form



4.12 READ STANDARD ENUMERATION NAME

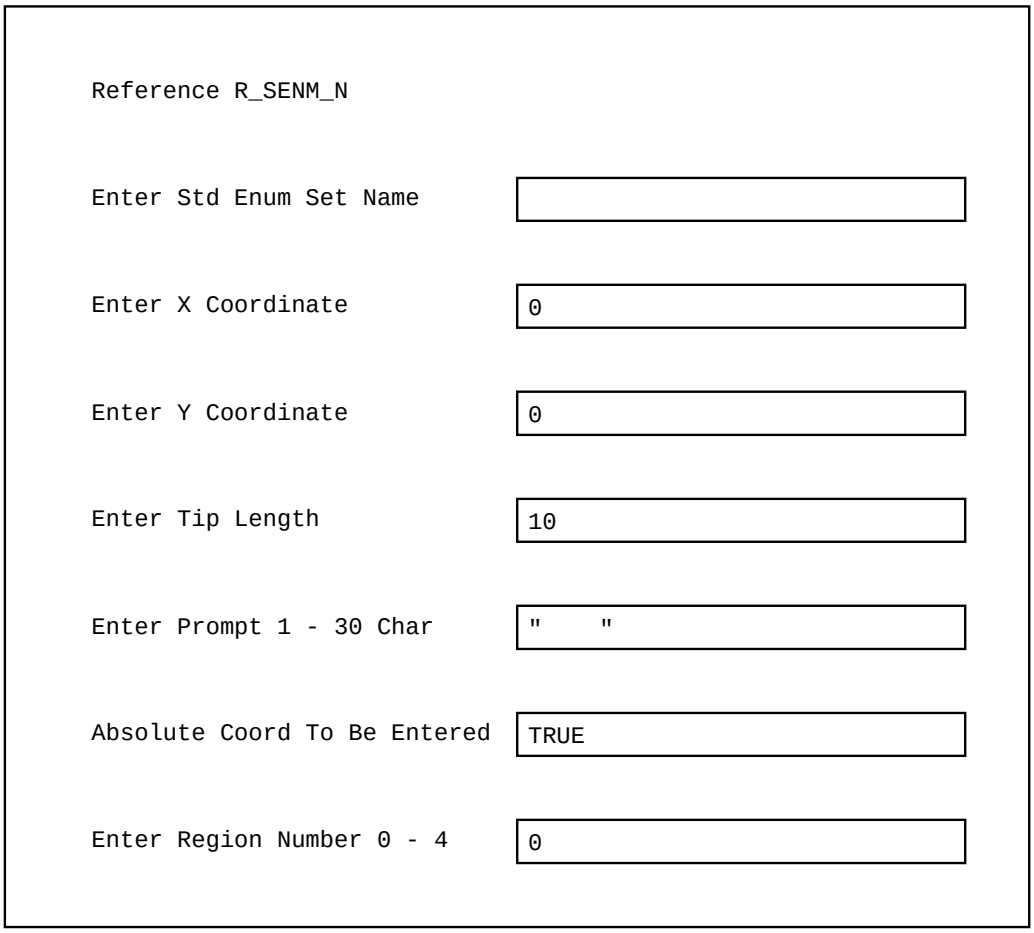
This actor allows enumeration members to be entered through a Text Input Port (TIP), it operates on only standard enumerations. The Text Input Port data-entry format is MEMBER. The Set Name is an identifier that names the particular set, i.e., UNIT. Refer to the Parameter Reference Dictionaries for more information on enumerations.

Calling Sequence— R_SENM_N("SET NAME",X,Y,TIP LENGTH,"PROMPT", ABSOLUTE COORD,REGION NO)

A string is returned; the string is a member of the enumerations set.

Example—S_STR(STRING01,**R_SENM_N**("MODE",0,1,20,"ENTER: AUTO or CAS or MAN",FALSE,1))

Optional Parameter Entry Form



4.13 READ STRING DATA

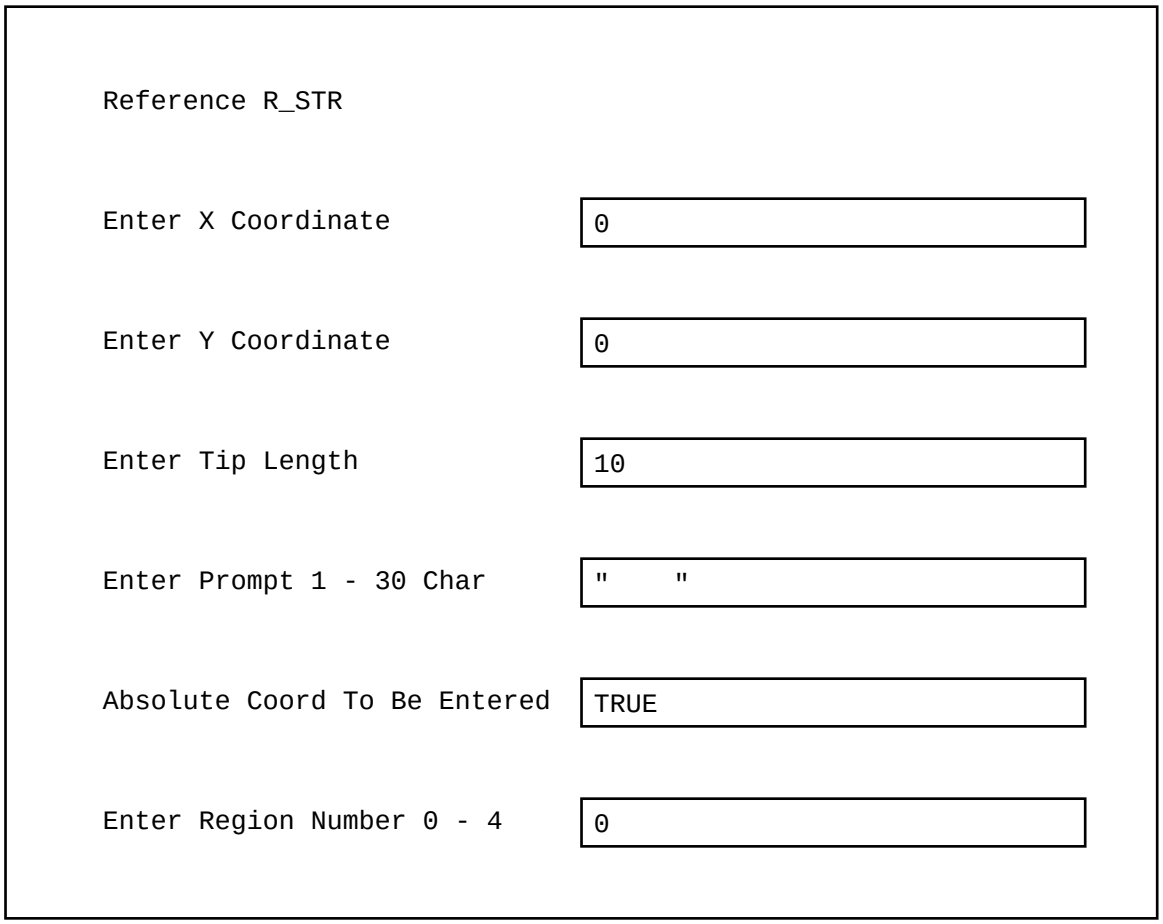
This actor allows from 1 through 80 characters of String data to be entered through a Text Input Port (TIP).

Calling Sequence— R_STR(X,Y,TIP LENGTH,"PROMPT",ABSOLUTE COORD, REGION NO)

A String value is returned.

Example— S_STR(STRING01,**R_STR**(35,1,20,"ENTER STRING VALUE", TRUE,1))

Optional Parameter Entry Form



4.14 READ VARIABLE

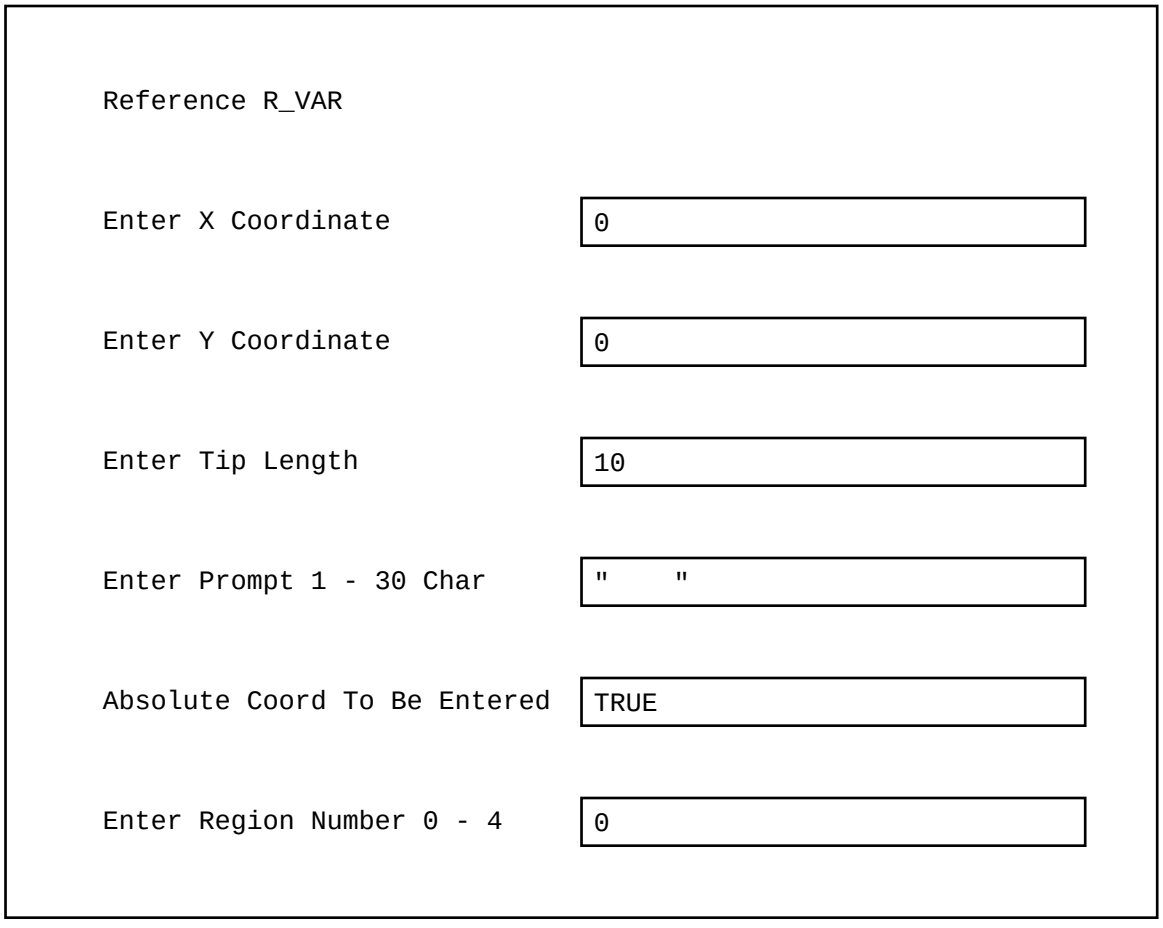
This actor allows variable IDs (e.g., an entity.parameter) to be entered through a Text Input Port (TIP). Examples: A100.PV, B100(1).TSTS(1).

Calling Sequence— R_VAR(X,Y,LENGTH,"PROMPT",ABSOLUTE COORD, REGION NO)

A variable ID is returned.

Example— S_VAR(VAR01,**R_VAR**(1,24,30,"ENTER TIME VALUE", TRUE,1))

Optional Parameter Entry Form



5 STORE INTO THE DISPLAY DATABASE

This section describes actors that allow values to be stored into the Display Data Base (DDB). All the actors have a common form and each takes two parameters:

- *the first is the name of the Display Database Variable.*
- *the second is the value to be stored into that variable.*

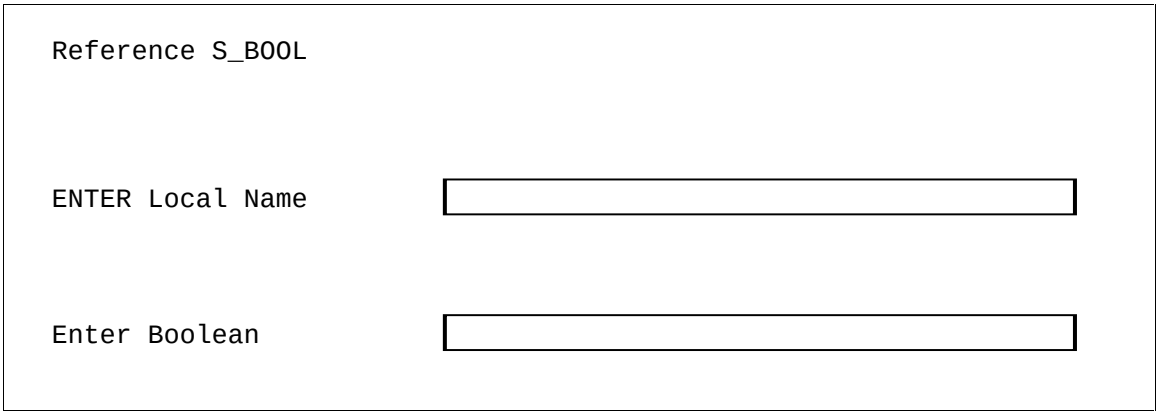
5.1 STORE BOOLEAN DATA INTO THE DISPLAY DATABASE

This actor allows Boolean data to be stored into the Display Database.

Calling Sequence— S_BOOL(NAME,BOOLEAN)

Examples—S_BOOL(BOOL01,TRUE)
S_BOOL(BOOL01G,ON)

Optional Parameter Entry Form



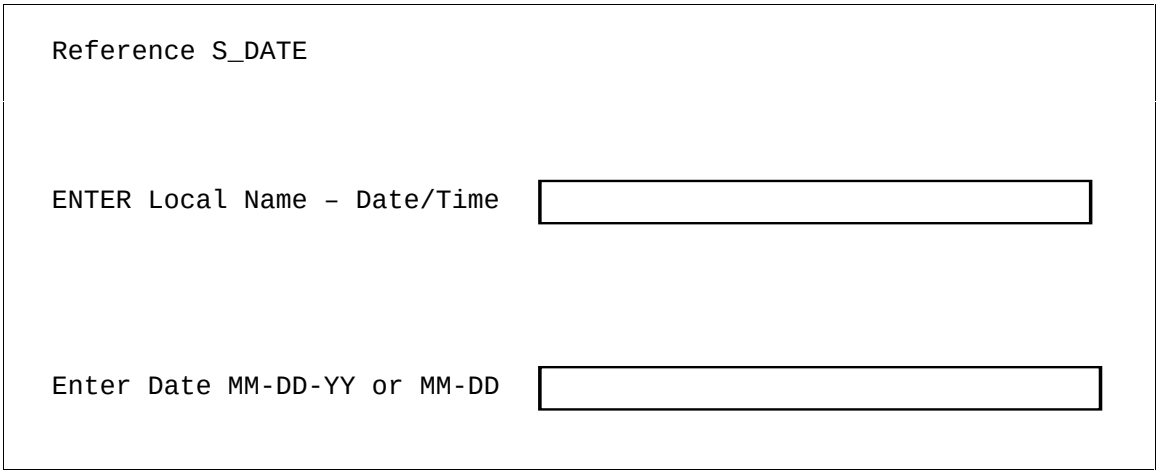
5.2 STORE DATE INTO THE DISPLAY DATABASE

This actor allows a Date to be stored into the Display Database.

Calling Sequence— S_DATE(NAME,DATE)

Examples—S_DATE(DATIME1,12-29-85)
S_DATE(DATIME1G,12-15-85)

Optional Parameter Entry Form



Note that when the date is entered through the Picture Editor, the format is type checked. Only a date format is legal. If the date entry is nested with another actor that returns a Date Time record, the format allowed can be Date or Time.

Parameter Explanation— Local Name is one of the Local Display Data Base Variables for Date/Time; for example, DATIME1. A typical date entry is MAR-15-85 or 03-15.

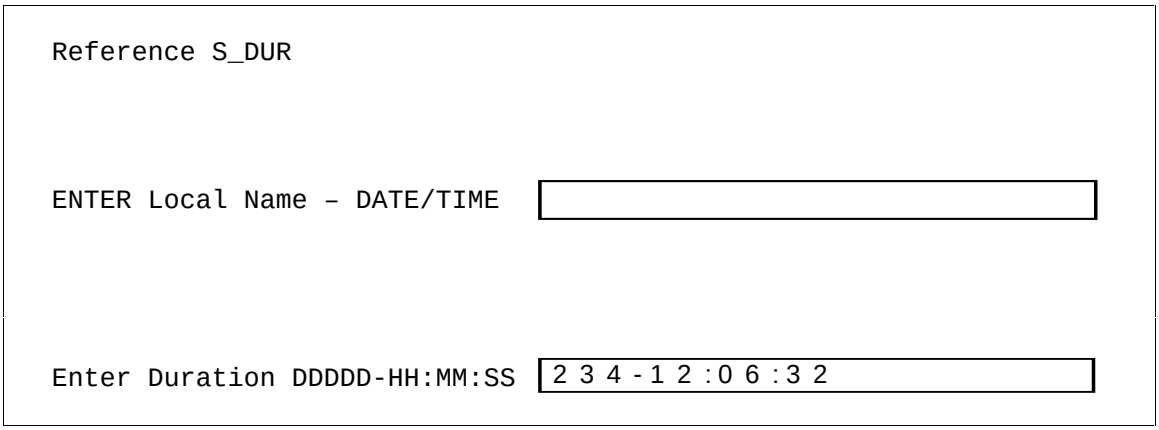
5.3 STORE DURATION INTO THE DISPLAY DATABASE

This actor stores duration data into the Display Database.

Calling Sequence— S_DUR(Name,Duration)
Where Name is a Local DDB Date/Time variable and Duration is duration data in the format DDDDD-HH:MM:SS (see below).

Examples— S_DUR(Datime1,133-05:00:59)
S_DUR(Datime2,23:30:00)

Optional Parameter Entry Form



Note that when the duration is entered through the Picture Editor, the format is type checked. Only a legal duration format is allowed. If the duration entry is nested with another actor that returns a Date/Time record, the format allowed can be Date or Time.

Parameter Explanation— Local Name is one of the Local Display Database Variables for Date/Time, e.g., DATIME1.

Valid formats for the Duration entry are—

DDDDD-HH:MM:SS or HH:MM:SS or HH:MM

where DDDD = the number of days in the duration
HH = the number of hours in the duration
MM = the number of minutes in the duration
SS = the number of seconds in the duration

Typical duration entries are 231-07:40:50, 01:00:50, or 12:59

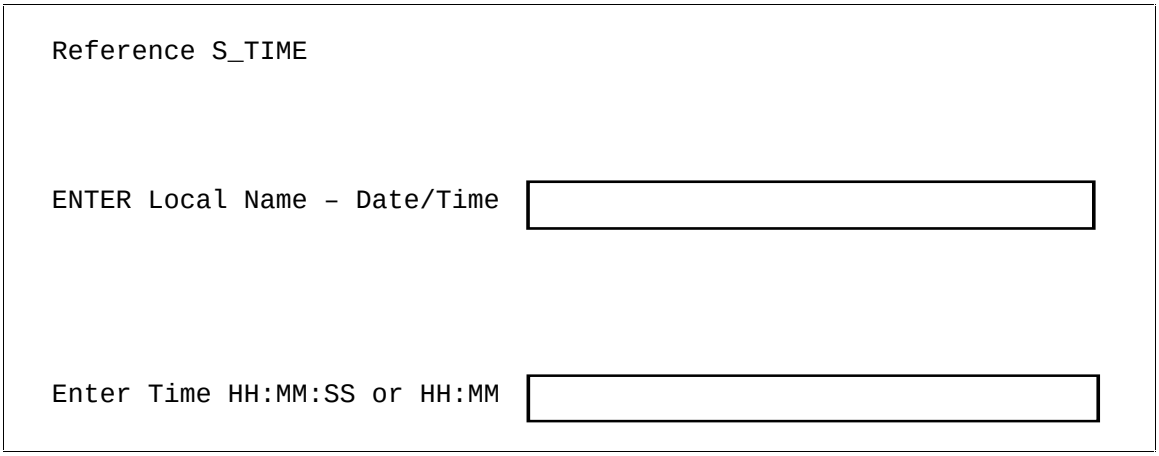
5.4 STORE TIME INTO THE DISPLAY DATABASE

This actor allows Time data to be stored into the Display Database.

Calling Sequence— S_TIME(NAME,TIME)

Examples— S_TIME(DATIME2,07:45)

Optional Parameter Entry Form



Parameter Explanation—Local Name is one of the Local Display Database Variables for Date/Time, e.g., DATIME1. A typical time entry is 07:45:20 or 07:45.

Note that when time data is entered through the Picture Editor, the format is type checked. Only a time format is legal. If the time entry is nested with a date/time actor, the format allowed can be Date or Time.

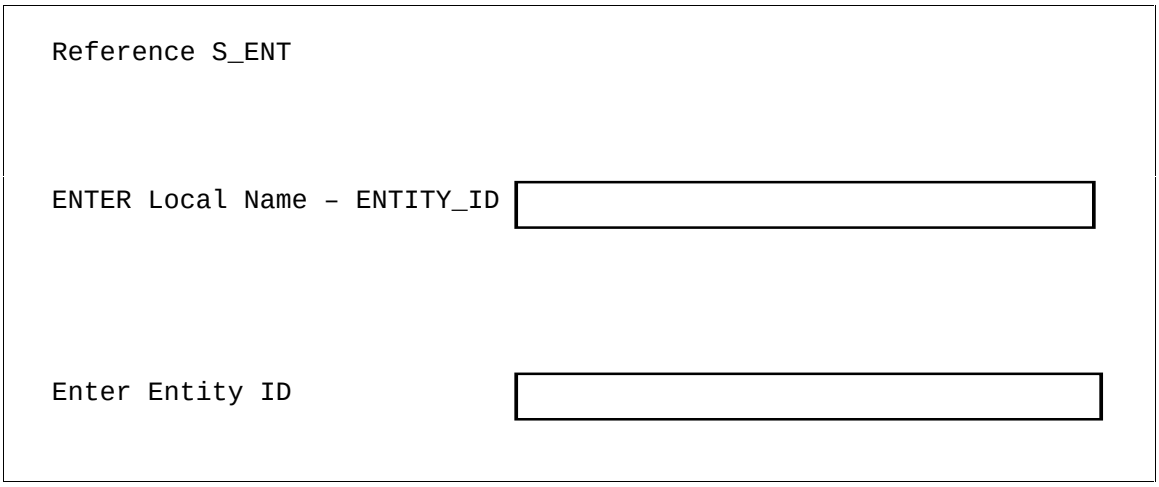
5.5 STORE ENTITY ID INTO THE DISPLAY DATABASE

This actor allows an Entity ID to be stored into the Display Database.

Calling Sequence— S_ENT(NAME,POINT NAME)

Examples—S_ENT(ENT01,HG100)
S_ENT(ENT05G,A100)
S_ENT(ENT01,&A)

Optional Parameter Entry Form



Parameter Explanation—Local Name is one of the Local Display Database Variables for an Entity; for example, ENT01. Entity ID is the point name that is to be stored into the Display Database.

5.6 EQUIPMENT LIST INITIALIZE

This actor is used to initialize or change the local Display Database (DDB) variables to those that are specified in an Equipment List Object Alias table.

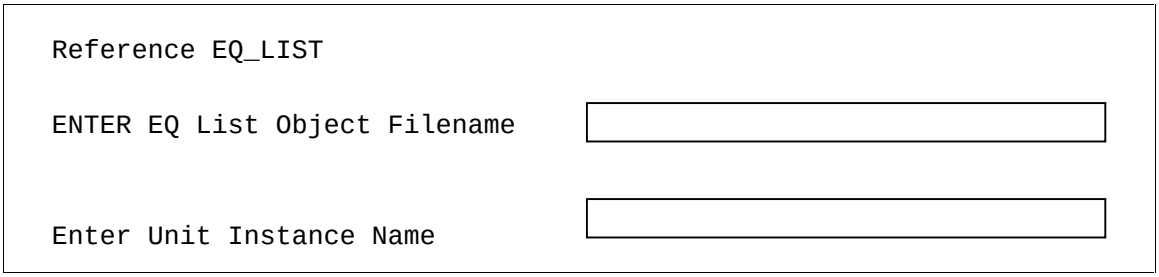
Calling Sequence— EQ_LIST("EQ List Object Filename","Instance Name")

Examples—EQ_LIST("REACTOR","REACTOR1")

EQ_LIST("REACTOR","G_STR(STR01G"))
This form of the actor could be put in an Initial Target (see Define Command) to specify the Instance Name when the display is called up.

While this actor is executing, the message: Processing EQ_LIST Actor is displayed at the top of the screen.

Optional Parameter Entry Form



Parameter Explanation

- **EQ List Object Filename**—the Equipment List Object File Name that is loaded into the Picture Editor (see LOADEQ command).
- **Unit Instance Name**—one of the Unit Instances defined in the Equipment List. This specifies the data set used to initialize or change the alias-named DDB variables.

Note that Pathname Catalog in the Area Database should specify the directory where the EQ List Object File is located. The search for EQ List Object Files is done in reverse order; therefore, for maximum performance, you should specify the schematic path first and the EQ List Object File path last.

The table below illustrates an Alias Name/Instance table in an Equipment Object List File.

ALIAS NAME	INSTANCE NAME		
	REACTOR1	REACTOR 2	REACTOR 3
Temp	TIC001	TIC002	TIC003
Heatcont	HIC001	HIC002	HIC003
Valve	VALVE01	VALVE02	VALVE03

Data is specified by an Alias Name and the Unit Instance Name. For example, if the Alias Name is TEMP and the Instance Name is REACTOR2, then the variable substituted from this table is TIC002.

Initial Action Notes—

- if you put the EQ_List actor in an Initial Action, the alias Named DDB variables in the schematic are initialized to the values in the table for the specified Instance column when the schematic is called up.

- you should put this actor in the Initial Action if the Equipment List Object File is included in the schematic. Otherwise, the alias DDB variables are displayed as having a bad value.

This actor can also be used in a regular target. The advantage is that you can change or re-initialize the alias assignments without recalling the schematic.

Error Messages —If any errors are detected (at run time), one or more of the following messages could result:

ERROR MESSAGE	EXPLANATION
SEQ Obj not used in Abst	The specified Equipment List in this actor was not loaded into the abstract (picture) at compile time.
Abstract VER mismatch	The abstract generated version does not match the equipment list support version.
EQ OBj file not found	The specified Equipment List Object File was not found on the media.
Unit Instance does not exist	The specified Unit Instance was not found in the Equipment List Object File.
EQ Obj TimeStmp mismatch different time stamp from Equipment Object	The specified Equipment List Object File exists, but ithas a File time stamp that was loaded at compile time.
Prior EQ_LIST executing	Prior EQ_List actor is still being processed.
Cannot Load EQ Obj	The specified Equipment List cannot be loaded because of a file access error.
EQ_LIST time out	A time out occurred while processing the EQ_List actor.

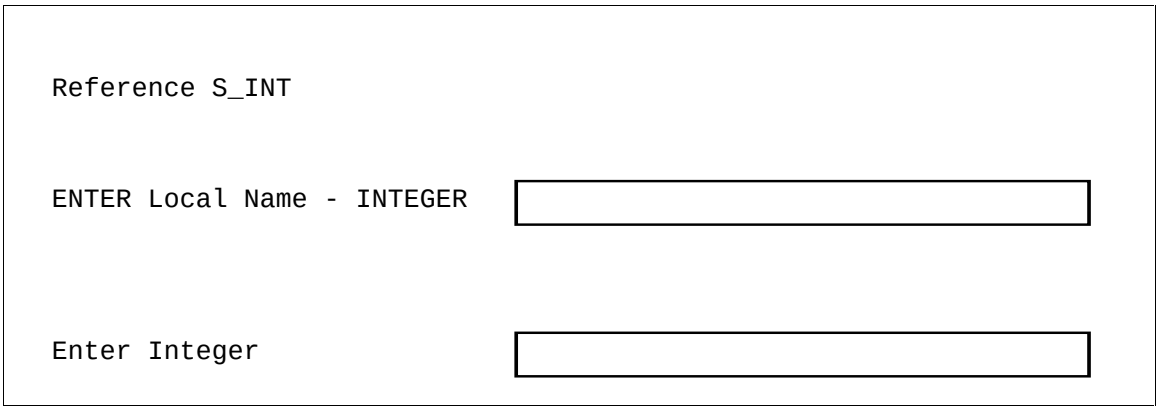
5.7 STORE INTEGER DATA INTO THE DISPLAY DATABASE

This actor allows Integer data to be stored into the Display Database.

Calling Sequence— S_INT(NAME,INTEGER)

Examples—S_INT(INT01,100)
S_INT(INT05G,250)

Optional Parameter Entry Form



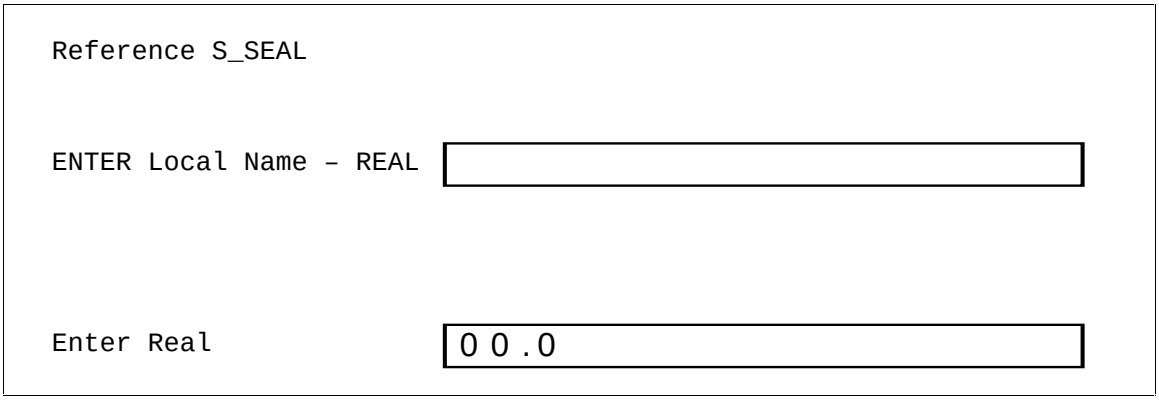
5.8 STORE REAL DATA INTO THE DISPLAY DATABASE

This actor allows Real data to be stored into the Display Database.

Calling Sequence— S_REAL(NAME,REAL)

Examples—S_REAL(REAL01,123.4)
S_REAL(REAL05G,62.3)

Optional Parameter Entry Form



5.9 STORE STANDARD ENUMERATION DATA INTO THE DISPLAY DATABASE

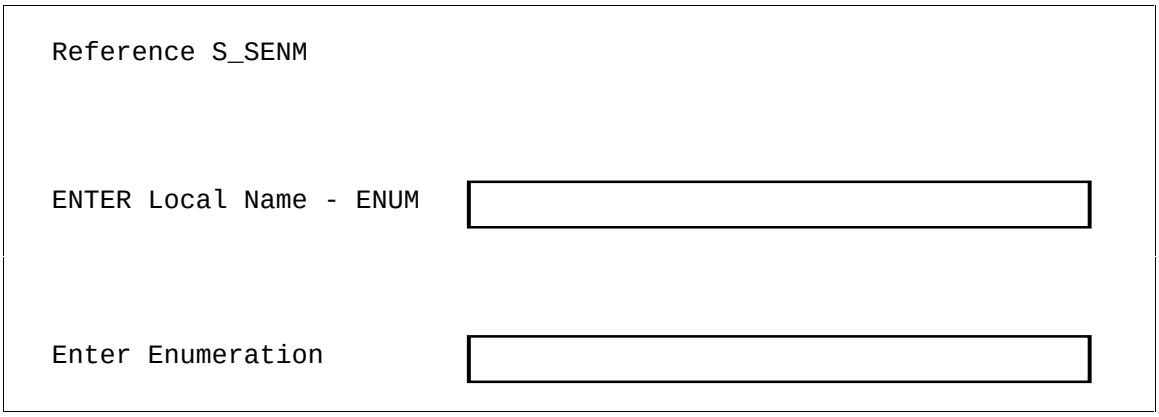
This actor allows Enumeration data to be stored into the Display Database. This actor operates only on Standard Enumerations.

Calling Sequence— S_SENM(NAME,ENUMERATION)

Examples—S_SENM(ENM01,R_SENM("MODE",0,1,20,"ENTER: AUTO or CAS or MAN",FALSE,2))

S_SENM(ENM01G,MODE:AUTO)

Optional Parameter Entry Form



Specification of the enumeration requires the set name followed by the member name, for example, UNIT:XXXX.

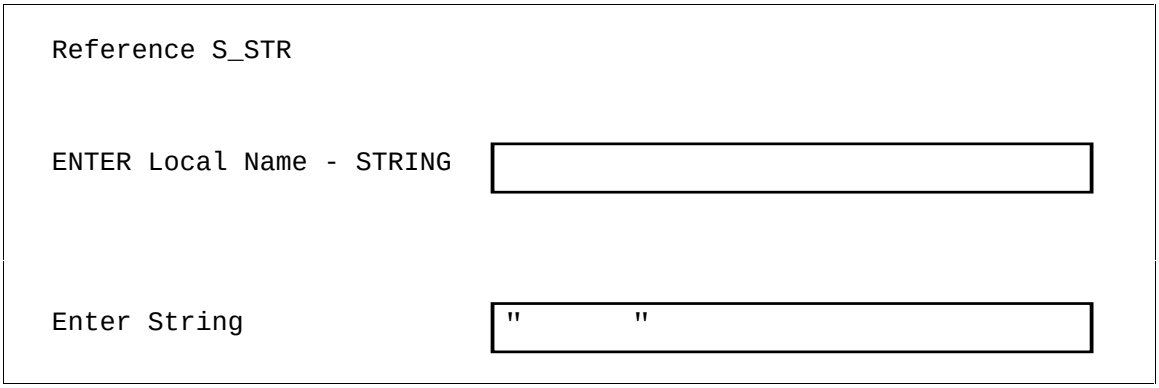
5.10 STORE STRING DATA INTO THE DISPLAY DATABASE

This actor allows String data to be stored into the Display Database.

Calling Sequence— S_STR(NAME,"STRING")

Examples—S_STR(STRING01,"AUTO")
S_STR(STR02G,"GLOBAL STRING")

Optional Parameter Entry Form



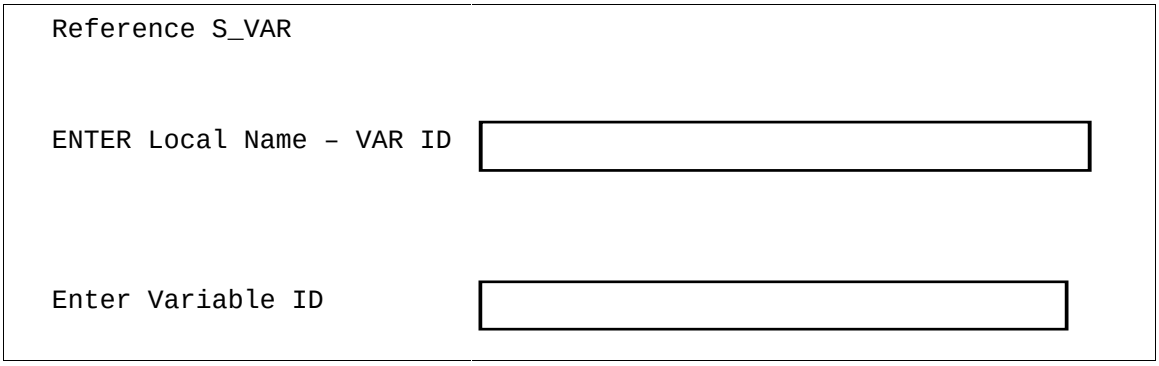
5.11 STORE VARIABLE ID INTO THE DISPLAY DATABASE

This actor allows a Variable ID (i.e., Point.Parameter) to be stored into the Display Database.

Calling Sequence— S_VAR(NAME,VARIABLE ID)

Examples—S_VAR(VAR01,HG100.OP)
S_VAR(VAR01G,A100.PV)

Optional Parameter Entry Form



6 READ AND STORE OPERATOR INPUT DATA INTO DISPLAY DATABASE

6.1 READ AND STORE INTO LOCAL DISPLAY DATABASE

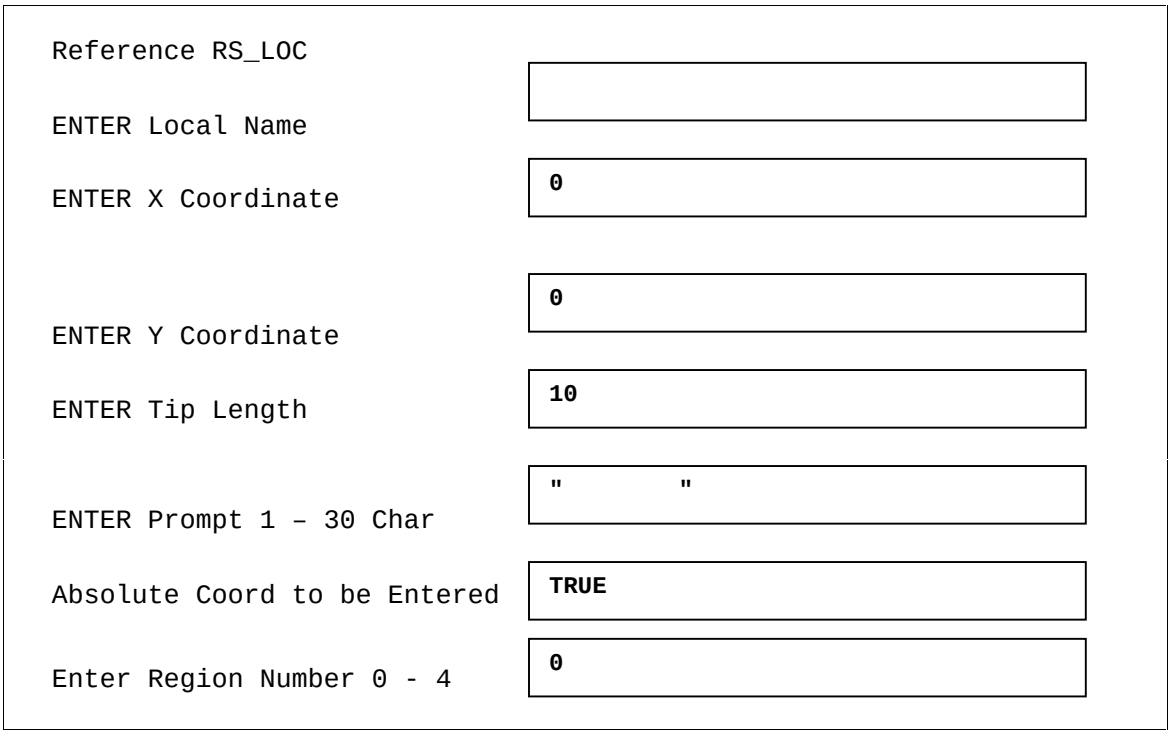
This actor allows you to specify the screen location for a Text Input Port (TIP), and have data entered through that TIP read and stored into the Display Database (DDB).

Note that if this actor is used to store a date/time DDB variable, the type is defaulted to Date. If you want the type to be Time, use the actor R_TIME (Read Time). If only Date is required, use the actor R_DATE (Read Date).

Calling Sequence— RS_LOC(NAME,X,Y,TIP LENGTH,"PROMPT",ABSOLUTE COORD, REGION NO)

Example— RS_LOC(REAL01,10,20,10,"ENTER VALUE",TRUE,2)

Optional Parameter Entry Form



Parameter Explanation— see next page.

Parameter Explanation

- Local Name— The name of the Display Database variable.
- X-Y Coordinate— The X and Y coordinates (in character units) of the text input port that is used to enter data. These coordinates specify the lower-left corner of the TIP. Be careful when creating or moving a target to ensure that the coordinates for the resulting TIP are within the specified limits of the screen (X = 0 through 79, Y = 0 through 25).
- TIP Length— The length of the Text Input Port in character units. This parameter can range from 1 through 80.
- Prompt— A prompt string that will query the operator for the value. The Length of the prompt string ranges from 1 to 30 characters and must be enclosed in quotes.
- Absolute Coord— Specifies the location of the Text Input Port as relative to the region or relative to the target.

  FALSE means to display the TIP at the X, Y coordinate relative to the target's upper left-hand corner.

  TRUE means to display the TIP at the X, Y coordinate relative to the origin of the region (see below).
- REGION NUMBER— The region that will contain the Text Input Port (see Figure 1-4 in Section 1). A region number equal to zero (0) means to use the region number where the target is located.

  Text Input Ports should not be placed on the lower line of region 1.
- CONFIGURABLE BUTTONS— The TIP location for an Operator Input actor that is activated by a configurable button is fixed at the top line of screen-region 1 (see Figure 1-4 in Section 1). **Only** the TIP LENGTH and PROMPT parameters are accepted.

NOTE

If a Text Input Port (TIP) is opened on top of a value that exists on the screen, the value that the TIP is on top of will overwrite an operator's entry and be used as the entered value. Always position text imports (ports) so they don't overlap any value objects.

NOTE

If the actor RS_LOC is used to store a data/time DDB variable such as DATIME1, the type is defaulted to date (RS_LOC can only read and store a date). If you want the type to be Time, use the actor R_Time along with the actor S_TIME. Similarly for enumerations, the actor RS_LOC should not be used for self-defining enumerations, only for standard enumerations.

7 READ DATA FROM DISPLAY DATABASE

These actors allow you to read data from the Display Database (DDB). This provides the ability to use the Display Database as a scratch-pad memory.

All Get actors have a single parameter that specifies the name of the DDB variable to be read. The actors return a value.

7.1 GET BOOLEAN DATA FROM DISPLAY DATABASE

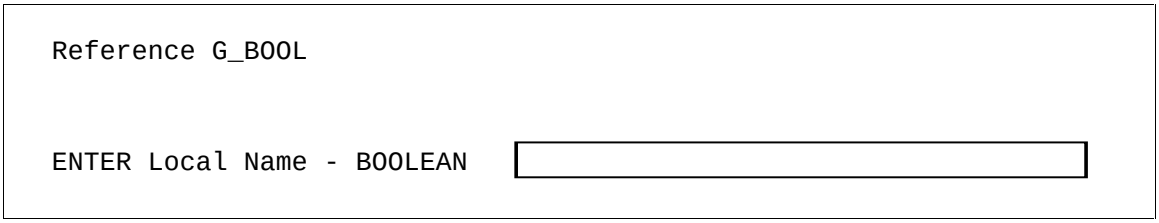
This actor allows Boolean data to be retrieved from the Display Database.

Calling Sequence— G_BOOL(DDB BOOLEAN VARIABLE)

A Boolean value is returned.

Example— S_BOOL(BOOL01,**G_BOOL**(BOOL02))

Optional Parameter Entry Form



7.2 GET DATE/TIME DATA FROM DISPLAY DATABASE

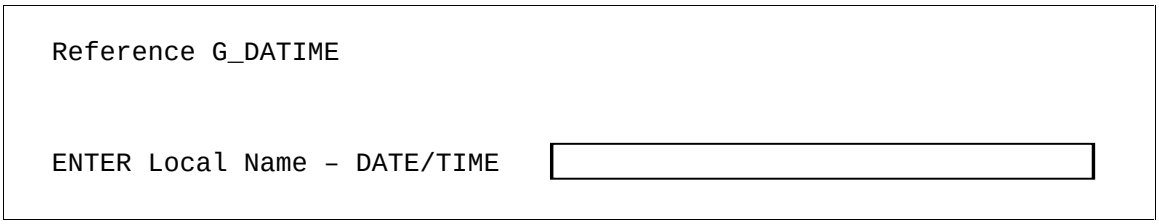
This actor allows DATE/TIME data to be retrieved from the Display Database.

Calling Sequence— G_DATIME(DDB DATE/TIME VAR)

Date/time is returned

Example— S_DATE(DATIME2,**G_DATIME**(DATIME1))

Optional Parameter Entry Form



7.3 GET ENTITY ID FROM DISPLAY DATABASE

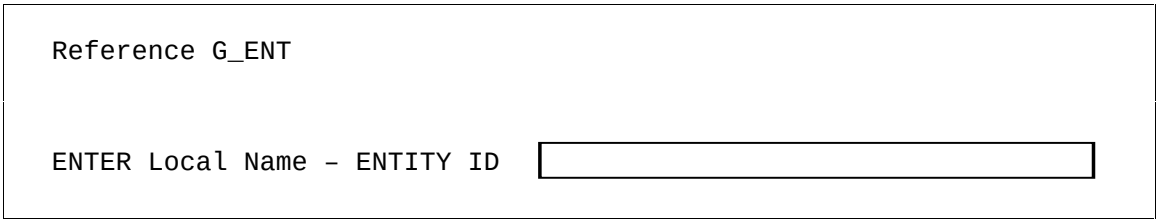
This actor allows an Entity ID to be retrieved from the Display Database.

Calling Sequence— G_ENT(DDB ENTITY ID VARIABLE)

An Entity ID is returned.

Example— SS_ENT(HG101.PRIMMOD,**G_ENT**(ENT01))

Optional Parameter Entry Form



7.4 GET INTEGER DATA FROM DISPLAY DATABASE

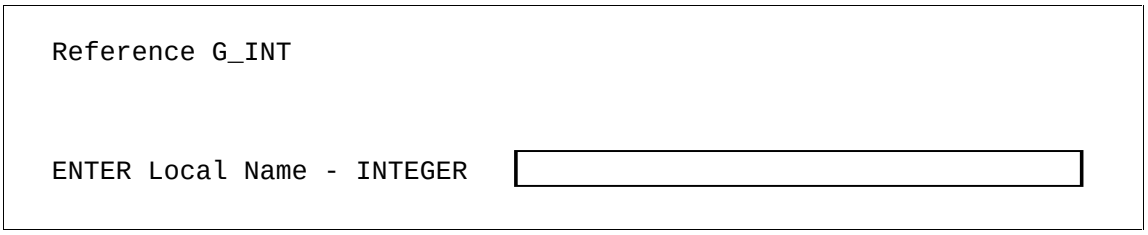
This actor allows Integer data to be retrieved from the Display Database.

Calling Sequence— G_INT(DDB INTEGER VARIABLE)

An Integer is returned.

Example— S_INT(INT02,**G_INT**(INT01))

Optional Parameter Entry Form



7.5 GET REAL DATA FROM DISPLAY DATABASE

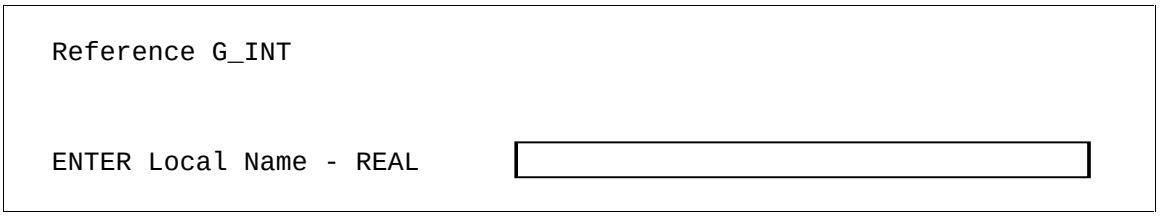
This actor allows Real data to be retrieved from the Display Database.

Calling Sequence— G_REAL(DDB REAL VARIABLE)

This actor returns Real data.

Example— SS_REAL(A100.PV,**G_REAL**(REAL02))

Optional Parameter Entry Form



7.6 GET STANDARD ENUMERATION DATA FROM DISPLAY DATABASE

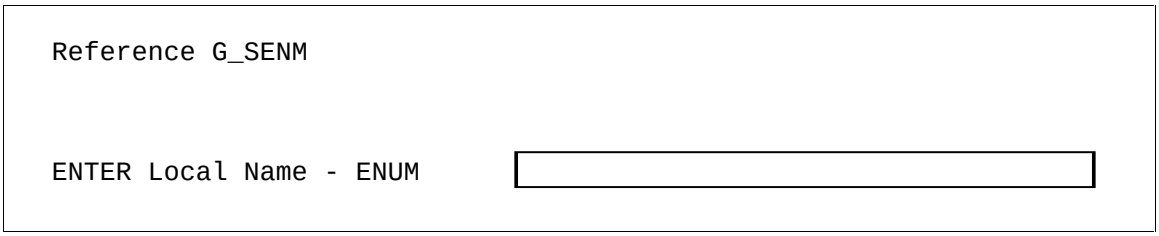
This actor allows Enumeration data to be retrieved from the Display Database. It operates on only standard enumerations.

Calling Sequence— G_SENM(DDB SENM VARIABLE)

An Enumeration value is returned.

Example— S_SENM(ENM01,**G_SENM**(ENM02))

Optional Parameter Entry Form



7.7 GET STRING DATA FROM DISPLAY DATABASE

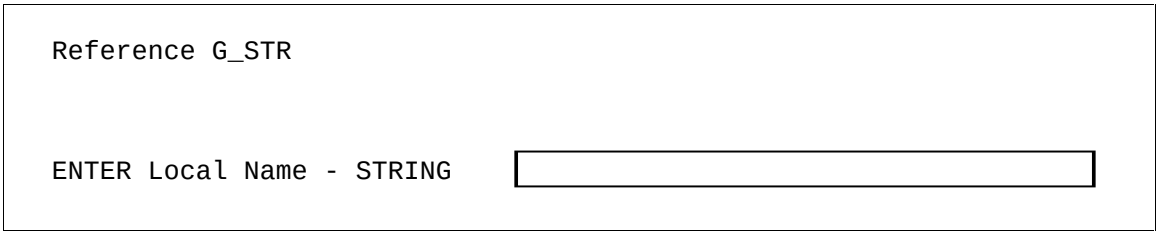
This actor allows String data to be retrieved from the Display Database.

Calling Sequence— G_STR(DDB STRING VARIABLE)

A String is returned.

Example— SS_STR(A100.OP,**G_STR**(STRING01))

Optional Parameter Entry Form



7.8 GET VARIABLE ID FROM DISPLAY DATABASE

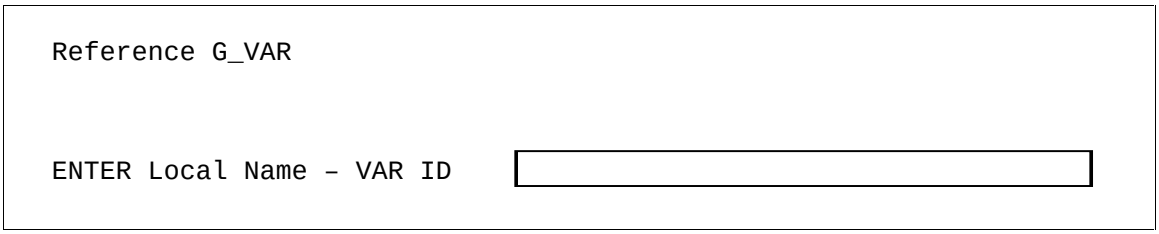
This actor allows a Variable ID to be retrieved from the Display Database.

Calling Sequence— G_VAR(DDB VARIABLE ID)

A Variable ID is returned.

Example— S_VAR(VAR01,**G_VAR**(VAR02))

Optional Parameter Entry Form



8 STORE INTO THE SYSTEM DATABASE

*These actors allow values to be stored into the **system** database.*

All of these actors take two parameters:

- *the first names the variable*
- *the second specifies the value*

In several cases, the examples contain custom parameters. These are only examples, and the calling sequence always indicates the appropriate type of entry.

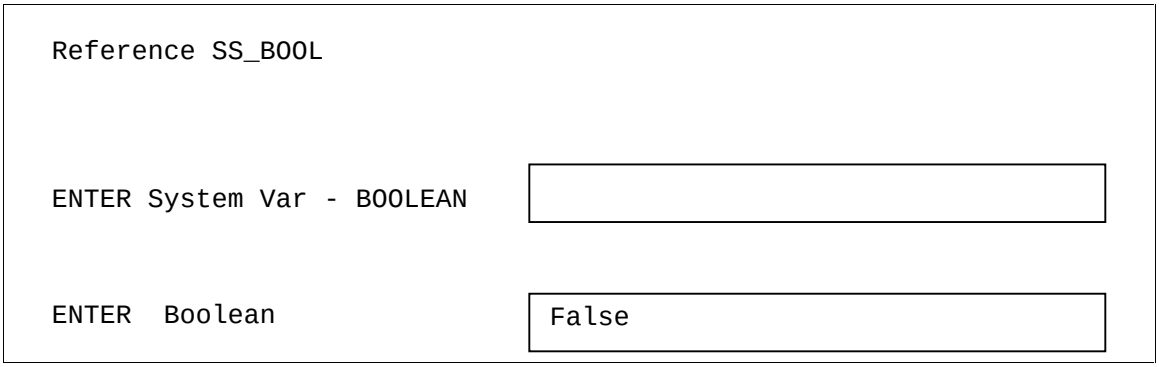
8.1 STORE BOOLEAN DATA INTO THE SYSTEM DATABASE

This actor allows Boolean data to be stored into the system database. When the actor successfully completes, the change is printed in the Real-Time Journal.

Calling Sequence— SS_BOOL(VARIABLE ID,BOOLEAN)

Examples—SS_BOOL(A100.PV,TRUE)
SS_BOOL(ENT01G.PPS,TRUE)

Optional Parameter Entry Form



8.2 STORE DATE INTO THE SYSTEM DATABASE

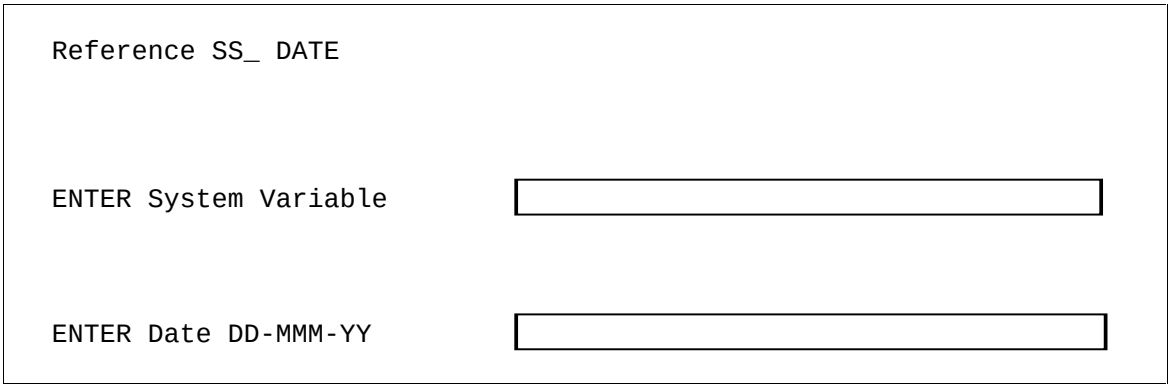
This actor allows Date data to be stored into the system database. When the actor successfully completes, the change is printed in the Real-Time Journal.

Calling Sequence— SS_DATE(VARIABLE ID,DATE)

Example— SS_DATE(AMTEST.CDS_DATE,05-24-94)

where AMTEST is a custom point and CDS_DATE is a custom parameter that contains a date.

Optional Parameter Entry Form



8.3 STORE DURATION DATA INTO THE SYSTEM DATABASE

This actor stores duration data into the system database. The Real-Time Journal prints a change if the action is successfully completed.

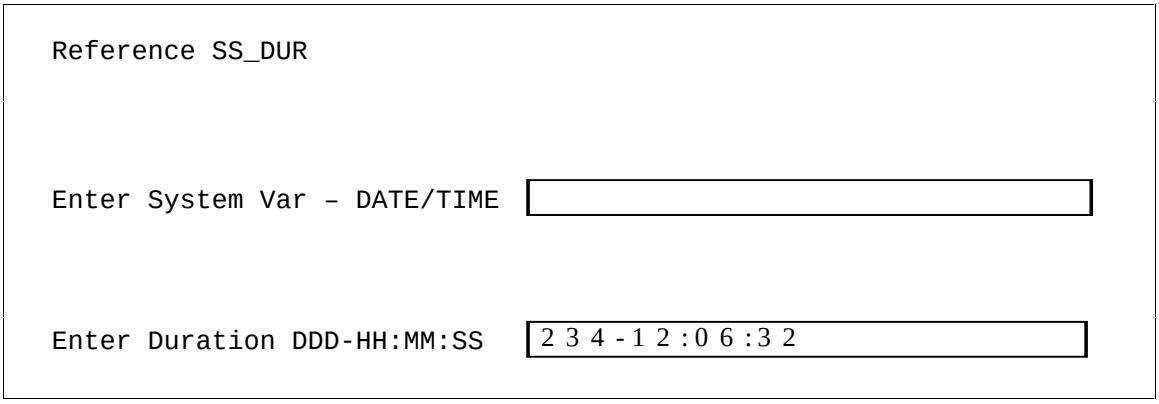
Calling Sequence— SS_DUR(VARIABLE ID,DURATION)

Where VariableID is a system variable and Duration is duration data in the format DDD-HH:MM:SS (see below).

Examples— SS_DUR(AMTEST.CDS_DUR,05:00:59)
SS_DUR(AMTEST.CDS_DUR,2-00:00:00)

where AMTEST is a custom point and CDS_DUR is a custom parameter that contains a duration.

Optional Parameter Entry Form



Parameter Explanation—

System Var—System Var is a system parameter or a custom parameter of the Time type.

Duration —Valid formats for the Duration entry are—

DDD-HH:MM:SS or HH:MM:SS or HH:MM

where DDD = the number of days in the duration
HH = the number of hours in the duration
MM = the number of minutes in the duration
SS = the number of seconds in the duration

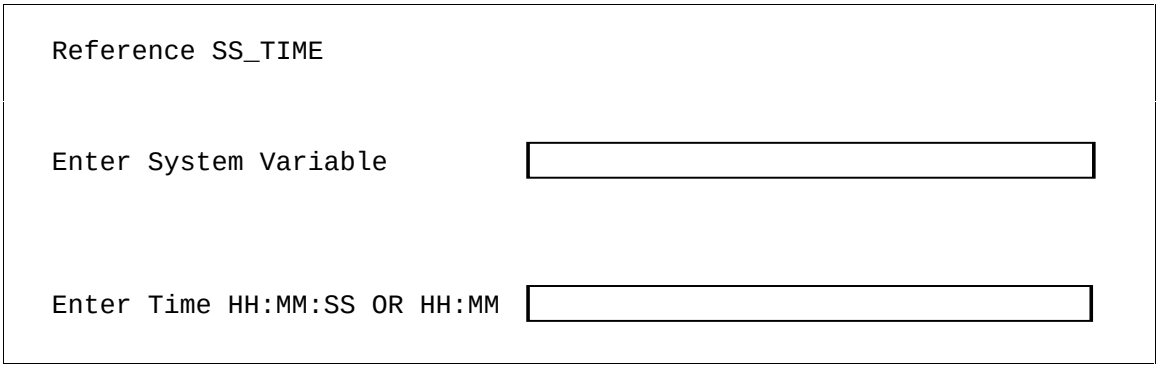
8.4 STORE TIME INTO THE SYSTEM DATABASE

This actor allows Time data to be stored into the system database. When the actor successfully completes, the change is printed in the Real-Time Journal.

Calling Sequence— SS_TIME(VARIABLE ID,TIME)

Example— SS_TIME(T123.TIME,11:09:54)

Optional Parameter Entry Form



8.5 STORE ENTITY ID INTO THE SYSTEM DATABASE

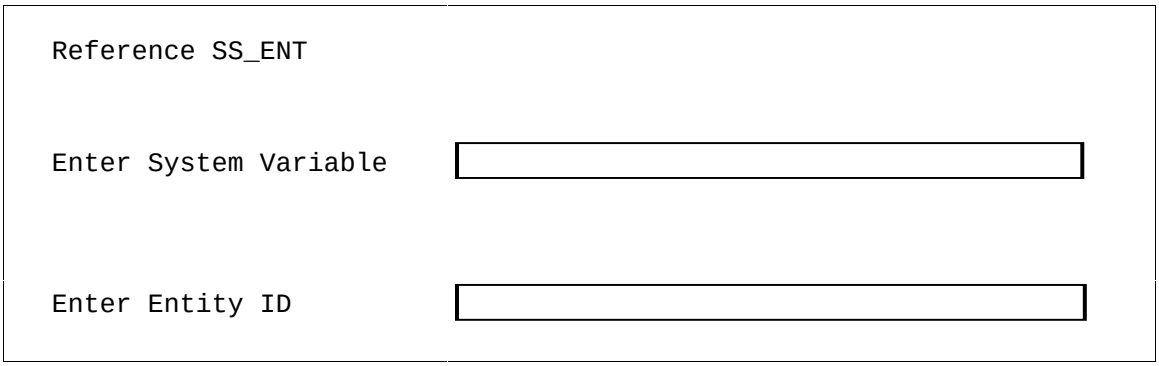
This actor allows an Entity ID to be stored into the system database. When the actor successfully completes, the change is printed in the Real-Time Journal.

Calling Sequence— SS_ENT(VARIABLE ID,Entity ID)

Example— SS_ENT(AMTEST.CDS_ENT,TIC100)

where AMTEST is a custom point and CDS_ENT is a custom parameter that contains a point name.

Optional Parameter Entry Form



8.6 STORE ENUMERATION DATA INTO THE SYSTEM DATABASE

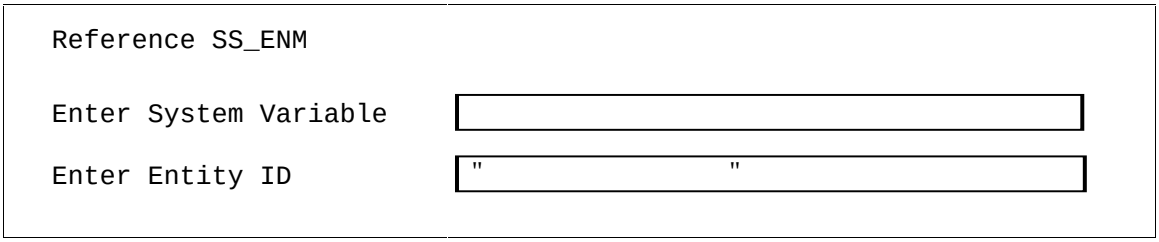
This actor allows Enumeration data to be stored into the system database. When the actor successfully completes, the change is printed in the Real-Time Journal.

Calling Sequence— SS_ENM(VARIABLE ID,STRING)

Example— SS_ENM(AMTEST.CDS_ENUM,"FORWARD")

where AMTEST is a custom point and CDS_ENUM is a custom parameter that contains a state name.

Optional Parameter Entry Form



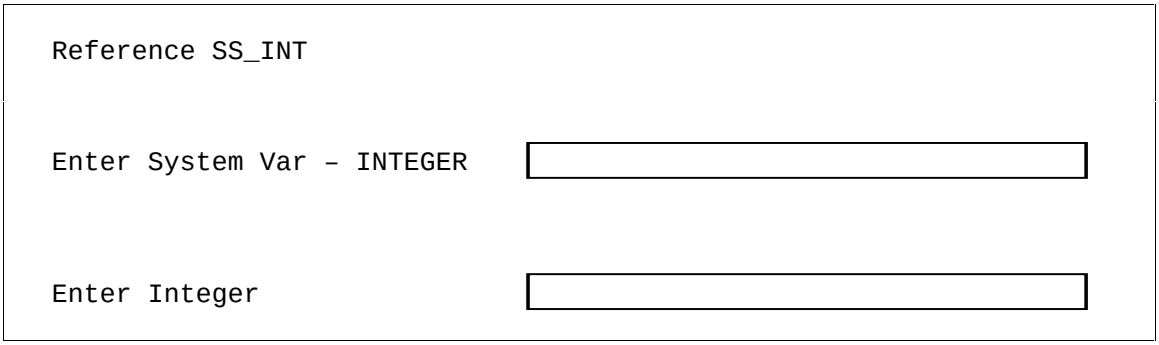
8.7 STORE INTEGER DATA INTO THE SYSTEM DATABASE

This actor allows Integer data to be stored into the system database. When the actor successfully completes, the change is printed in the Real-Time Journal.

Calling Sequence— SS_INT(VARIABLE ID,INTEGER)

Example— SS_INT(HG100.PV,123)

Optional Parameter Entry Form



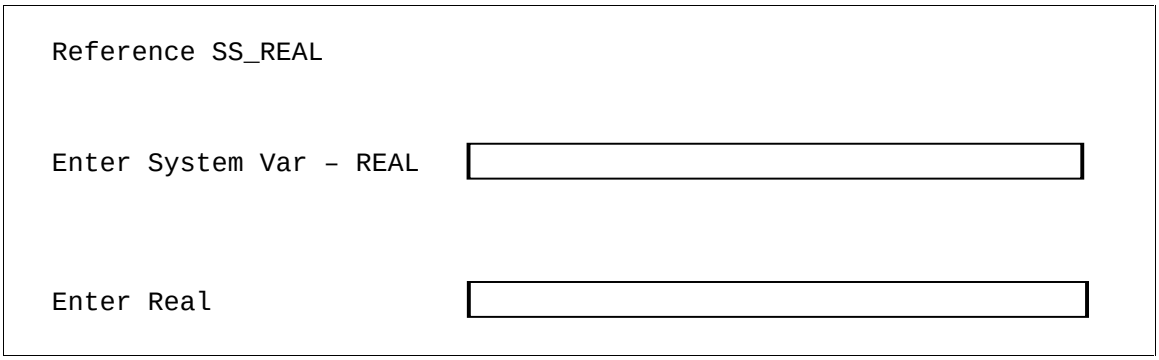
8.8 STORE REAL DATA INTO THE SYSTEM DATABASE

This actor allows Real data to be stored into the system database. When the actor successfully completes, the change is printed in the Real-Time Journal.

Calling Sequence— SS_REAL(VARIABLE ID,REAL)

Example— SS_REAL(R100.PV,100.0)

Optional Parameter Entry Form



8.9 STORE STRING DATA INTO THE SYSTEM DATABASE

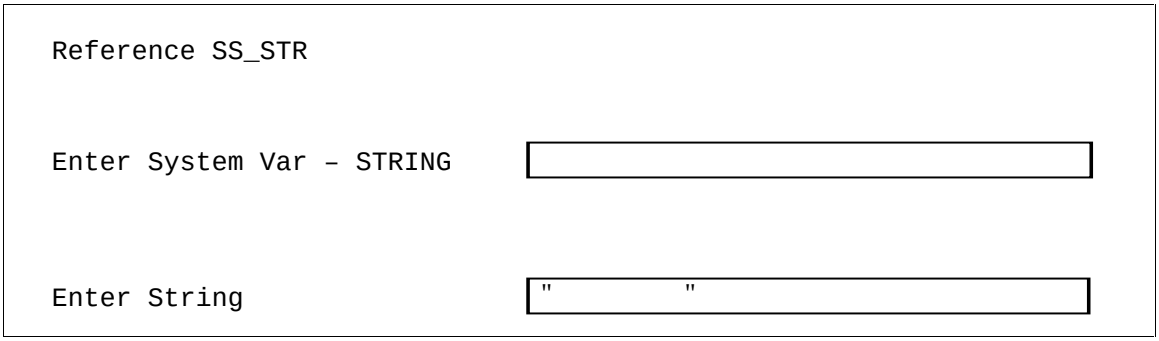
This actor allows String data to be stored into the system database. When the actor successfully completes, the change is printed in the Real-Time Journal.

Calling Sequence— SS_STR(VARIABLE ID,STRING)

Example— SS_STR(AMTEST.CDS_STR,"THIS IS A STRING")

where AMTEST is a custom point and CDS_STR is a custom parameter that contains ASCII characters.

Optional Parameter Entry Form



9 READ OPERATOR INPUT DATA AND STORE INTO SYSTEM DATABASE

These actors allow you to specify the screen location of a Text Input Port (TIP), and to have data entered through that TIP read and stored into the system database. When the actor successfully completes, the change is printed in the Real-Time Journal.

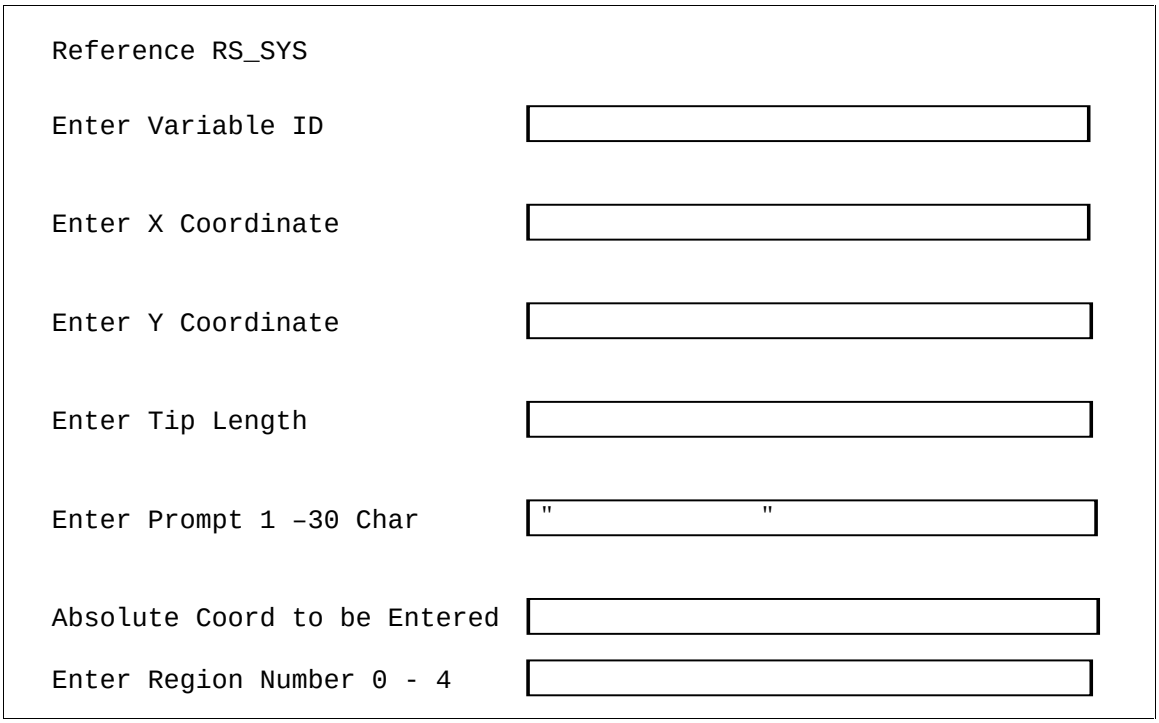
9.1 READ DATA AND STORE VARIABLE

This actor reads Variable data entered through a TIP and stores it into the system database.

Calling Sequence— RS_SYS(VARIABLE ID,X,Y,TIP LENGTH,"STRING", ABSOLUTE COORD,REGION)

Examples—RS_SYS(HG100.OP,0,0,15,"ENTER NEW VALUE",FALSE,0)
RS_SYS(&A.SP,0,0,15,"ENTER SETPOINT",FALSE,0)

Optional Parameter Entry Form



Parameter Explanation

- Variable ID—represents the variable name recognized on the network.

- X and Y coordinates—the coordinates (in character units) of the Text Input Port that is to be used to enter the data. These coordinates specify the lower-left corner of the port. Be careful when creating or moving a target to ensure that the coordinates for the resulting TIP are within the specified limits of the screen (X = 0 through 79, Y = 0 through 25).

- TIP Length—length of the Text Input Port in character units. This parameter can range from 1 through 80. Note that for longer tag name entries, a TIP length of 25 is needed to accept up to 16 characters for the entity, 1 character for the separator (.) and up to 8 characters for the parameter.

- Prompt—a prompt message that will query the operator for the value. The length of the prompt message can range from 1 through 30 characters and must be enclosed in quotes.

- Absolute Coord—Specifies the location of the Text Input Port as relative to the region or relative to the target.

  FALSE means to display the TIP at the X-Y coordinate relative to the target's upper left-hand corner.

  TRUE means to display the TIP at the specified absolute X-Y coordinate relative to the origin of the region.

- REGION NUMBER—The region that will contain the Text Input Port (see Figure 1-4 in Section 1).

- CONFIGURABLE BUTTONS—The TIP location for an Operator Input actor that is activated by a configurable button is fixed at the top line of screen-region 1 (see Figure 1-4 in Section 1). **Only** the TIP LENGTH and PROMPT parameters are accepted.

  Text Input Ports should not be placed on the lower line of region 1.

NOTE

If a Text Input Port (TIP) is opened on top of a value that exists on the screen, the value that the TIP is on top of will overwrite an operator's entry and be used as the entered value. Always position text imports (ports) so they don't overlap any value objects.

10 READ DATA FROM THE SYSTEM DATABASE

These actors allow data to be read from the system database. All of the actors take a single parameter that specifies the name of the variable to be read. The actor returns a value.

10.1 GET BOOLEAN DATA FROM THE SYSTEM DATABASE

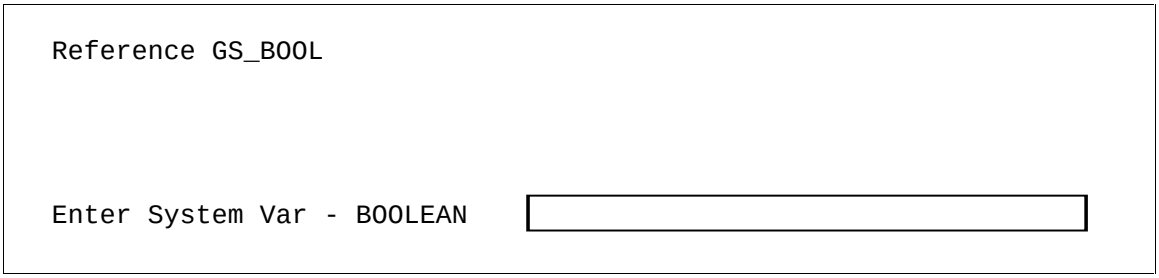
This actor allows Boolean data to be retrieved from the system database.

Calling Sequence— GS_BOOL(VARIABLE ID)

A Boolean value is returned.

Example— S_BOOL(BOOL01,**GS_BOOL**(A100.OPHIFL))

Optional Parameter Entry Form



10.2 GET DATE/TIME FROM THE SYSTEM DATABASE

This actor allows DATE/TIME data to be retrieved from the system database.

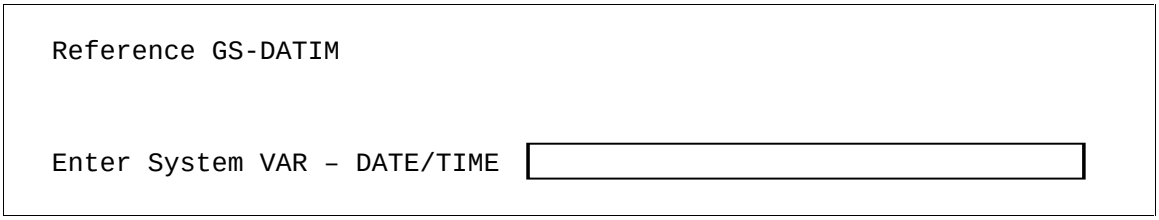
Calling Sequence— GS_DATIM(VARIABLE ID)

A date/time value is returned.

Example— S_DATE(DATIME1,**GS_DATIM**(AMTEST.CDS_TIME)

where AMTEST is an AM custom point and CDS_TIME is a custom parameter of the type Time.

Optional Parameter Entry Form



10.3 GET ENTITY ID FROM THE SYSTEM DATABASE

This actor allows an Entity ID to be retrieved from the system database.

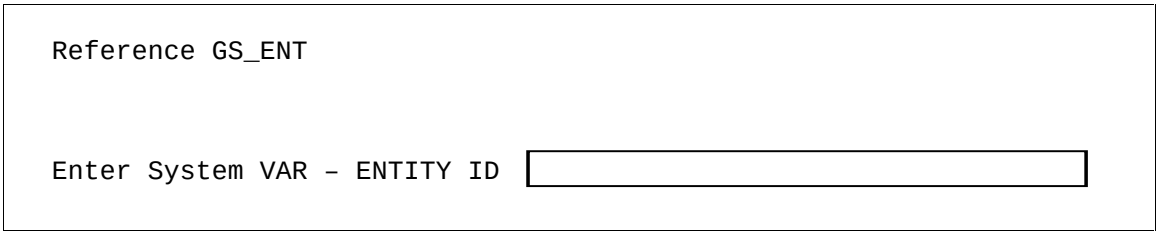
Calling Sequence— GS_ENT(VARIABLE ID)

This actor returns an internal Entity ID.

Example— S_ENT(ENT01,**GS_ENT**(AMTEST.CDS_ENT)

where AMTEST is an AM custom point and CDS_ENT is a custom parameter that contains a point name.

Optional Parameter Entry Form



10.4 GET ENUMERATION FROM THE SYSTEM DATABASE

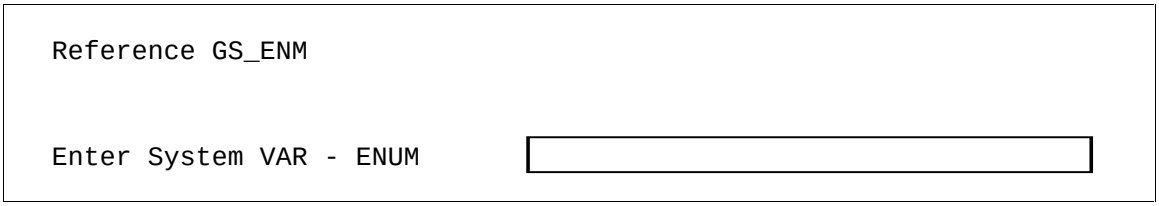
This actor allows a parameter's enumeration string name to be retrieved from a Variable ID in the system database. The data returned is a string.

Calling Sequence— GS_ENM(VARIABLE ID)

A String is returned. The string is a member of the Enumeration set.

Examples— SS_ENM(A100.MODE,**GS_ENM**(A100.MODE))
S_STR(STRING01,**GS_ENM**(ENT02G.OP))

Optional Parameter Entry Form



10.5 GET INTEGER DATA FROM THE SYSTEM DATABASE

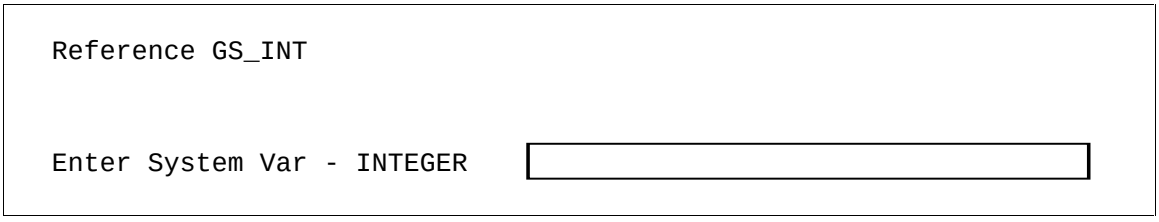
This actor allows Integer data to be retrieved from the system database.

Calling Sequence— GS_INT(VARIABLE ID)

This actor returns an Integer value.

Example— S_INT(INT01,**GS_INT**(A100.SLOTNUM))

Optional Parameter Entry Form



10.6 GET REAL DATA FROM THE SYSTEM DATABASE

This actor allows Real data to be retrieved from the system database.

Calling Sequence— GS_REAL(VARIABLE ID)

A Real value is returned.

Example— S_REAL(REAL01,**GS_REAL**(R100.PV))

Optional Parameter Entry Form

```
Reference GS_REAL


Enter System Var - REAL
```

10.7 GET STRING DATA FROM THE SYSTEM DATABASE

This actor allows String data to be retrieved from the system database.

Calling Sequence— GS_STR(VARIABLE ID)

This actor returns a String.

Examples— S_STR(STRING01,**GS_STR**(ENT01.NAME))
DETAIL(**GS_STR**(TIC100.NAME))

Optional Parameter Entry Form

```
Reference GS_STR


Enter System VAR - STRING
```

10.8 GET SYSTEM VARIABLE FROM THE SYSTEM DATABASE

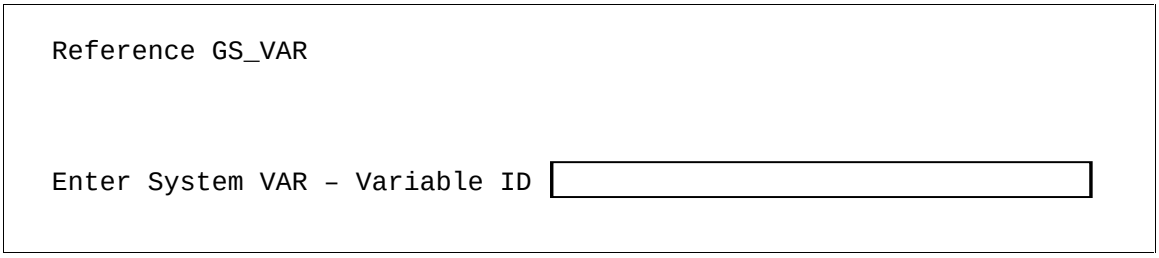
This actor allows a Variable ID to be retrieved from the system database.

Calling Sequence— GS_VAR(Variable ID)

This actor returns a Variable ID.

Examples—S_VAR(VAR01,**GS_VAR**(AMPT.CODSTN(1)))
S_VAR(VAR02G(**GS_VAR**(R_VAR(0,1,30,"ENTER POINT.PARAMETER",FALSE,0)))

Optional Parameter Entry Form



10.9 GET SYSTEM VARIABLE STRING FROM THE SYSTEM DATABASE

This actor allows a Variable ID (point.parameter) to be retrieved from the system database. The variable ID is converted to external form.

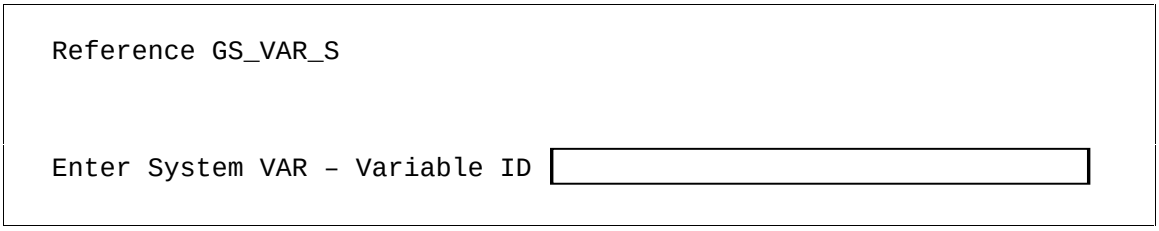
Calling Sequence— GS_VAR_S(VARIABLE ID)

This actor returns the value in the system data point as a string. Therefore, the actor must be used as a string input parameter for another actor.

Examples—

S_STR(STRING01,**GS_VAR_S**(AMPT.CODSTN(1)))
S_STR(STRING01,RNG_STR(**GS_VAR_S**(R_VAR(0,1,30,"ENTER POINT:",FALSE,0)),1,60))

Optional Parameter Entry Form



11 RANGE CHECKING ACTORS

This section describes actors that check to see if a value falls within a predefined limit. The actor then returns the value. In the cases of variable IDs and local DDB names, range checking amounts to type checking. These actors are recommended for use whenever a parameter that has a specified range is replaced by an operator-input actor. If a value is out of range, the error message Parameter Out Of Range *is presented to the operator.*

Consider the actor MODGROUP(402), which calls up Process Module Group display number 402. If you wished to replace this actor with one that allows the operator to specify the module group number, you might use the following:

MODGROUP(R_INT(35,1,8,"Enter Process Module Group", TRUE,1))

Here, the Read Integer actor (R_INT) takes operator input from a Text Input Port and supplies it as a parameter to the MODGROUP actor, which then calls up the specified display. But Process Module group numbers can only range from 401 through 450, and the operator might enter a number outside of the legitimate range, so a better approach is to use the action procedure

MODGROUP**(RNG_INT**(R_INT(35,1,8,"ENTER PROCESS MODULE GROUP",TRUE,1),401,450))

Now, the operator input is range-checked to make certain that it is an integer between 401 and 450. This approach prevents out-of-range values from being entered by operators at run time.

11.1 RANGE CHECK INTEGER DATA

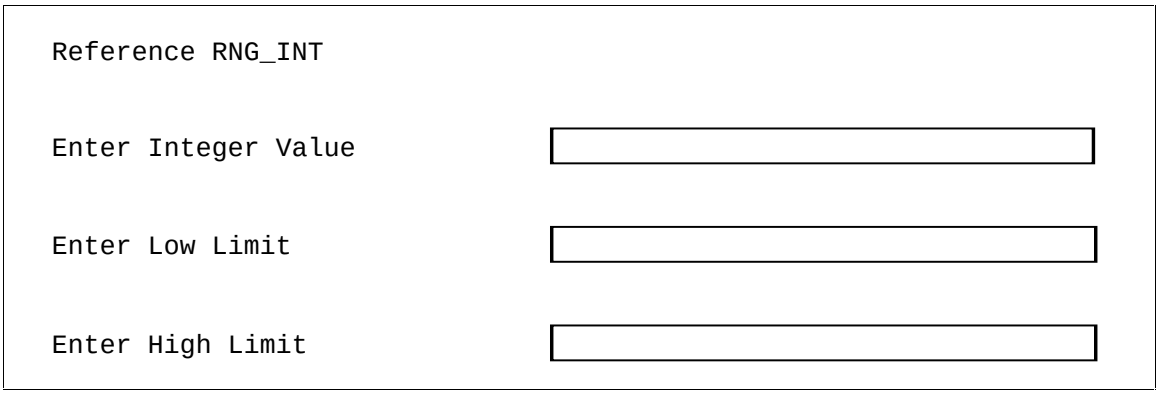
This actor checks the range of integer data.

Calling Sequence— RNG_INT(INTEGER VALUE,LOW LIMIT,HIGH LIMIT)

An Integer value is returned.

Example— See discussion on the previous page

Optional Parameter Entry Form



11.2 RANGE CHECK LOCAL DDB DATA

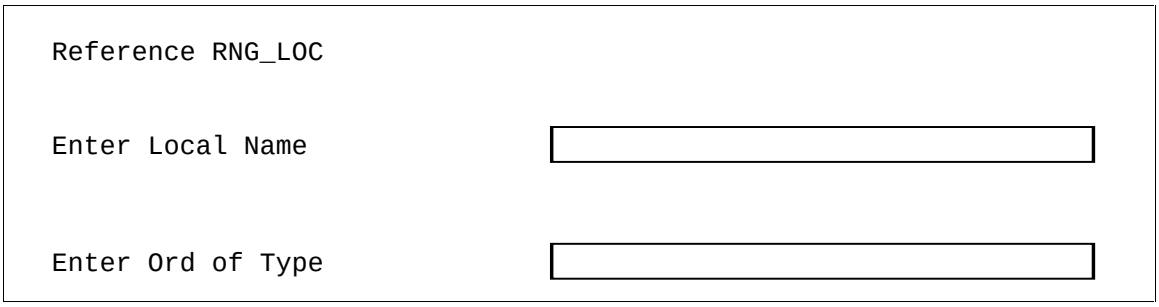
This actor checks the range of local or global DDB names by type (see Range Check Variable ID Data for types).

Calling Sequence— RNG_LOC(LOCAL VALUE,ORD OF TYPE)

This actor returns a Local DDB name.

Example— RS_LOC(**RNG_LOC**(INT01,0),0,1,20,"ENTER INTEGER",FALSE,1)

Optional Parameter Entry Form



Ord - the types are as follows:

Type	Ord
INTEGER	= 0
REAL	= 1
BOOLEAN	= 2
STRING	= 3
ENUMERATIONS	= 4
DATE/TIME	= 6
ENTITY ID	= 7
VARIABLE ID	= 8

11.3 RANGE CHECK REAL DATA

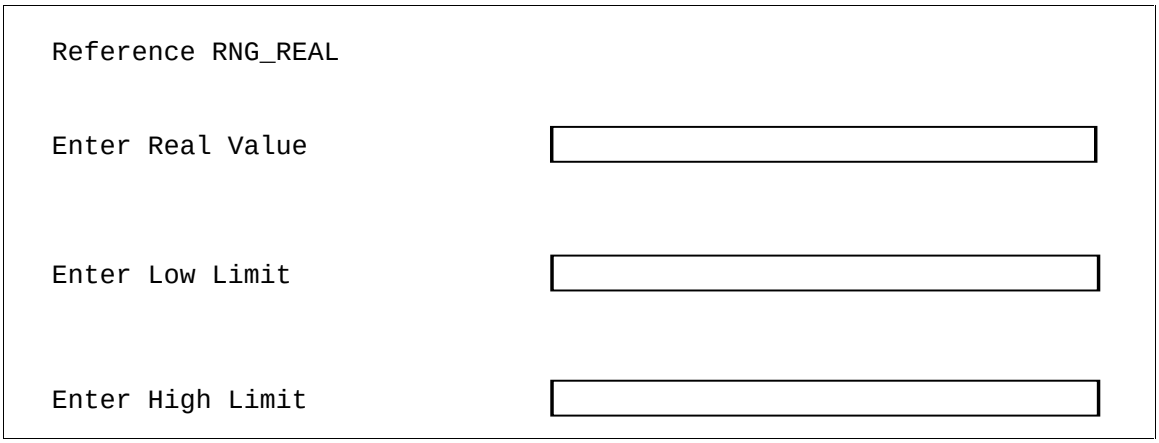
This actor allows range check of Real values.

Calling Sequence— RNG_REAL(REAL VALUE,LOW LIMIT,HIGH LIMIT)

A real value is returned.

Example— S_REAL(REAL05,**RNG_REAL**(R_REAL(35,1,8,"ENTER REAL VALUE",TRUE,1),1.5,99.5))

Optional Parameter Entry Form



11.4 RANGE CHECK STRING DATA

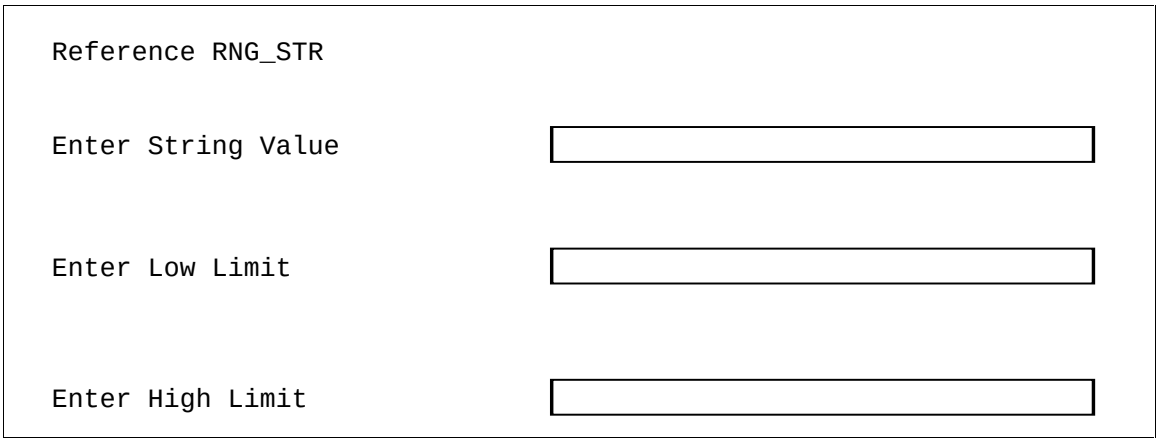
This actor checks the range of String values (by length).

Calling Sequence— RNG_STR(STRING VALUE,LOW LIMIT,HIGH LIMIT)

This actor returns a String value.

Example—S_STR(STRING01,**RNG_STR**(R_STR(35,1,20,"ENTER STRING",TRUE,1),5,20))

Optional Parameter Entry Form



11.5 RANGE CHECK VARIABLE ID DATA

This actor checks the range of Variable IDs (by type). The types are as follows:

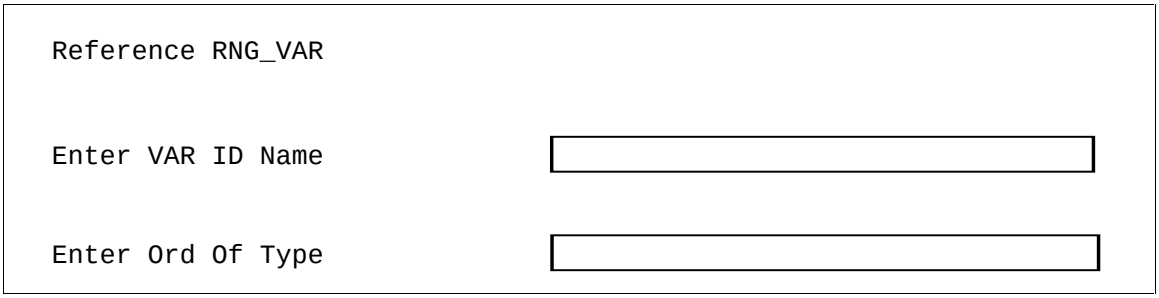
Type	Ord
INTEGER	= 0
REAL	= 1
BOOLEAN	= 2
STRING	= 3
ENUMERATIONS	= 4
DATE/TIME	= 6
ENTITY ID	= 7
VARIABLE ID	= 8

Calling Sequence— RNG_VAR(VAR ID,ORD OF TYPE)

This actor returns a Variable ID.

Example— S_VAR(VAR01,**RNG_VAR**(A100.PV,1))

Optional Parameter Entry Form



12 GENERAL ACTORS

12.1 AREA CHANGE

This actor initiates an Area change request for a specified Universal Station. The minimum keylevel necessary to change the Area depends on the level configured in the System Wide Values section of the Network Configuration File (NCF). The same keylevel requirement also applies to area changes from the Console Status display

The Area Database File must reside on the HM and have the pathname:

`NET>&Dnn>AREAnn.DA` where nn is the Area number

This actor is available for Release 520 and later systems.

This actor can initiate an area change for the station where it appears or for other stations (if so configured) however once an area change has been initiated from a US, that US cannot initiate another area change until the previous area change has completed. The other stations must be in the same console.

Upon completion of an Area change, the US that changed areas will attempt to call up a custom-built schematic if the schematic is stored by the file name `AREACHnn`, where `nn` is the Area that the US changes to. The Schematic's object file (AREACHnn.DO) must reside in the same directory as the Area Data Base file AREAnn.DA. If the schematic is found, prior display information is deleted. This means that pressing the Prior Display key is ineffective after an area change (until some other display has been called). If an AREACHnn schematic is not found, the Console Status display is called up.

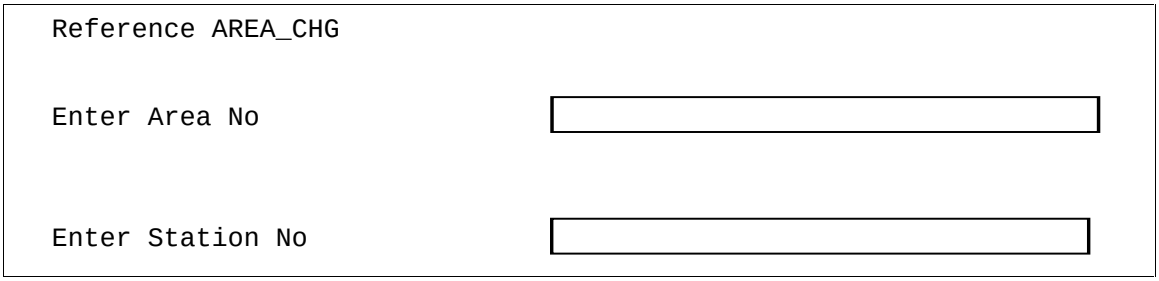
In Release 520, Area changes initiated from the Console Status Display also attempt to call up an AREACHnn schematic if one is built, named, and stored as described above.

Calling Sequence— AREA_CHG(AREA NUMBER,STATION NUMBER)

Example—AREA_CHG(02,08)

When selected, this actor changes Universal Station 8 to Area 2.

Optional Parameter Entry Form



Parameter Explanation—

AREA NUMBER is an Area number between 1 - 10 that the station will change to.

STATION NUMBER is a number between 0 - 10 for the Universal Station (US) that will change Areas. 0 indicates the local station. The US must be in the same console as the station that initiates the actor.

Error Messages—

If the keylevel at the station initiating an Area change is lower than required by the NCF System Wide Values configuration, the message: Function Keylocked appears.

If an Area database does not exist for the requested Area, the message: Area nn does not exist appears.

If an Area change is requested for a station not configured in the same console as the requestor, the message: Sta. n does not exist appears.

12.2 CHECK FOR VARIABLES WITH BAD VALUES (R600)

This new actor in R600 checks for variables with bad values. It also will accept array subscripted parameters as input. Parameters can be two levels deep which means they can have one level of indirection.

This actor takes a variable ID as an input parameter and returns:

1. TRUE if the value is bad.
2. FALSE if the value is good.

The input parameter will be considered a bad value if:

- The data owner is down
- The parameter is NULL
- The value is NAN
- If the point has been deleted
- If the point is invalid
- Any other condition exists that prevents a good value from being retrieved by data access.

NOTE

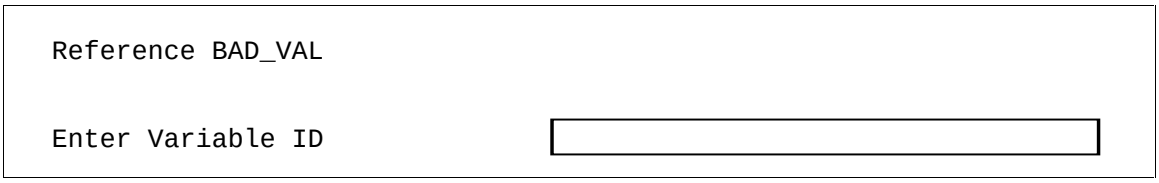
BAD_VAL will return FALSE if passed a reference parameter that is NULL, but will return TRUE if passed an indirect parameter that is referenced by a NULL parameter.

Calling Sequence— BAD_VAL(Variable ID)

Example— S_BOOL(BOOL01G,BAD_VAL(A100.ENT01(4).PV))

This is a valid example of an array subscripted parameter with one level of indirection.

Optional Parameter Entry Form



12.3 CHECK FOR NON-EXISTENT VARIABLES (R600)

This new actor in R600 checks for non-existent variables. It also will accept array subscripted parameters as input. Parameters can be two levels deep which means they can have one level of indirection.

This actor takes a variable ID as an input parameter and returns:

1. TRUE if the variable name exists.
2. FALSE if the variable name does not exist.

A variable is considered non-existent if:

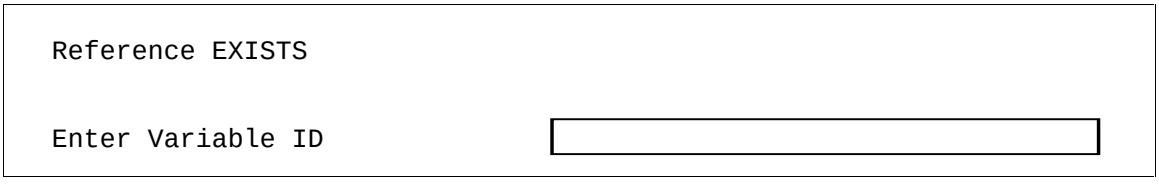
- It can not be fully resolved by data access.
- If the point resides on a data owner that is down, the EXISTS actor will be unable to determine if the parameter exists and will return FALSE.

Calling Sequence— EXISTS(Variable ID)

Example— IF(EXISTS(A200.OP)); {then logic}; ELSE; {else logic}; ENDIF

If the point in the example exists but does not have an OP, EXISTS will return FALSE. If A200 does not exist, EXISTS will return FALSE.

Optional Parameter Entry Form



12.4 COMBINE DATE/TIME

This actor combines a specific date with time. The values for date and time are stored in one variable. It can be a DDB such as DATIME1 or DATIME1G. The variable can also be a custom parameter of type Time.

The only way to store one of the values (date or time) into a parameter without changing the other is to use the C_DATTIM actor. Remember that date is on the left and time is on the right (07-17-94 14:12:00)

Calling Sequence— C_DATTIM(DD-MMM-YY,HH:MM:SS)

This actor returns a date time.

Examples—

To change both the date and time in the Global DDB DATIME1–

SS_DATE(DATIME1,**C_DATTIM**(R_DATE(0,0,10,"ENTER DATE",TRUE,0), R_TIME(0,0,10,"ENTER TIME",TRUE,0))

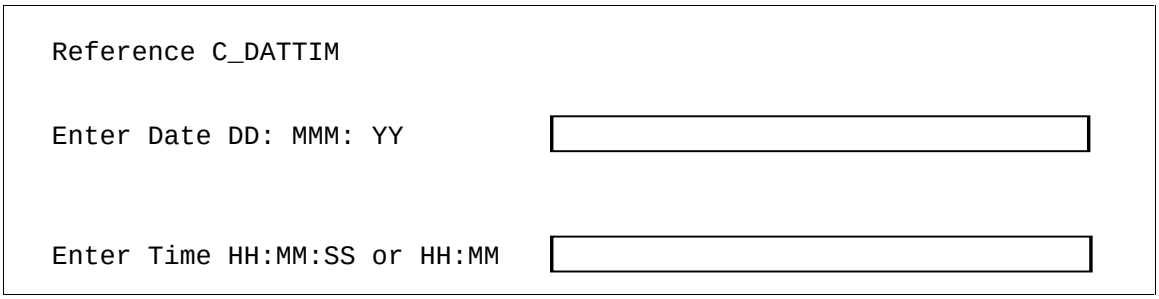
To read in a new date value from the operator without changing the time–

SS_DATE(TEST.DATIME,**C_DATTIM**(R_DATE(0,1,9,"ENTER DATE DDMMMYY",FALSE,0), GS_DATIM(TEST.DATETIME)))

To read in a new time value from the operator without changing the date–

SS_DATE(TEST.DATIME,**C_DATTIM**(GS_DATIM(TEST.DATETIME), R_TIME(0,1,8,"ENTER TIME HH MM SS",FALSE,0)))

Optional Parameter Entry Form



12.5 COMBINE VARIABLE (ENUMERATION INDEX)

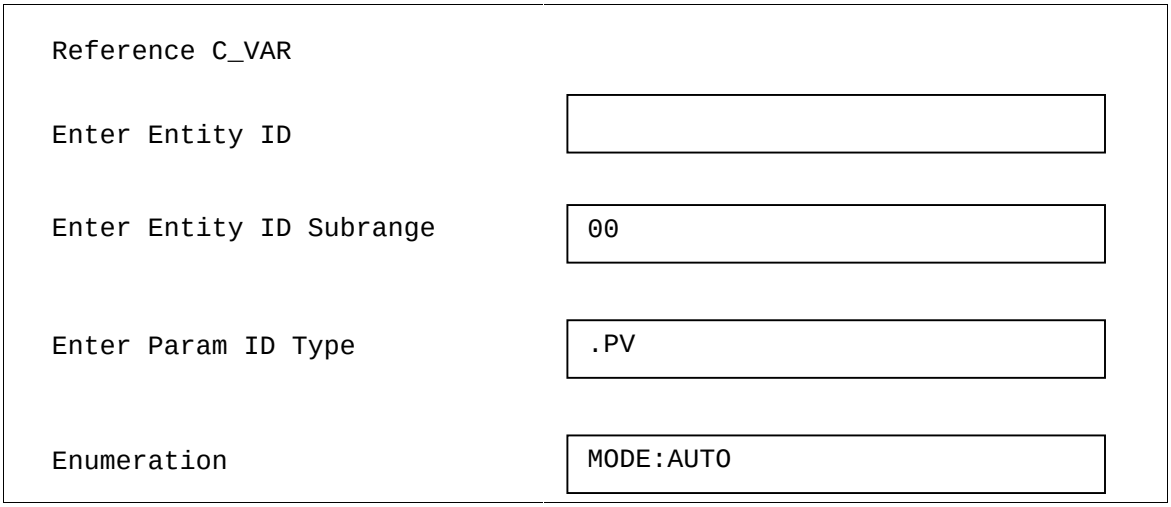
This actor combines an indexed entity and a parameter with an enumeration index to form a variable ID (e.g., Point Name**.**Parameter). An index value of zero (0) references just the entity.

Calling Sequence— C_V_ENUM(ENTITY,INDEX,PARAM,ENUM**:**<enum_type>)

A variable ID is returned.

Example— C_V_ENUM(HG1002,0,**.**PV,MODE**:**AUTO)

Optional Parameter Entry Form



CAUTION

Do not use a Remote System Variable in this actor. It could return erroneous information at run time.

12.6 COMBINE VARIABLE ID

This actor combines an indexed entity and an indexed parameter to form a variable ID (e.g., Point Name.Parameter). An index value of zero (0) references just the entity and a value of -9999 references just the parameter.

Calling Sequence— C_VAR(ENTITY,INDEX,PARAM,INDEX)

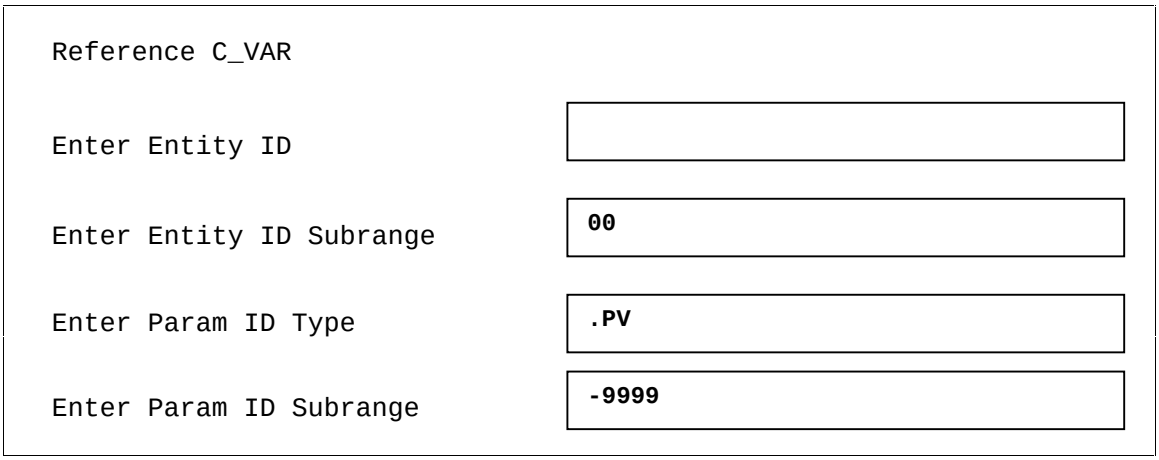
A variable ID is returned.

Example—

assume A100.P1(INT01+INT01G) is of type integer.
to store the number 44 into A100.P1(INT01+INT01G), use the following action phrase:

SS_INT(C_VAR(A100,0,.P1,ADD_I(G_INT(INT01),G_INT(INT01G))),44)

Optional Parameter Entry Form



CAUTION

Do not use a Remote System Variable in this actor. It could return erroneous information at run time.

12.7 CONCATENATE STRINGS

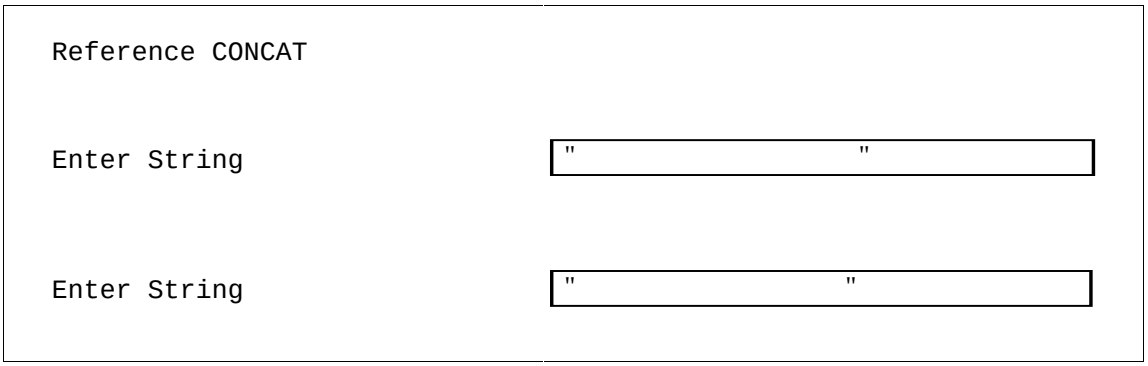
This actor concatenates two input strings and returns the result.

Calling Sequence— CONCAT(STRING1, STRING2)

A string value is returned.

Example—S_STR(STRING01,**CONCAT**("ABC",G_STR(STRING02))) String "ABC" is concatenated to the front of the string contained in STRING02 and stored in STRING01.

Optional Parameter Entry Form



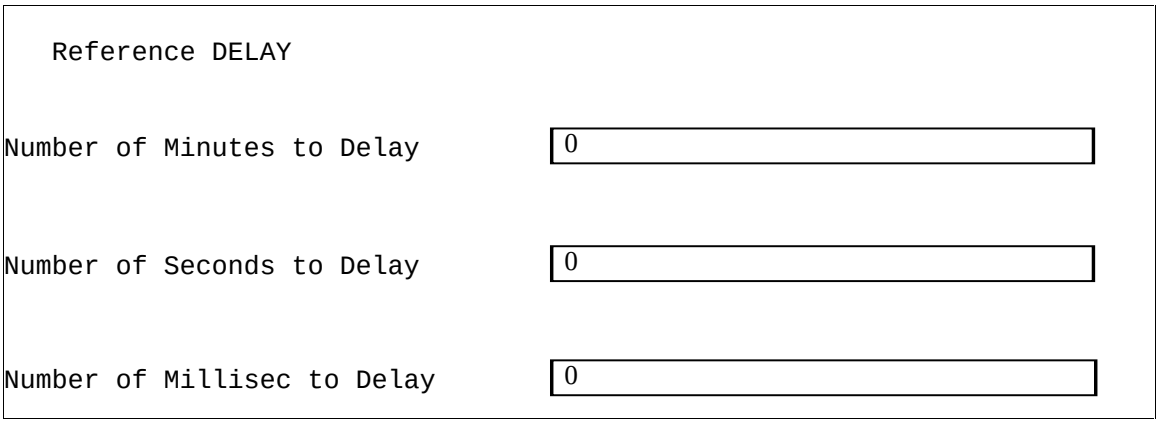
12.8 DELAY

This actor delays target or button actions for the specified period. During the delay period, all input from the keyboards, the touch screen, or cross-screen input to the Universal Station is queued, but not acted upon until the delay period is complete.

Calling Sequence— DELAY(MINUTES, SECONDS,MILLISECONDS)

Example— DELAY(0,8,500)

Optional Parameter Entry Form



Parameter Explanation—

MINUTES is the number of minutes to delay from 0 to 59

SECONDS is the number of seconds to delay from 0 to 59

MILLISECONDS is the number of milliseconds to delay from 0 to 1000

Note that if 0 is specified for all parameters, the time delay is virtually zero time.
If maximum values are specified, the delay period is one hour.

CAUTION

Because keyboard and touch-screen input are ignored during the delay period, the Universal Station will appear to be malfunctioning. For process applications, this actor should be used judiciously and the time delay should be minimal.

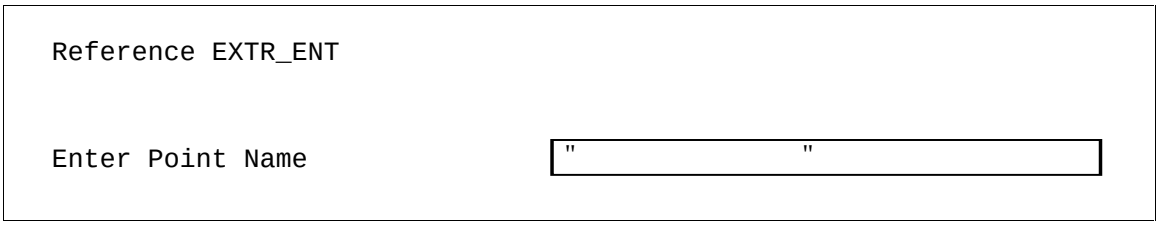
12.9 EXTRACT ENTITY NAME

This actor is used to extract the entity name from a string of data that is considered to be a point name containing both a network identifier and an entity name. The actor returns a string.

Calling Sequence— EXTR_ENT ("nnnn")
where nnnn is a valid point name.

Examples—S_STR(STRING01,**EXTR_ENT**(IE_ENT(ENT01)))
An entity name containing a network ID is converted to its external representation and the entity name is extracted and stored in String01.

Optional Parameter Entry Form



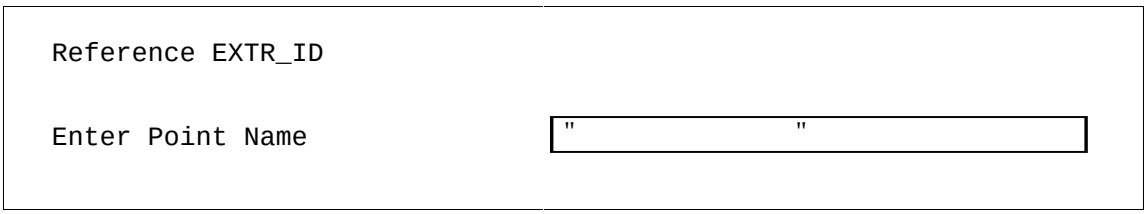
12.10 EXTRACT NETWORK ID

This actor is used to extract the network identifier from a string of data that is considered to be a point name containing both a network identifier and an entity name. It returns a string.

Calling Sequence— EXTR_ID ("nnnn")
where nnnn is a valid point name.

Example— S_STR(STRING01,**EXTR_ID**(IE_ENT(ENT01)))
An entity name containing a network ID is converted to its external representation and the network identifier is extracted and stored in String01.

Optional Parameter Entry Form



12.11 GET CONSOLE NUMBER

This actor returns the current console number from which the actor is called. The value that is returned by this actor is an integer. No input parameters are returned by this actor.

Calling Sequence—CON_NBR

Example—S_INT(INT01,**CON_NBR**)
An integer representing the current console number is returned and stored in INT01.

12.12 GET NODE NUMBER

This actor returns the current node number from which the actor is called. The value that is returned by this actor is an integer. No input parameters are returned by this actor.

Calling Sequence—NODE_NBR

Example—S_INT(INT01,**NODE_NBR**)
An integer representing the current node number is returned and stored in INT01.

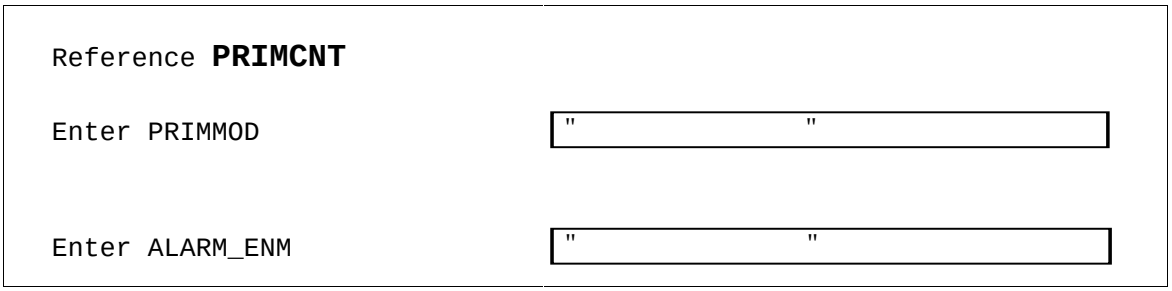
12.13 GET PRIMMOD COUNT

This actor is used to retrieve a Primmod or Multiple Primmod alarm count. It returns an integer that represents an alarm count for a specified alarm category. This actor is valid for Release 520 and later systems.

Calling Sequence— PRIMCNT ("PRIMMOD", ALARM_ENM)

Example — S_INT(INT01,**PRIMCNT** ("AM102", UNACKEM))
An integer representing the number of unacknowledged emergency alarms on primmod point AM102 is returned and stored in INT01.

Optional Parameter Entry Form



Parameter Explanation—

PRIMMOD is the name of Primmod or $MPRODn alarm group represented by a string of 1 - 16 characters.

ALARM_ENM is one of the following alarm categories for the PRIMMOD/$MPROD alarm group:

ACKEM	ACKHI	UNACKEM	UNACKHI
ACKLO	NOALARM	UNACKLO	

12.14 GET PRIMMOD STATUS

This actor is used to retrieve a Primmod or Multiple Primmod alarm status. It returns a string that represents the highest alarm category from a specified PRIMMOD/$MPROD alarm group. This actor is valid for Release 520 and later systems.

The string value returned is one of the following:

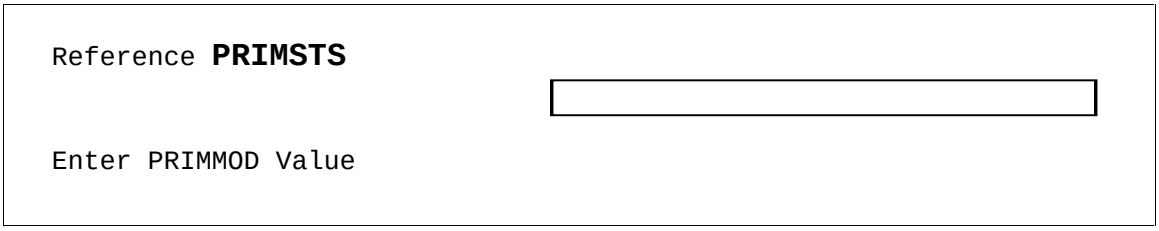
ACKEM	ACKHI	UNACKEM	UNACKHI
ACKLO	NOALARM	UNACKLO	

Calling Sequence— PRIMSTS("PRIMMOD")

Example— S_STR(STRING01,**PRIMSTS**("AM102"))

A string representing an alarm status on primmod point AM102 is returned and stored in STRING01.

Optional Parameter Entry Form



Parameter Explanation—

PRIMMOD is the name of Primmod or $MPRODn alarm group as represented by a string of 1 - 16 characters.

12.15 GET STATION NUMBER

This actor returns the current station number from which the actor is called. The value that is returned by this actor is an integer. No input parameters are returned by this actor.

Calling Sequence—STN_NBR

Example—S_INT(INT01,**STN_NBR**)

An integer representing the current station number is returned and stored in INT01.

12.16 NO OPERATION

This actor does nothing. It can be used to remove Define targets such as the Initial/Final target actions (refer to the Define Command).

Calling Sequence— NOP

12.17 OPTION

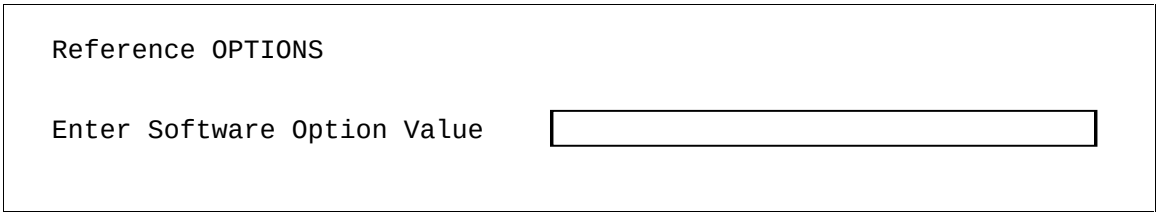
This actor is used to determine the state of a specified software option. If the software option is available, this actor returns a Boolean value of TRUE. If the software option is not available, FALSE is returned.

This actor's output must be used as an input source parameter in another actor that accepts a Boolean input.

Calling Sequence— OPTION(Option Integer)

Examples—S_BOOL(BOOL01,**OPTION**(**11**));
S_BOOL(BOOL01,**OPTION**(R_INT(1,1,50,"ENTER OPTION NUMBER:",1,0)));

Optional Parameter Entry Form



Parameter Explanation— The software option integer value has a range of +1 to +256.

> **NOTE**
>
> This actor is intended for internal use only by Honeywell development personnel.

13 TREND ACTORS

Trend actors are used to present or manipulate trend lines (traces) in either of two Application Subpictures, TREND_AX or TREND. The subpicture TREND_AX presents trend lines with axes; the subpicture TREND presents trend lines without axes (see Appendix G in the *Picture Editor Reference Manual*). Up to four simultaneous trend lines (traces) can appear in each subpicture.

Each use of a Trend or Trend_AX subpicture requires an associated Trend Record where trend data is gathered. Twelve Trend Records known as TREND01 through TREND12 are available. Trend Records 1 through 12 are designated in the actor calling sequences, by the corresponding Trend ID number (1 through 12).

When a display that contains trend subpictures is called up, all Trend Record traces are inactive and the time base is set to 20 minutes. The actors in this section add and delete traces, adjust scales, etc. The trend collectors in Appendix H of the *Picture Editor Reference Manual* can be used to associate traces colors with variables, data sources, the time base, etc.

Precautions—Note that trend actors cannot be called from a configurable button. Trend subpictures and targets that contain trend actors, must be in the same region.

13.1 ADD TRACE TO TREND

This actor is used to add a trend line (trace) to a TREND or TREND_AX Application Subpicture.

Traces are numbered in the order that they are added to a trend subpicture and adding a trace makes it active.

Of course, HM data is needed to show a trend line for an earlier period.

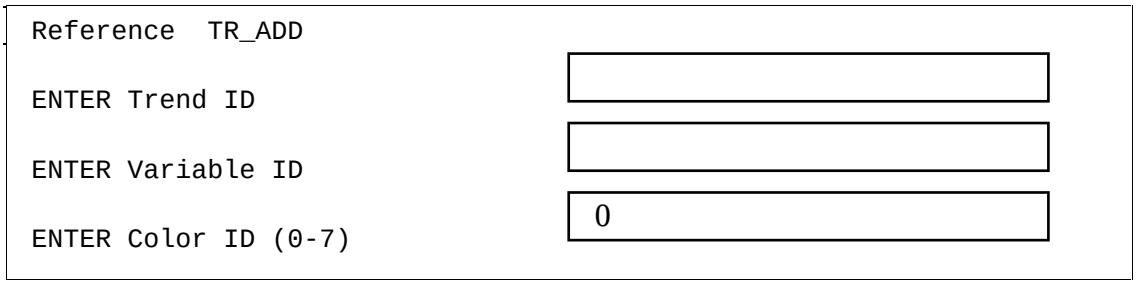
Calling Sequence— TR_ADD(TREND ID,VARIABLE ID,COLOR ID)

Example 1— TR_ADD(1,HG0501.PV,2)

Example 2*— TR_ADD(R_INT( ),R_VAR( ),R_INT( ))

*Example 2 shows how an actor could be configured to accept the parameters at operating time. Refer to the descriptions in Section 4 of this manual for more information about R_VAR, and R_INT. In practice, the parentheses would contain the mechanism to prompt the operator, and open a text input port.

Optional Parameter Entry Form



Parameter Explanation

TREND ID is an integer (from 1-12) that designates the Trend Record.

VARIABLE ID is any valid Point.Parameter combination that you want to see the trend.

COLOR ID specifies the trace color and can be any one of the following:

0 Next available color
1 Cyan
2 Yellow
3 Red
4 Green
5 Blue
6 Magenta
7 White

If a color isn't specified, the next available color for the trend record is assigned.

Example 3— A display is called up that contains a trend subpicture. The operator selects a target that contains the action sequence TR_ADD(1,AM100.PV,0). As a result of this action

AM100.PV is added to trend record 1 as trace 1.

The next available color (cyan) is assigned to the trace.

The default time-base is set to 20 minutes.

The trace becomes active.

AM100 is an AM point and the system determines that the most appropriate data source is the History Module (HM). The system tries to obtain 20, 1-minute snapshots from the HM. If it cannot obtain the information from the HM, it collects real-time data to generate the trace.

13.2 CENTERLINE TRENDING

This actor allows the centerline mode to be turned on or off in a trend display. It also sets the center line to the date and time specified in the actor. When centerline mode is on, trend data is presented with the requested date and time in the middle of the trend time axis. If the centerline mode is turned off, trend data does not change but the center line disappears.

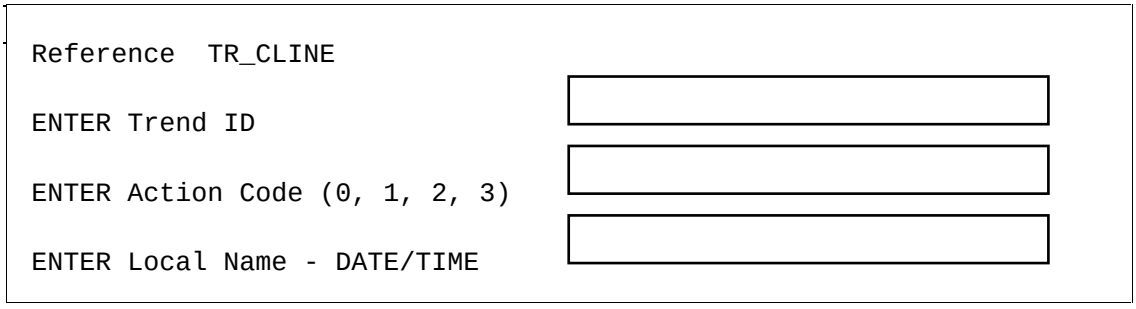
For HM data, if the time/date selected for the centerline is such that data has not yet been collected, the time at the right axis will automatically default to the current time and date, and the time at the centerline will be adjusted appropriately for the time base.

Also, refer to the TR_CLINE collector that provides the date and time for the center line.

Note that if this actor is used to switch center line trending on and off, the target sequence should include a way to re-label the time axes with the appropriate time label.

Calling Sequence— TR_CLINE(TREND ID,ACTION CODE, DATIME)

Optional Parameter Entry Form



TREND ID is an integer (from 1-12) that designates the Trend Record.

Action Code is an integer that indicates one of the following:

0 Turn off centerline trending (the date/time parameter is ignored).

1 Turn off centerline trending and cancel scrolling (the date/time parameter is ignored).

2 Turn on centerline trending without changing the trend data (the date/time parameter is ignored).

3 Turn on centerline trending. Update the trend graph data to display trend data for the time indicated by the center line date and time/parameter.

Example:

Local Name is a DATIME DDB parameter for the centerline date and time and must contain a valid date and time. If not (or if an error occurs in the date/time format) when the actor executes, one of the following messages results: Date Expected or Time Expected. If the date or time is in the future you will get the message: Future Times Not Allowed.

The objective is to switch centerline trending on and off in an existing trend display. One way to do this follows:

Use the Define Initial command to initialize a flag when the trend display is called up. For this example, configure the target action to set Boolean DDB BOOL01 false when the trend display is invoked.

Add a target to turn centerline trending on and off. Then the action sequence is

```
IF(G_BOOL(BOOL01);
  S_BOOL(BOOL01, FALSE);
  TR_CLINE(TREND01,0,DATIME1);
ELSE;
  S_BOOL(BOOL01,TRUE);
  TR_CLINE(TREND01,2,DATIME1);
ENDIF;
```

Label the target (for example, C-LINE) and add conditional behavior to change the target and text color depending on the flag status. For example, white if BOOL01 = true; green if BOOL01 = false.

To provide a time label for the centerline, add a Value to the center of the time axis. Enter TR_CLINE as the Value Expression. TR_CLINE is a collector used to return the date and time of the centerline.

To allow the operator to input the centerline date and time add an invisible target around the Value with the following target action—

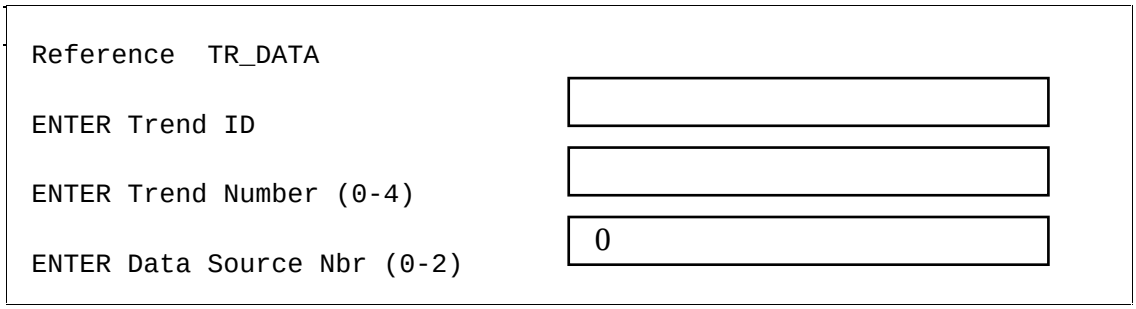
```
S_DATE(DATIME1,C_DATTIM(R_DATE(30,1,9,"Enter Centerline
Date",TRUE,1),R_TIME(30,1,5,"Enter Centerline Time",TRUE,1)));
TR_CLINE(1,3,DATIME1)
```

13.3 DEFINE DATA SOURCE FOR TREND TRACE

This actor sets the data source for the desired trace(s) to that designated by the parameter DATA SOURCE. If the specified data source is inappropriate, the system automatically attempts to switch to a source that is appropriate.

Calling Sequence— TR_DATA(TREND ID,TRACE NUMBER, DATA SOURCE)

Optional Parameter Entry Form



TREND ID is an integer (from 1-12) that designates the Trend Record.

TRACE NUMBER designates the trace for which the source of data is changed (0 means all traces). Note that trace numbers are determined by the order added/deleted and some care needs to be used. After four TR_ADD's, traces 1, 2, 3, and 4 exist. Subsequently, deleting trace 2 would cause trace 3 to become the new trace 2, and trace 4 to become the new trace 3. Trace 4 ceases to exist.

DATA SOURCE is one of the following:

0 HM (Continuous History)
1 HG (Data Hiway Box Trend Memory)
2 Real Time (current values)

Example 1— Trend Record 1 is operating with two traces. The data source for both traces is the HM. An operator selects a target with the action sequence TR_DATA(1,1,2). The result is that the data source for trace 1 is changed to real time.

Example 2— Trend Record 1 is operating with three traces. The data source for traces 1 and 2 is the HM. Trace 1 is associated with a hiway box that has trend memory. Trace 2 is associated with a digital point. The source for trace three, an AM point, is real time (the point is not historized). An operator selects a target with the action sequence TR_DATA(1,0,1). The result is

the data source for trace 1 is the HG,
the data source for trace 2 is the HM (HG was not a valid source),
the data source for trace 3 is real time (the only valid source).

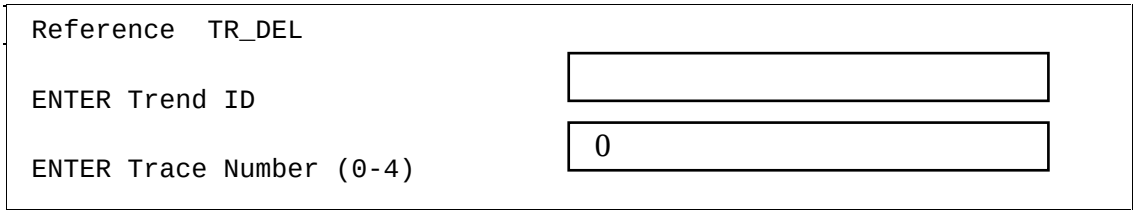
13.4 DELETE TRACE FROM TREND

This actor deletes a specific trace from a trend subpicture. Except when the highest numbered trace is deleted, the system automatically renumbers the remaining traces into consecutive order (see the example).

Special cases— The trend subpicture becomes inactive if it contains only one trace and that trace is deleted. If the trace being deleted is the only trace being scrolled and other traces are active, TR_DEL causes a return to current time and the remaining traces are redisplayed.

Calling Sequence— TR_DEL(TREND ID,TRACE NUMBER)

Optional Parameter Entry Form



Parameter Explanation

TREND ID is an integer from 1-to-12 that designates one of the 12 Trend Records.

TRACE NUMBER specifies which trace to delete (0 indicates all traces).

Example — An active trend subpicture (Trend Record 1) contains four traces. The operator selects a target with the action sequence TR_DEL(1,2). The following results:

trace 2 in Trend Record 1 is deleted.
traces 3 and 4 become trace 2 and 3, respectively.

Note that because there is no trace 4 at this point, the actor TR_DEL(1,4) would have no affect on this Trend Record.

13.5 TREND MODE

This actor selects the trend mode (and sometimes scrolling/update features) for the trend graph as specified by the action code. In general, the action code parameter allows selection of one of the following modes:

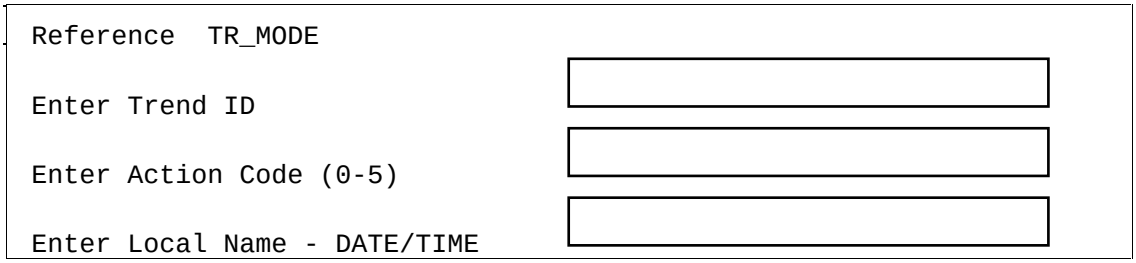
- right-axis
- center-line
- hair-line cursor

You can specify a time and date for the center line or for the position of the hair-line cursor. Other actions such as updating are combined with the mode selections as explained in the Action Code parameter description.

Note that this actor is available for use with R510 and later systems.

Calling Sequence— TR_MODE(TREND ID,ACTION CODE,LOCAL NAME)

Optional Parameter Entry Form



Parameter Explanation

Trend ID is an integer (from 1-12) that designates the Trend Record.

Action Code—These codes cause the following action:

0 - Change to right-axis mode without repositioning the trend graph data.
The date/time parameter is ignored.

1 - Change to right-axis mode and cancel scrolling.
The date/time parameter is ignored.

2 - Turn on center-line trending with no change to the trend data.
The date/time parameter is ignored.

2 - Turn on center-line trending and update the trend graph to display trend data for the time indicated by the trend graph mode date/time parameter.
The time indicated by the date/time parameter is centered on the trend graph time axis unless the time indicated causes the right axis to be a future time.

4 - Turn on hairline trending with no change to the trend data.
The date/time parameter is ignored.

5 - Turn on hair-line trending and update the trend graph data to display trend data for the time indicated by the trend graph mode date/time parameter.

Local Name—DATE/TIME is a DDB parameter for the center line or hair-line date and time. If no date and time are specified, the cursor defaults to center time position.

The date and time are specified as a 17-character string in Gregorian format where

MM/DD/YY HH:MM:SS

specify the month, day, year, hour, minutes and seconds.

If there is an error in the date/time format when the actor executes, one of the following messages results: Date Expected or Time Expected. If the date or time is in the future you will get the message: Future Times Not Allowed.

Example — An active trend subpicture (Trend Record 1) is in right axis mode and contains four traces. The operator selects a target with the action: TR_MODE(1,5,DATIME1).

The following results: The mode changes to hair-line, the date and time change to that specified in the DATIME1 parameter, and the trend graph is updated.

13.6 HAIR LINE CURSOR MOVE

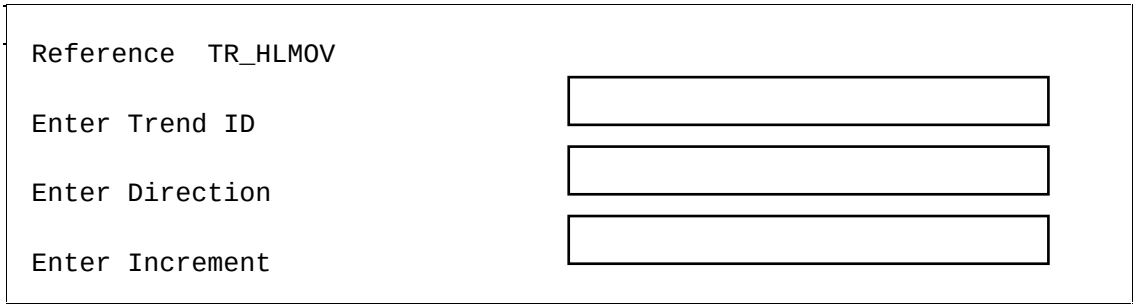
This actor moves the hair-line cursor to the right or left of its current position. The direction and increment of movement are specified as parameters to the actor.

The date/time, numeric value, etc. at the new cursor position are returned through the collectors TR_LTIME, TR_HLVAL, AND TR_HLSTS (refer to the Picture Editor Reference Manual, Appendix H)

Note that this actor is available for use with R510 and later systems.

Calling Sequence— TR_HLMOV(TREND ID,DIRECTION,INCREMENT)

Optional Parameter Entry Form



Parameter Explanation

Trend ID is an integer (from 1-12) that designates the Trend Record.

Direction designates the direction to move the hair-line cursor:

- TRUE = move the hair-line cursor to the left of its current position
- FALSE = move the hair-line cursor to the right of its current position

Increment is an integer that determines the distance to move the hair-line cursor. A larger number cause the cursor to move the greatest distance. If, however the increment value would move the cursor off the graph, movement is limited to the end of the graph in the direction specified and no error message results. The range is 1 - 480

Examples—

A target with the action `TR_HLMOV(1,TRUE,1)` moves the hair-line cursor 1 increment to the left.

A target with the action `TR_HLMOV(1,FALSE,10)` moves the hair-line cursor 10 increments to the right.

13.7 SAVE TREND SETTINGS

This R530 actor allows you to save a set of trend settings from a 4-point specified trend graph to a user-specified location on an HM or removable media. The current set of trend settings can be saved to a specified file. Specification of a trend graph and file location are implementation-dependent. The elements of saved trend settings are

- the trend's current Time Base,
- the trend's Y-Scale range,
- up to four Point Ids (Point Parameter),
- each Point Id's associated trend display range
- each Point Id's associated Data Source,
- each Point Parameter's trend trace color

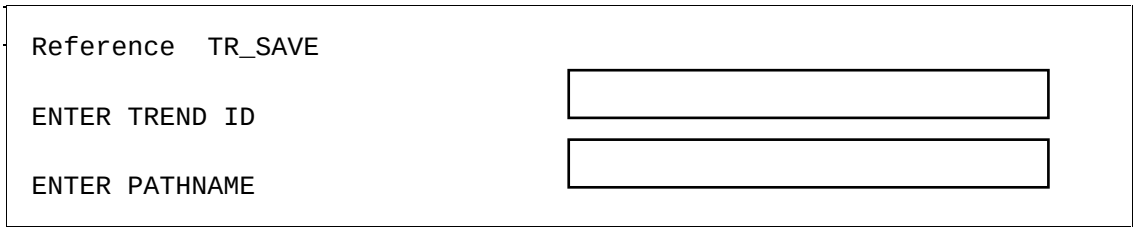
The set of trend settings to be saved for each trend graph are the same, regardless of implemented trend functionality.

This actor accepts runtime determination of the input parameter values. You can configure the Trend_Id and Pathname parameters to request from the user, at runtime, the desired Trend_Id and fully qualified pathname.

Calling Sequence— TR_SAVE(TREND_ID,PATHNAME)

Example—TR_SAVE(R_INT(30,2,1,"ENTER TREND ID (1-12)", TRUE, 1),
R_STR(30,2,30,"ENTER PATH AND FILENAME",TRUE,1));

Optional Parameter Entry Form



TREND ID is an integer (from 1-12) that designates the Trend Record.
PATHNAME is a fully qualified path and filename. (NET>TEST>TRENDAR1.SS)

13.8 DISPLAY (RESTORE) SAVED TREND SETTINGS

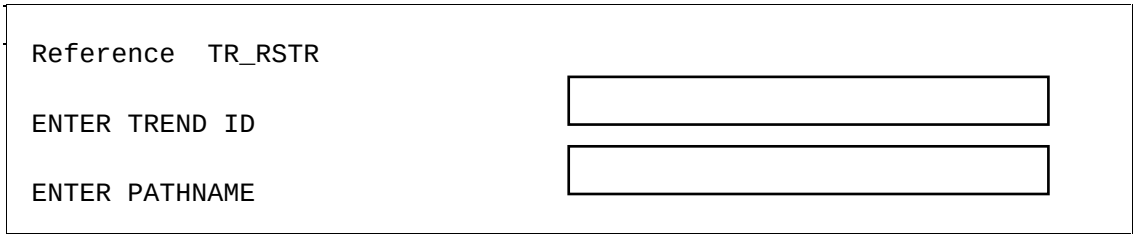
This R530 actor allows you to incorporate previously saved trend settings from an HM or removable media into a specified trend graph.

This actor accepts runtime determination of the input parameter values. You can configure the Trend_Id and Pathname parameters to request from the user, at runtime, the desired Trend_Id and fully qualified pathname.

Calling Sequence— TR_RSTR(TREND_ID,PATHNAME)

Example—TR_RSTR(R_INT(30,2,1,"ENTER TREND ID (1-12)", TRUE, 1),
R_STR(30,2,30,"ENTER PATH AND FILENAME",TRUE,1));

Optional Parameter Entry Form



Parameter Explanation

TREND ID is an integer (from 1-12) that designates the Trend Record.
PATHNAME is a fully qualified path and filename. (NET>TEST>TRENDAR1.SS)

13.9 SET RANGE FOR TREND TRACE

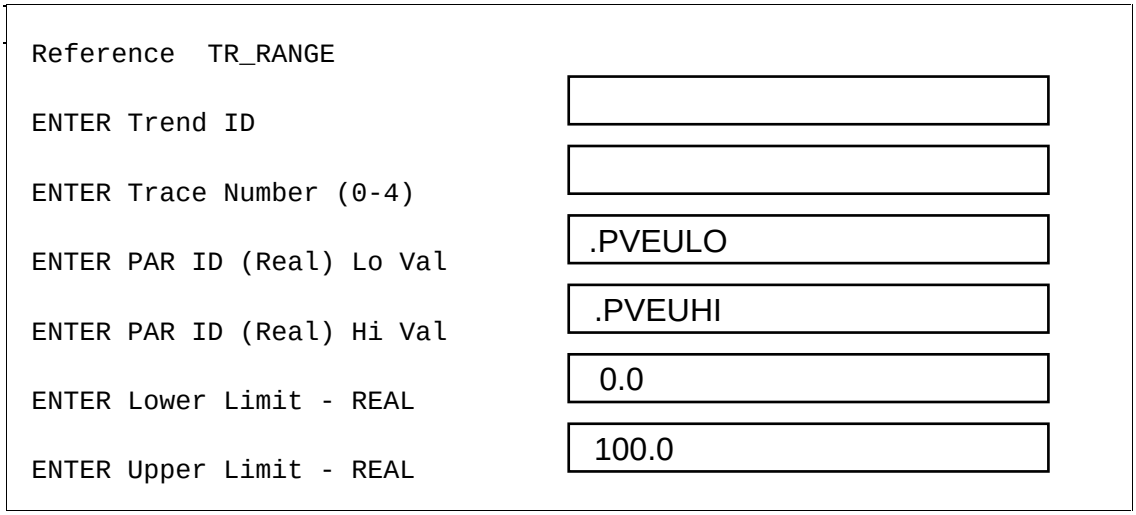
This actor sets the range for any or all traces in the designated Trend Record. If you want the new range to be taken from parameters of the point, the Parameter IDs should be entered (in which case the values for Low limit and High Limit are ignored). A valid parameter ID is a legal, non-subscripted parameter name of type Real. The default range parameters are usually PVEULO and PVEUHI. Other possible parameters are RANGELO and RANGEHI (for AM points) or SP.

If a constant value is desired for either or both range limits, the Parameter ID entries for those limits should be entered as ._ _ _ _ _ _ _ _ (period followed by eight underscores), and the appropriate constant values will be used. The default constants are 0.0 and 100.0. The change will occur with the next trend update.

Calling Sequence— TR_RANGE(TREND ID,TRACE NUMBER,PAR ID LOW,PAR ID HIGH,LOW LIMIT,HIGH LIMIT)

Example—TR_RANGE(1,1,.PVEULO,.PVEUHI,0.0,0.0)
TR_RANGE(1,1,.RANGELO,._ _ _ _ _ _ _ _,0.0,100.0)
TR_RANGE(1,1,._ _ _ _ _ _ _ _,._ _ _ _ _ _ _ _,50.0,150.0)

Optional Parameter Entry Form



Parameter Explanation

TREND ID is an integer (from 1-12) that designates the Trend Record.
TRACE NUMBER is the trace for which you want to set the range (0 means all).
PAR ID Low is the Parameter ID of a Real for the desired Low Limit.
PAR ID High is the Parameter ID of a Real for the desired High Limit.
LOW LIMIT is a Real to be used if PAR ID LO is entered as ._ _ _ _ _ _ _ _
HIGH LIMIT is a Real to be used if PAR ID LO is entered as ._ _ _ _ _ _ _ _

Example— Trend Record 1 contains four traces. The default parameters PVEULO AND PVEUHI were used and resulted in range limits of 0.0 to 100.0. The operator selects a target with the action sequence TR_RANGE(1,2,._ _ _ _ _ _ _ _ OPHILM,-100.0,0,0).
The result is

the low range for trace 2 changes to -100.0.

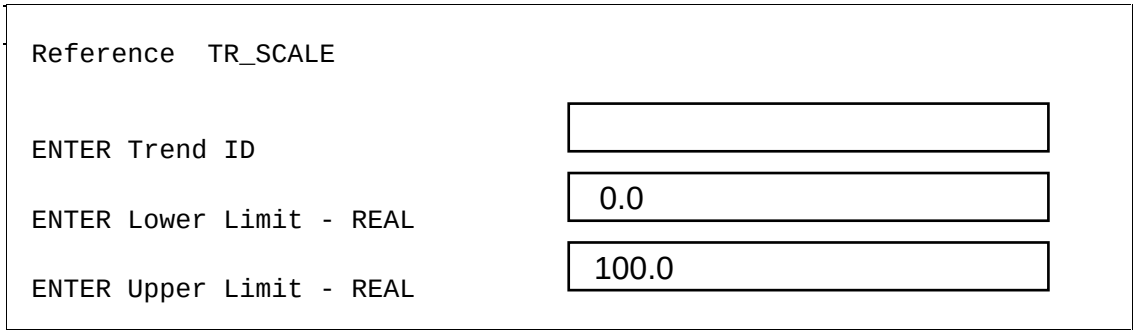
the high range value for trace 2 is derived from the parameter OPHILM.

13.10 SET SCALE FOR TREND

This actor is used to set the scale for the traces in a Trend Record Subpicture (either subpicture Trend or Trend_AX). By default, the scale is Lower Limit = 0% of the range and Upper Limit = 100% of the range for the particular trace. This window can be narrowed or widened as desired.

Calling Sequence— TR_SCALE(TREND ID,LOW LIMIT,HIGH LIMIT)

Optional Parameter Entry Form



Parameter Explanation

TREND ID is an integer (from 1-12) that designates the Trend Record.
LOW LIMIT is the percent modification for the lower range.
HIGH LIMIT is the percent modification for the upper range.

Example— Trend Record 1 is operating with default scale values. An operator selects a target with the action sequence TR_SCALE(1,25.0,75.0). The result is

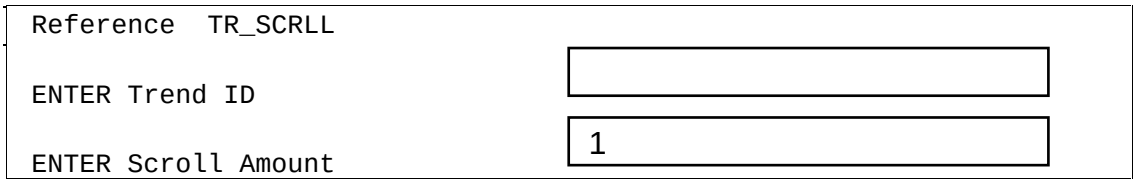
the vertical axis for Trend Record 1 is rescaled so that only the parts of the traces between 25.0% and 75.0% are displayed.

13.11 SET SCROLL FOR TREND

This actor causes scrolling through the specified trend subpicture (Trend or Trend_AX). Scroll Amount indicates how many scroll frames to shift (a scroll frame is equal to the currently selected time base). A positive value indicates a scroll back in time, a negative value indicates a scroll forward in time. Only traces with an HM data source are displayed when scrolling backward from the current time; however, data collection continues for non-HM-based traces. In R510 and later systems, if you attempt to scroll monthly, daily, shift, or hourly averages, scrolling is inhibited and the message Invalid Data Error results.

Calling Sequence— TR_SCRLL(TREND ID,SCROLL AMOUNT)

Optional Parameter Entry Form



Parameter Explanation

TREND ID is an integer (from 1-12) that designates the Trend Record.

SCROLL AMOUNT is a positive or negative integer that indicates the direction and magnitude of the scroll.

Example 1— Trend Record 1 contains four traces. All traces have the HM as a data source. The time base currently selected is eight hours. An operator wishing to see yesterdays values selects a target with the action sequence TR_SCRLL(1,3). The result is

the system requests data for three time bases earlier (i.e., the data start time is current time minus 24 hours).

The operator could return to current time by selecting a target that causes scrolling in the opposite direction (i.e., TR_SCRLL(1, -3). This is acceptable if the scroll amount is known; however, the actor TR_TIME(1, 8) could always be used to return to the current time with an 8-hour time base no matter how many times backward scrolling was used.

Example 2— Trend Record 1 contains four traces. Two traces have the HM as a data source; the other two have real time as a data source. The currently selected time base is two hours. At 10:30 an operator selects a target with the action sequence TR_SCRLL(1,1). The result is

- the traces based on real time data disappear.
- the system requests data for the two HM-based traces with a start time of one time base earlier (i.e., if the start time is currently 10:30, the system requests data with a start time of 8:30).

Next, the operator selects a target with the action sequence TR_SCRLL(1,-1). The result is

- the traces based on real time data reappear.
- the system requests data for the two HM based traces. The start time is current time.

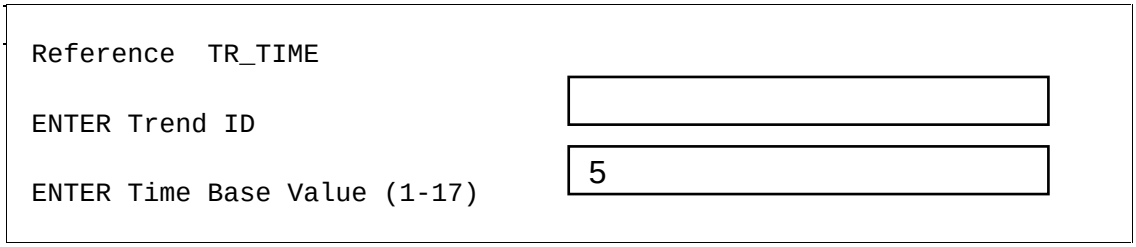
13.12 SET TIME BASE FOR TREND

This actor sets the time base for a Trend Record and requests data for the active traces. Data is not requested for a Trend Record that has no active traces.

If centerline trending is in use, the centerline time does not change, but the display is updated for the time base requested by the actor.

Calling Sequence— TR_TIME(TREND ID,TIMEBASE VALUE)

Optional Parameter Entry Form



Parameter Explanation

TREND ID is an integer (range = 1-17) that designates the Trend Record. If an out of range value is entered during build time, the message: Parameter Out Of Range results. If an out of range value is entered at run time (for example through an operator input actor), the error message: INVALID DATA appears:

TIMEBASE VALUE is

Value	Timebase	Value	Timebase
1	1 Minute	10	24 Hours
2	2 Minutes	11	48 Hours
3	5 Minutes	12	96 Hours
4	10 Minutes	13	Hourly Average
5	20 Minutes *	14	Shift Average
6	1 Hour	15	Daily Average
7	2 Hours	16	Monthly Average
8	8 Hours	17	User Average
9	16 Hours		

*Default value

Example 1— The operator selects a target with the action sequence TR_TIME(1,6). This actor sets the time base for Trend Record 1 to one hour and requests data for the active traces.

Example 2— Trend Record 1 contains four active traces. Two are based on HM data, and the other two are based on real-time data. The current time base is two hours and the traces have been scrolled. The operator selects a target with the action sequence TR_TIME(1,7). The result is

- the time base for Trend Record 1 was, and still is, 2 hours. The trend remains scrolled.
- data is requested for two traces based on HM data.

Note that the real time traces are not displayed when the trend is scrolled.

Example 3— Assume the same initial conditions as in example 2. If the operator selects a target with the action sequence TR_TIME(1,8), the result is

- the time base for Trend Record 1 is changed to 8 hours. The trend remains scrolled.
- the best data source for the four traces is determined.
- data is requested for all four traces.

Example 4— Assume the same initial conditions as in example 2. If the operator selects a target with the action sequence TR_TIME(1, 13), the result is

- the time base for Trend Record 1 is changed to hourly. The trend updates to the time of the last completed hourly average.
- The data source for all four traces is HM.
- Data is requested for all four traces.

13.13 TREND FUNCTIONALITY

As the next example illustrates, it is possible to add a trace without specifying every detail. Trend Functionality supplies missing parameters as follows:

Data Source— If the data source for a variable isn't specified, the system attempts to get data from various sources. For an Analog Input point, the current time base is considered as shown:

Data Source Order of Selection	10 Minutes Or Less Time Base	20 Minute Or Less Time Base	2 Hour Or More Time Base
1	Real Time	HG	HM
2		HM	HG
3		Real Time	Real Time

Note: If the history is not being collected on the point in either the Hiway box or HM, or if the current time base is 10 minutes or less, the only choice is real time.

For an HG digital point, a PM point, or an AM point, history can be accumulated in only the HM; therefore, the system attempts to get data from the HM first, but if history is not available on the point, the data source is set to real time.

Color— If not specified, the next available trend trace color is selected. See the table for TR_ADD.

Range— If range isn't specified, trend functionality chooses

Point Type	Range Selection
Analog	PVEUO - PVEUHI
Numeric	RANGELO - RANGEHI
all others	0 - 100

Time Base— If not specified, the default time base is set to 20 minutes.

Scale— If not specified, the default scale is 0.0 to 100% of the trace's value.

Trends in Schematics-Construction Example

The desired display is to contain two Trend Records; each Trend Record is to contain four traces. You want one target to activate both Trend Records.

The desired time base for Trend Record 1 is 20 minutes and the desired time base for Trend Record 2 is eight hours. You want to specify the trace colors in Trend Record 2.

The point.parameters are

Trend Record 1	Trend Record 2
HG1001.PV	AM101.PV
HG1002.PV	HG0501.PV
HG1003.PV	HG0501.MODE
HG1004.PV	HG0501.SP

The variables in Trend Record 1 are associated with hiway boxes that have trend memory; therefore, the data source is the HG. The data source for variables in Trend Record 2 is the HM (i.e., continuous history). You decide to let the system determine the data sources.

Build Procedure— After selecting the Picture Editor, add two subpictures (TREND or TREND_AX). The Trend Data Record for one is TREND01 and for the other it is TREND02. Two simulated trend pictures appear on the screen.

Next, add a box target with the action sequence

```
TR_ADD(1,HG1001.PV,0);TR_ADD(1,HG1002.PV,0);
TR_ADD(1,HG1003.PV,0);TR_ADD(1,HG1004.PV,0);
TR_TIME(2,8);
TR_ADD(2,AM101.PV,2);TR_ADD(2,HGO501.PV,6);
TR_ADD(2,HG0501.MODE,7);TR_ADD(2,HG0501.SP,3)
```

Explanation— When this target is activated, the first four actors will each add a trace to Trend Record 1. The system will determine the best data source for all traces (in this case, the HG). The (default) time base will be 20 minutes and the next available color is used for each trace (i.e., traces 1-4 are assigned cyan, yellow, red, and green, respectively).

The TR_TIME actor sets the time base for Trend Record 2 to eight hours. Because no traces are active in the Trend Record yet, no data is requested. This is the most efficient way, because if the traces were added first, a second data request would have to be issued for the 8-hour time base. The last four actors each add a trace to Trend Record 2. The system will determine the most appropriate data source, which in this case is continuous history.

Finally, you might want to label the target using an Add Text command. Figure 13-1 illustrates the completed drawing.

Continuation— occasionally, the operator needs to look at two other traces. You decide to add a target that will delete traces 1 and 3 from Trend Record 1 and add two others. The target action sequence is

```
TR_DEL(1,3);TR_DEL(1,1);TR_ADD(1,HG1005.PV;TR_ADD(1,HG1006.PV,0)
```

Comment— deleting traces in numerically decreasing order prevents confusion that may result when the system renumbers the remaining traces. Traces 2 and 4 become traces 1 and 2 respectively after the target executes. The two new traces become traces 3 and 4.

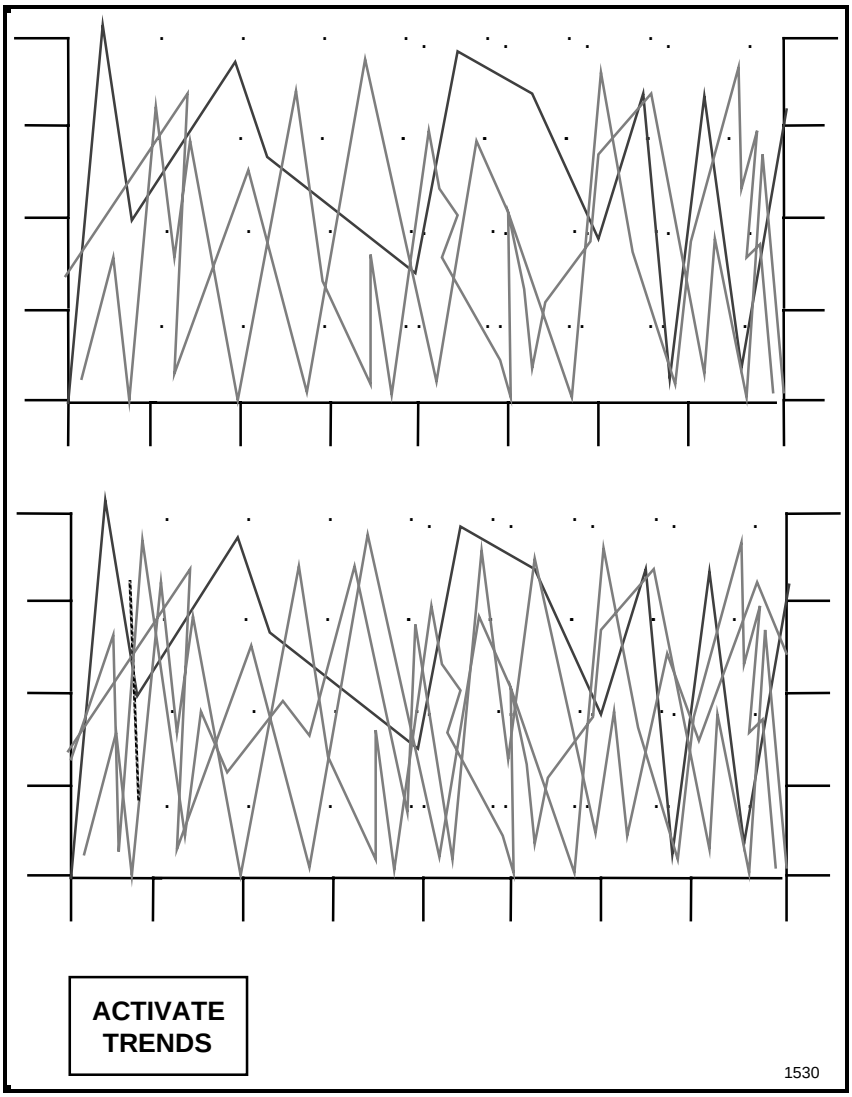


Figure 13-1 — Trends in Schematics Example

14 ARITHMETIC ACTORS

These actors allow arithmetic operations to be performed.

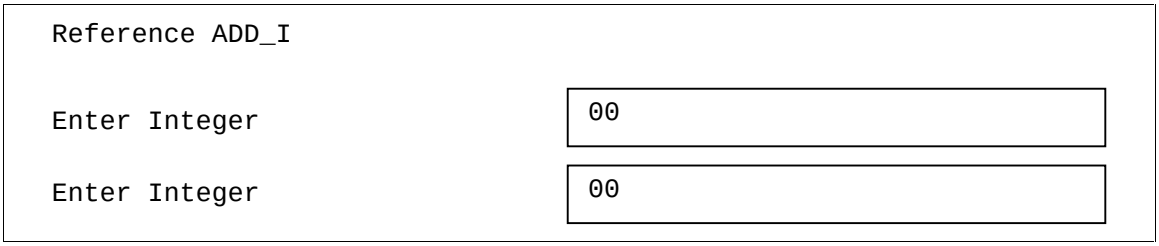
14.1 ADD INTEGERS

This actor adds integer 2 to integer 1. If arithmetic overflow occurs, the actor string is aborted and an error message is displayed.

Calling Sequence— ADD_I(Integer_1, Integer_2)

An integer result is returned.

Optional Parameter Entry Form



Example— S_INT(INT02,**ADD_I**(G_INT(INT02),1))

In this example the integer INT02 is incremented by 1.

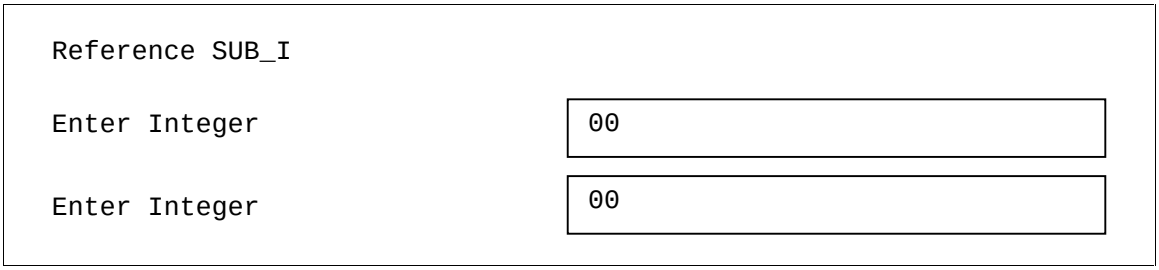
14.2 SUBTRACT INTEGERS

This actor subtracts integer 2 from integer 1. If arithmetic overflow occurs, the actor string is aborted and an error message is displayed.

Calling Sequence— SUB_I(Integer_1, Integer_2)

An integer result is returned.

Optional Parameter Entry Form



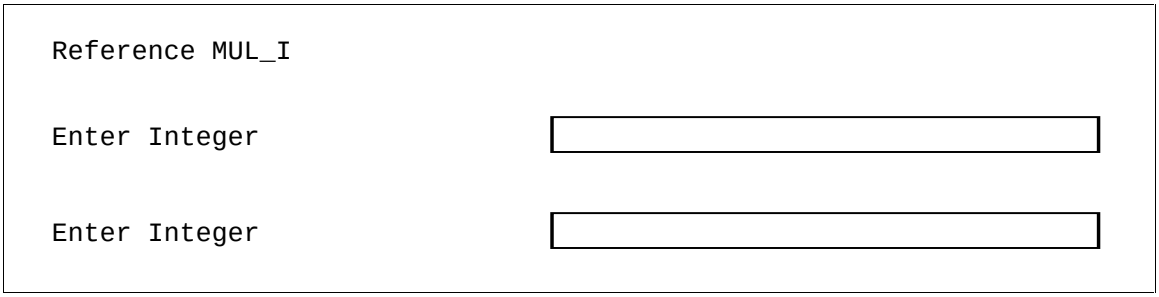
14.3 MULTIPLY INTEGERS

This actor multiplies integer 1 by integer 2. If arithmetic overflow occurs, the actor string is aborted and an error message is displayed.

Calling Sequence— MUL_I(Integer_1, Integer_2)

An integer result is returned.

Optional Parameter Entry Form



Example— SS_REAL(A100.OP,FLOAT(**MUL_I**(G_INT(INT01G),7)))

In this example, the MUL_I actor is used to obtain the product of INT01G and the integer 7. The rest of the action sequence converts that product to a real number and stores the result in A100.OP.

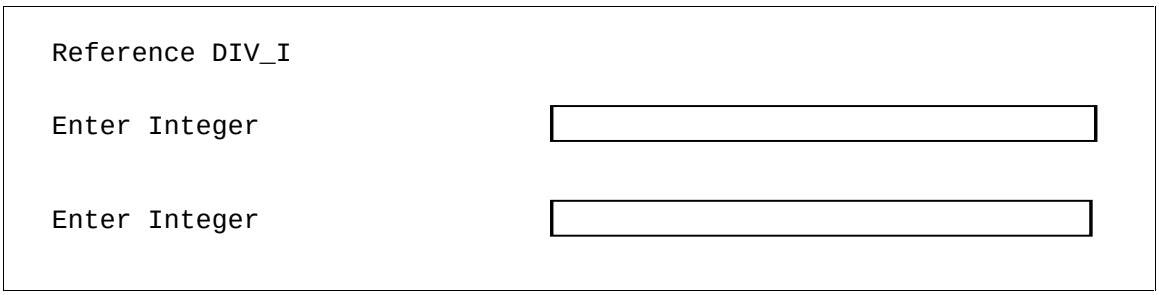
14.4 DIVIDE INTEGERS

This actor divides integer 1 by integer 2. If integer 2 is zero, the actor string is aborted and an error message is displayed.

Calling Sequence— DIV_I(Integer_1, Integer_2)

An integer quotient is returned.

Optional Parameter Entry Form



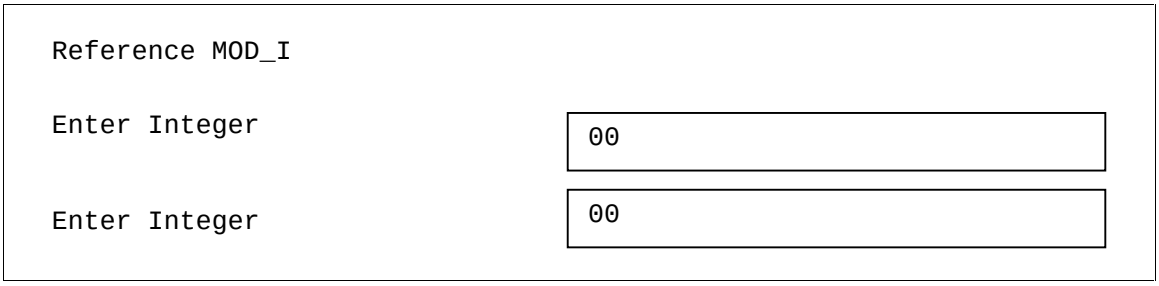
14.5 MODULO INTEGERS

This actor divides integer 1 by integer 2. If integer 2 is zero, the actor string is aborted and an error message is displayed.

Calling Sequence— MOD_I(Integer_1, Integer_2)

An integer remainder is returned.

Optional Parameter Entry Form



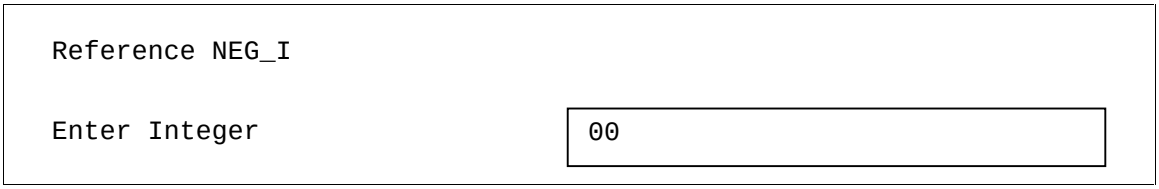
14.6 NEGATE INTEGER

This actor negates integer 1.

Calling Sequence— NEG_I(Integer_1)

An integer result is returned.

Optional Parameter Entry Form



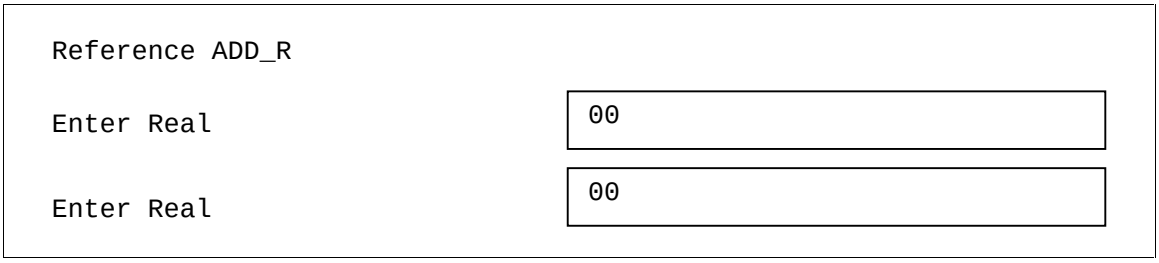
14.7 ADD REALS

This actor adds real number 2 to real number 1. Arithmetic overflow is not detected.

Calling Sequence— ADD_R(Real_1, Real_2)

A real result is returned.

Optional Parameter Entry Form



Example— SS_REAL(A100.PV,**ADD_R**(GS_REAL(A100.PV),1.0))

In this example, the real variable A100.PV, is incremented by 1.

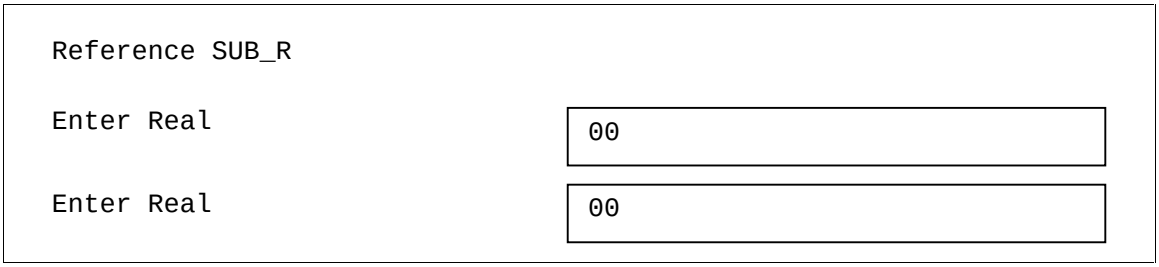
14.8 SUBTRACT REALS

This actor subtracts real number 2 from real number 1. Arithmetic overflow is not detected.

Calling Sequence— SUB_R(Real_1, Real_2)

A real result is returned.

Optional Parameter Entry Form



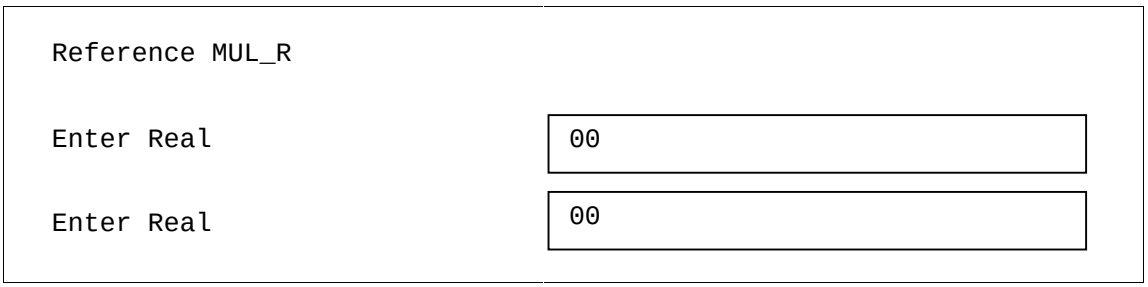
14.9 MULTIPLY REALS

This actor multiplies real number 1 by real number 2. Arithmetic overflow is not detected.

Calling Sequence— MUL_R(Real_1, Real_2)

A real result is returned.

Optional Parameter Entry Form



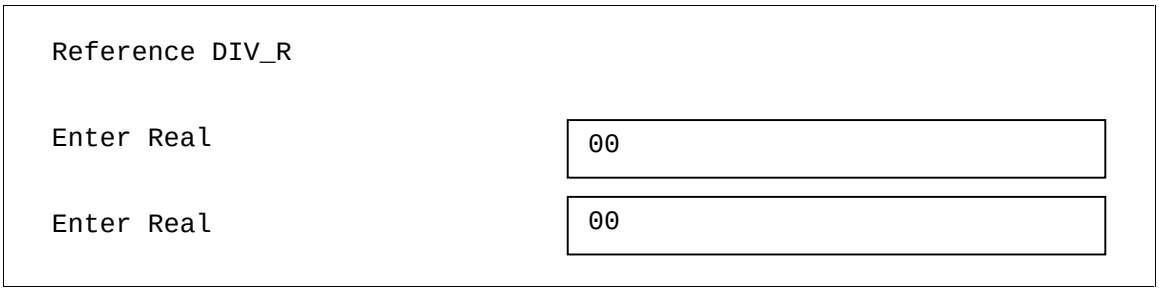
14.10 DIVIDE REALS

This actor divides real number 1 by real number 2. Arithmetic overflow is not detected. If real number 2 is zero, the actor string is aborted and an error message is displayed.

Calling Sequence— DIV_R(Real_1, Real_2)

A real result is returned.

Optional Parameter Entry Form



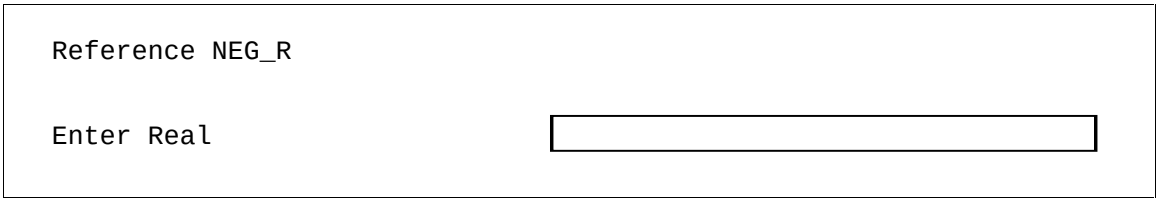
14.11 NEGATE REAL

This actor negates a real number.

Calling Sequence— NEG_R(Real_1)

A real result is returned.

Optional Parameter Entry Form



15 LOGICAL ACTORS

These actors allow logical operations to be performed.

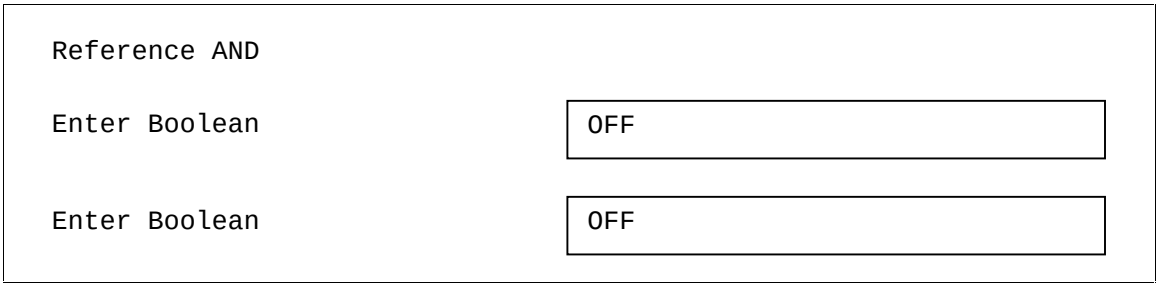
15.1 AND

This actor logically ANDs Boolean 1 with Boolean 2.

Calling Sequence— AND (Boolean_1,Boolean_2)

A Boolean result is returned.

Optional Parameter Entry Form



Example— S_BOOL(BOOL04,AND(G_BOOL(BOOL01),G_BOOL(BOOL03G)))

This example ANDs BOOL01 with BOOL03G and stores the result in BOOL04.

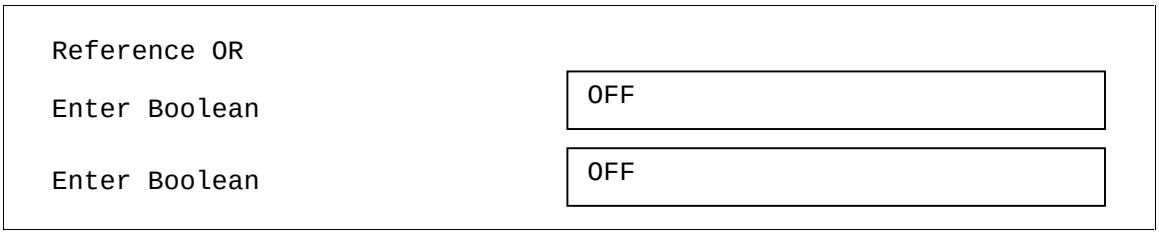
15.2 OR

Boolean 1 is logically ORed with Boolean 2

Calling Sequence— OR (Boolean_1,Boolean_2)

A Boolean result is returned.

Optional Parameter Entry Form



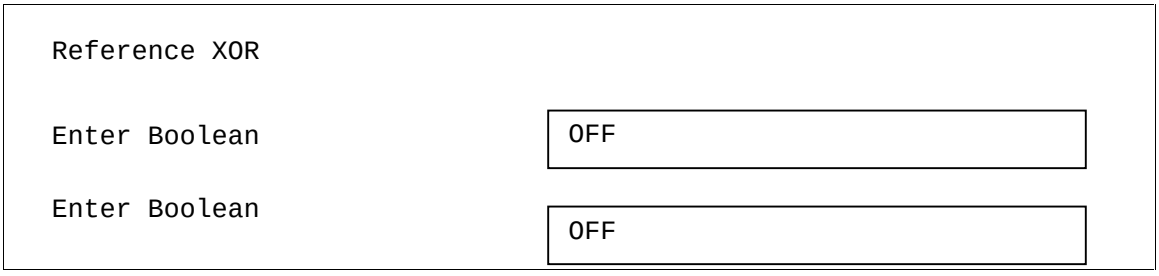
15.3 XOR (EXCLUSIVE OR)

Boolean 1 is logically exclusive ORed with Boolean 2.

Calling Sequence— XOR (Boolean_1,Boolean_2)

A Boolean result is returned.

Optional Parameter Entry Form



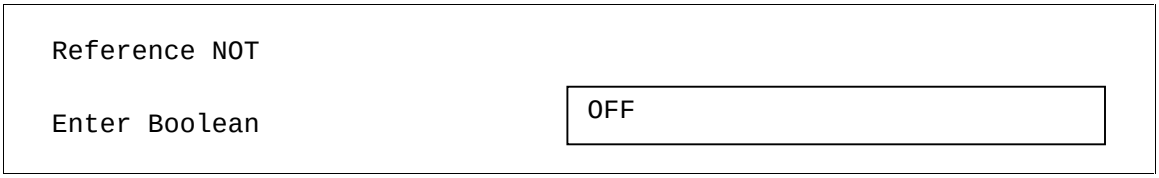
15.4 NOT

Boolean 1 is logically NOTed.

Calling Sequence— NOT(Boolean_1)

A Boolean result is returned.

Optional Parameter Entry Form



Example—SS_BOOL(CUST.BOOL,**NOT**(GS_BOOL(CUST.BOOL)))

In this example, the entity SS_BOOL toggles between true and false.

16 COMPARE ACTORS

These actors are used to perform comparative operations.

The notation ***op*** refers to one of the following comparison operators:

EQ	is equal to
NEQ	is not equal to
LT	is less than
GT	is greater than
LTEQ	is less than or equal to
GTEQ	is greater than or equal to

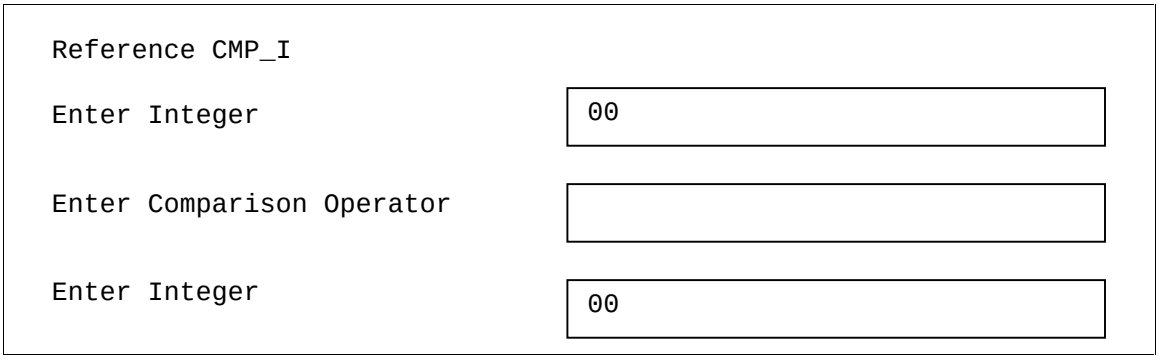
16.1 COMPARE INTEGERS

This actor compares Integer 1 with Integer 2 using the supplied operator.

A Boolean result is returned.

Calling Sequence— CMP_I(Integer_1,op,Integer_2)

Optional Parameter Entry Form



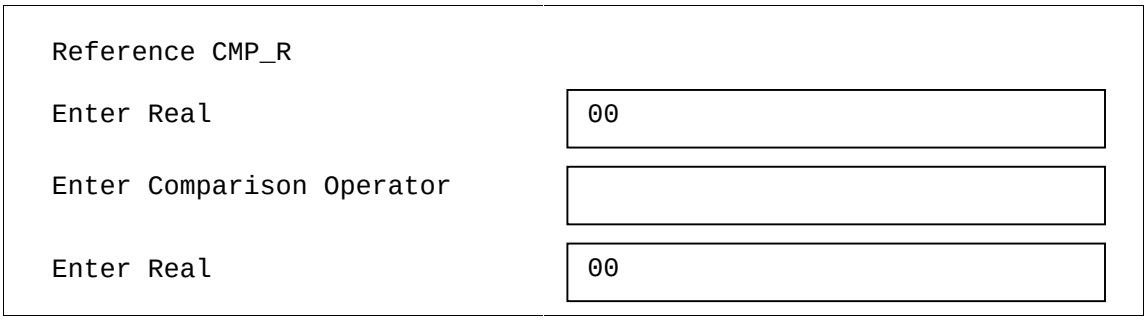
16.2 COMPARE REALS

This actor compares Real 1 with Real 2 using the supplied operator.

A Boolean result is returned.

Calling Sequence— CMP_R(Real_1,op,Real_2)

Optional Parameter Entry Form



Example— S_BOOL(BOOL01,**CMP_R**(GS_REAL(A100.PV),GT,G_REAL(REAL01)))

This action sequence determines whether A100.PV is greater than REAL01. The result is stored in BOOL01.

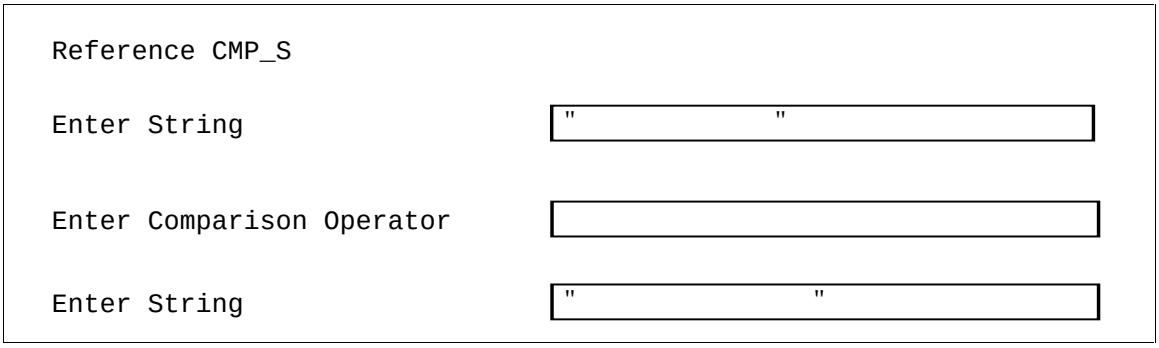
16.3 COMPARE STRINGS

This actor compares String 1 with String 2 using the supplied operator.

A Boolean result is returned.

Calling Sequence— CMP_S(String_1,op,String_2)

Optional Parameter Entry Form



This actor is also used to compare system variables of the type standard or self-defining enumeration when a *string* is returned by a retrieval actor. For example, in the action sequence IF(CMP_S(GS_ENM(A100.MODE),EQ,"AUTO")), the MODE enumeration is returned as a string and can be compared to the string "AUTO" by this actor.

16.4 COMPARE ENUMERATIONS

This actor compares Enumeration 1 with Enumeration 2, using the supplied operator. Enumeration refers to standard or custom enumerations, but not to self defining enumerations.

NOTE

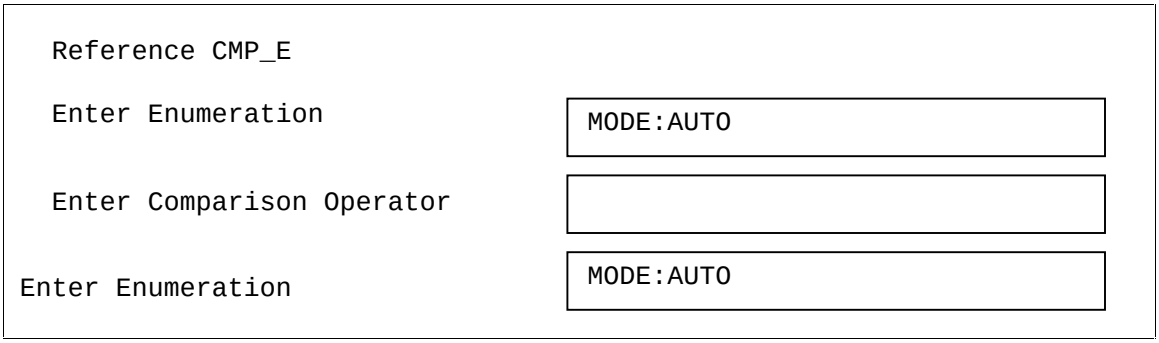
If the enumerations being compared are *strings*, as might be returned by a previously executed retrieval actor, use the **Compare Strings** actor described elsewhere in this section.

Valid comparison operators for this actor are EQ (is equal to) and NEQ (is not equal to). In order for the enumerations to test equal, both the enumeration set and the enumeration member must match.

Calling Sequence— CMP_E(Enumeration_1,op,Enumeration_2)

A Boolean result is returned.

Optional Parameter Entry Form



Examples— S_SENM(ENM01,MODE:MAN);S_BOOL01,**CMP_E**(G_SENM(ENM01),EQ,MODE:MAN))
S_BOOL(BOOL01,**CMP_E**(G_SENM(ENM01),EQ,MODE:MAN))

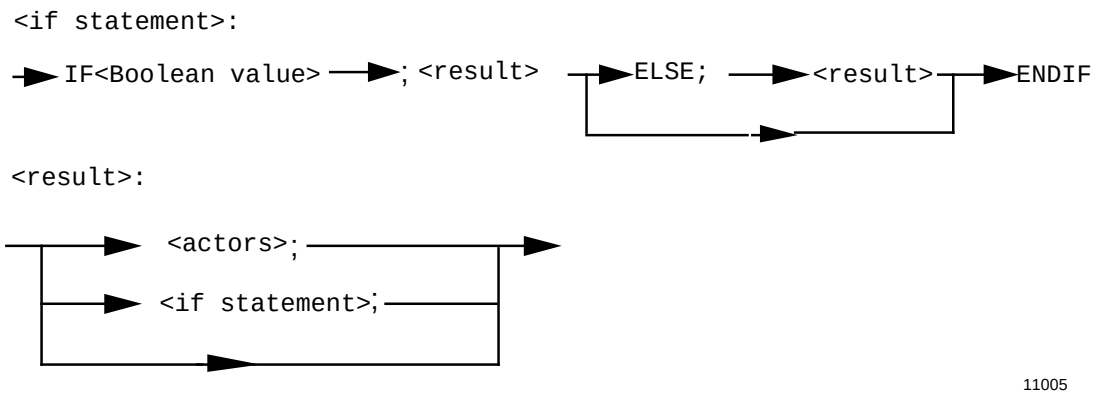
17 CONDITIONAL ACTORS

These actors are used to perform conditional operations. They consist of IF, ELSE, and ENDIF. You can use these actors to create IF statements within targets and buttons, or by combining these actors with the compare and arithmetic actors, you can generate complex conditional statements.

Conditional actors have the following characteristics:

- IF actors can be nested as required, but each IF actor must have a corresponding ENDIF actor.
- Extra ELSE actors, or extra or missing ENDIF actors, are not detected when built, but they are detected at operating time. When detected, an error message is displayed and the actor string terminates. Note that all actors preceding the error are executed if allowed to do so by the conditional context.
- Whether or not conditional actors are used, when an actor has an error, the actor string terminates at the point where the error occurred.

Syntax diagram:

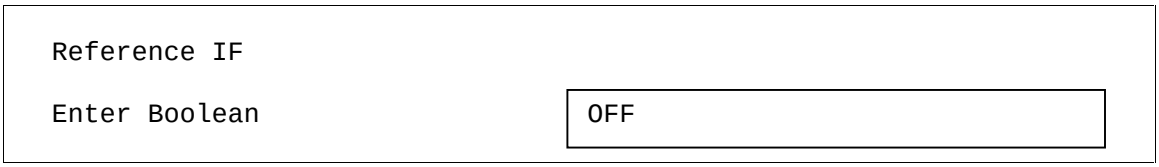


17.1 IF

This actor tests Boolean 1 and if true, execution continues. If false, execution ceases until a corresponding ELSE or ENDIF actor is encountered.

Calling Sequence— IF (Boolean_1)

Optional Parameter Entry Form



17.2 ELSE

When paired with an IF actor, ELSE causes actor execution to stop if actors were being executed. If actors were not being executed, ELSE allows actor execution to start. Note that the execution state is affected only if the state has been determined by the paired IF actor.

Calling Sequence— ELSE

Optional Parameter Entry Form—none

17.3 ENDIF

When paired with an IF actor, this actor allows execution state to continue only if the execution state had been determined by the paired IF actor.

Calling Sequence— ENDIF

Optional Parameter Entry Form— none

Examples

The following examples show how conditional actors can be used.

Example 1

```
IF(CMP_E(G_SENM($KEYLEVL),EQ,KEYLEVL:ENGR));
    RS_SYS(ENT01G.SP,0,0,15,"ENTER NEW VALUE",FALSE,0);
ELSE;
    PROMPT("Key Level Must Be ENG");
ENDIF;
```

This sequence allows the entity ENT01G.SP to be changed only if the Universal Station's key is in the ENGINEER position. If the key is in another position, the message **Key Level Must Be ENG** is displayed. Note that $KEYLEVL is the collector and KEYLEVL is an enumeration-set name.

Example 2

This example shows how conditional actors can be used to test and make changes based on the outcome.

In this example—

```
 A100 is an AM Regulatory point
CUST1 and CUST2 are custom parameters

 IF(CMP_R(GS_REAL(A100.PV),GT,50.0));
    SS_BOOL(A100.CUST1,TRUE);
    SS_REAL(A100.PV,SUB_R(GS_REAL(A100.PV),5.0));

    IF(CMP_R(GS_REAL(A100.SP),LTEQ,25.0));
      SS_BOOL(A100.CUST2,TRUE);
    ELSE;
      SS_BOOL(A100.CUST2,FALSE);
    ENDIF;
 ELSE;
    SS_BOOL(A100.CUST1,FALSE);

    IF(GS_BOOL(A100.CUST2));
      SS_REAL(A100.PV,ADD_R(GS_REAL(A100.PV),5.0));
    ELSE
      SS_REAL(A100.SP,SUB_R(GS_REAL(A100.SP),1.0));
    ENDIF;
 ENDIF;
```

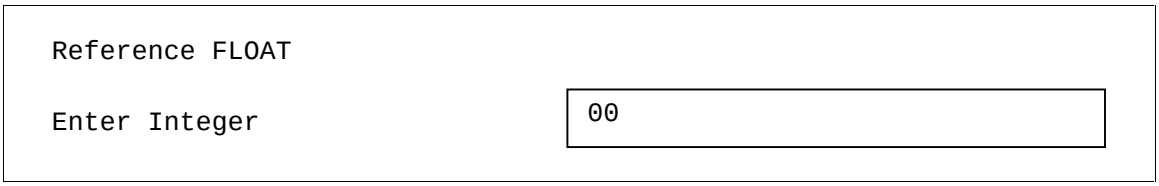
18 CONVERSION ACTORS

18.1 INTEGER TO REAL CONVERSION

This actor converts an integer to a real number and returns the result.

Calling Sequence— FLOAT (Integer_1)

Optional Parameter Entry Form



Example— SS_REAL(A100.OP,**FLOAT**(MUL_I(G_INT(INT01G),7)))

This action sequence converts the product of INT01G and the integer 7 to a real number and stores the result in A100.OP.

18.2 INTEGER TO STRING CONVERSION

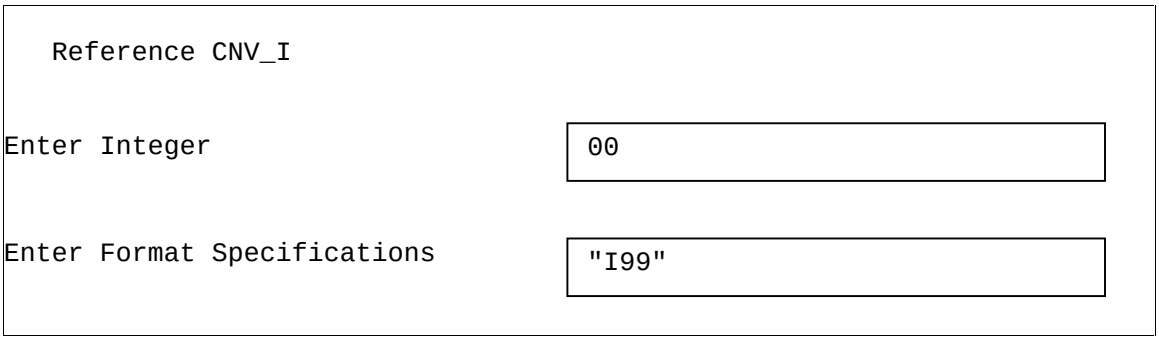
This actor converts an integer into a string using the specified format. An error results if the specified format is invalid or if the integer is too large to be converted using the specified format.

Calling Sequence— CNV_I(INTEGER, STRING1)

A string value is returned.

Example— S_STR(STRING01,**CNV_I**(G_INT,(INT01),"IZZZZ9")) the integer contained in INT01 is converted to a string and stored in STRING01.

Optional Parameter Entry Form



Parameter Explanation

INTEGER - The integer to be converted.

STRING1 - The format specification. This format is identical to the format used to output integers with the ADD VALUE command (see Appendix A). Note that the format string must be entered in upper case letters.

18.3 INTERNAL DISPLAY DATABASE ENTITY ID TO EXTERNAL ENTITY ID

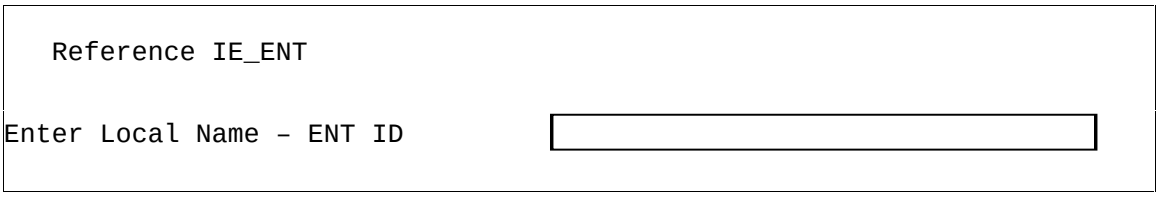
This actor takes a Local (Display Database) Entity name, like ENT01, and returns the external string-representation of the entity.

Calling Sequence— IE_ENT(DDB NAME)

This actor returns a String.

Example— S_STR(STRING01,**IE_ENT**(ENT04))

Optional Parameter Entry Form



18.4 INTERNAL DISPLAY DATABASE VARIABLE ID TO EXTERNAL ENTITY ID

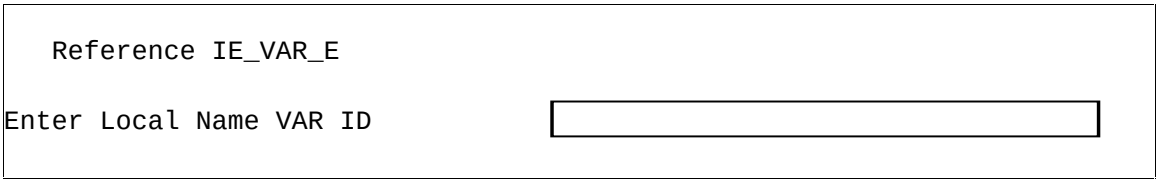
This actor takes a Local (Display Database) variable name, like VAR01, and returns the external string-representation of the entity.

Calling Sequence— IE_VAR_E(DDB NAME)

This actor returns a String.

Example— S_STR(STRING01,**IE_VAR_E**(VAR01))

Optional Parameter Entry Form



18.5 INTERNAL DISPLAY DATABASE VARIABLE ID TO EXTERNAL PARAMETER ID

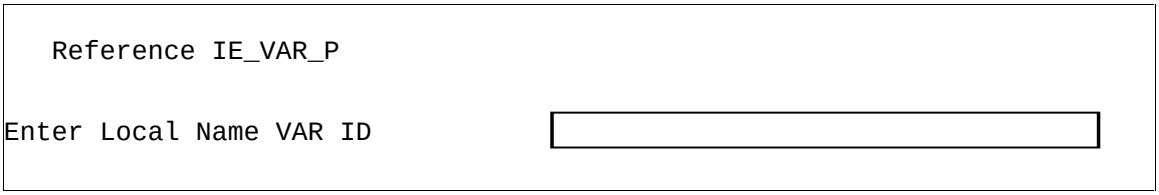
This actor takes a Local (Display Database) variable name, like VAR01, and returns the external string-representation for the parameter.

Calling Sequence— IE_VAR_P(DDB NAME)

This actor returns a String.

Example— S_STR(STRING01,**IE_VAR_P**(VAR01)

Optional Parameter Entry Form



18.6 INTERNAL DISPLAY DATABASE VARIABLE ID TO EXTERNAL VARIABLE ID

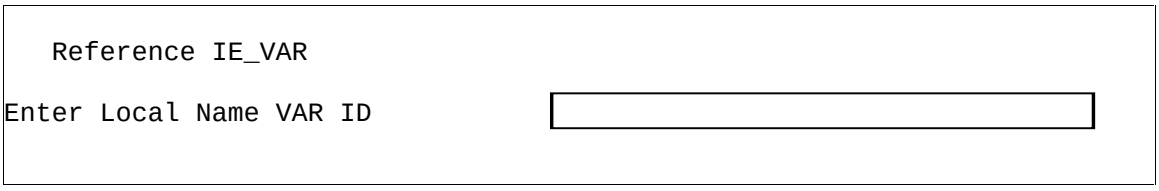
This actor takes a Local (Display Database) variable location, like VAR01, and returns the external string-representation of the variable ID such as A100.PV or A100(x).PV(x).

Calling Sequence— IE_VAR(DDB NAME)

This actor returns a String.

Example— S_STR(STRING01,**IE_VAR**(VAR01))

Optional Parameter Entry Form

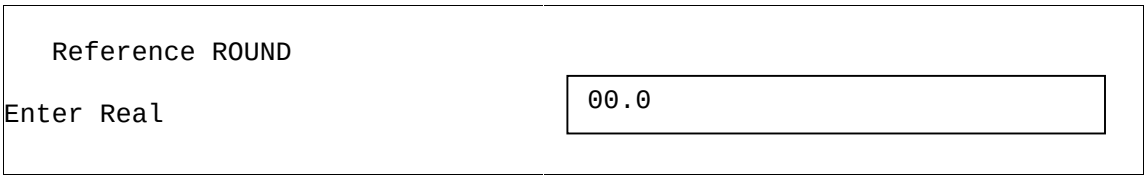


18.7 REAL TO INTEGER CONVERSION BY ROUNDING

This actor converts a real number to an integer by rounding and returns the result. If the rounded result is larger than an integer, the actor string is aborted and an error message is displayed.

Calling Sequence— ROUND(Real_1)

Optional Parameter Entry Form

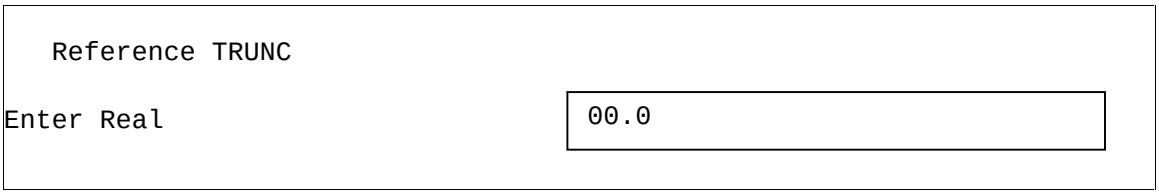


18.8 REAL TO INTEGER CONVERSION BY TRUNCATION

This actor converts a real number to an integer by truncation and returns the result. If the truncated result is larger than an integer, the actor string is aborted and an error message is displayed.

Calling Sequence— TRUNC(Real_1)

Optional Parameter Entry Form



18.9 STRING TO INTERNAL ENTITY ID CONVERSION

This actor converts a string into an internal entity Id. An error results if the input string is not the name of a valid system entity.

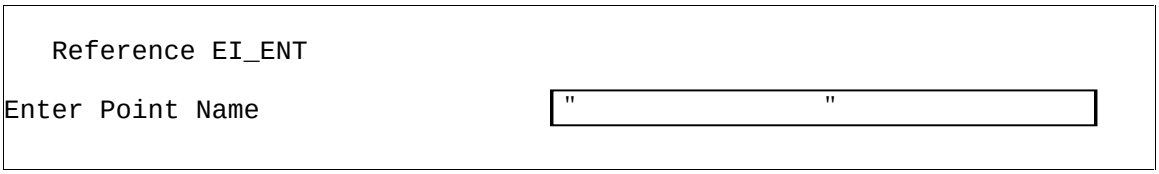
Calling Sequence— EI_ENT(STRING)

An entity Id is returned.

Example— S_ENT(**ENT01**,EI_ENT(CONCAT("$PRSTS",CNV_I,(35,"I99"))))

The entity Id, $PRSTS35, is generated and stored in ENT01.

Optional Parameter Entry Form



18.10 UNIT ID TO UNIT NUMBER

This actor converts a Unit ID to the area relative Unit Number (1..36) as configured in the Area Database (Unit Assignment).

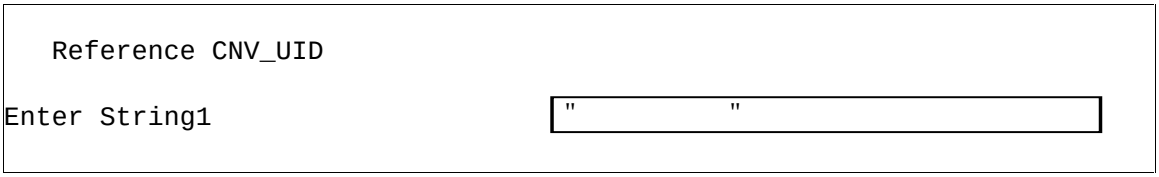
The area relative Unit Number is typically stored in a local DDB variable and then used as an index to fetch one of the PSDP alarm parameters ($PRSTS) listed in Section 22 of the *Engineer's Reference Manual*.

Calling Sequence— CNV_UID("STRING1")

An integer value is returned. If the specified Unit ID does not exist, the error message: INVALID IDENTIFIER is displayed.

Example 1—S_INT(INT01,**CNV_UID**("U1"))
Example 2 S_INT(INT01,**CNV_UID(R_STR(0,0,10,** "Enter The Configured Unit ID", TRUE,0)))

Optional Parameter Entry Form



Parameter Explanation

STRING1- A two character string specifying one of the valid Unit IDs as configured in the Unit Names section of the Network Configuration File.

19 QUERY ACTORS

With Release 510 and later software, these actors allow an operator to invoke Documentation Tool queries from a custom-built schematic and display the query results.

Query definitions must be built by using the Universal Personality's Documentation Tool function. Then one or more custom schematics can use the actors in this section to invoke and display, print, or store the result of the query. Other actors permit manipulation of the query functions and files.

Only one Invoke Query actor can be active at a time from the same Universal Station. Additional query requests from the same Universal Station are rejected and the following error message appears:

Another Query In Progress

Simultaneous query requests from different Universal Stations are allowed but may adversely affect performance.

While a query invoked from a schematic is still in progress, you cannot use the engineering functions at that US. An attempt causes the following error message: Denied - Query in progress - Query must complete or be cancelled.

NOTE

These actors cannot be used to invoke prebuilt queries with historized journal access.

The Documentation Tool query collectors described in subsection H.9 of the *Picture Editor Reference* Manual are typically used with these actors. Also, refer to the PSDP parameter QSTS(n) in the *Engineer's Reference Manual*.

Refer to the *Documentation Tool* Manual, for more information on Documentation Tool functions.

19.1 INVOKE QUERY WITH DISPLAY OUTPUT

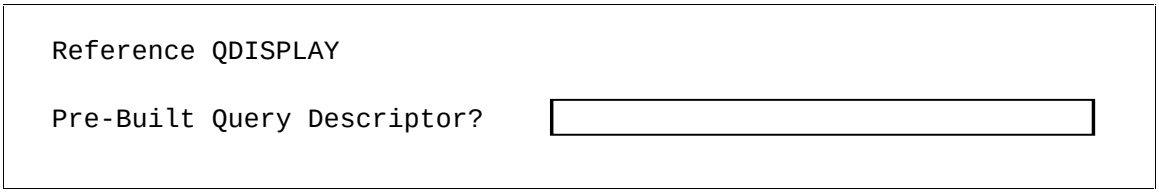
This actor invokes the Documentation Tool Query function to execute a specified pre-built query and display the query results at the requesting Universal Station.

The query is cancelled if you exit the schematic before the query function is completed.

Calling Sequence— QDISPLAY("Query Descriptor")

where Query Descriptor is the descriptor name for the rebuilt query.

Optional Parameter Entry Form



Example— QDISPLAY("All entities on UCN 3")

The Query Descriptor is checked at run time. If invalid, the following message appears on the top line of the screen:

Pre-built Query Invalid

NOTE

This actor is not valid for use with the configurable buttons.

19.2 INVOKE QUERY WITH FILE OUTPUT

This actor invokes the Documentation Tool Query function to execute a specified rebuilt query, send the results to a specified file, and (optionally) display the query results at the requesting Universal Station.

If the specified output file already exists, it is overwritten.

Calling Sequence— QFILE("Query Descriptor", "Pathname", Display Option)

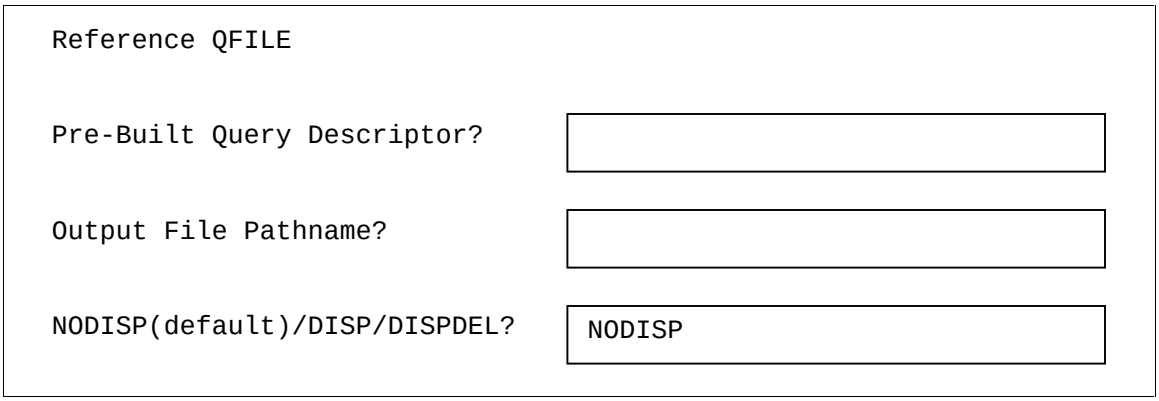
where:

- Query Descriptor is the descriptor of a rebuilt query.
- Pathname is the pathname to a file where the results are output.
- Display Option is one of the following:

  NODISP - do not display results
  DISP - display results
  DISPDEL - display results; delete results file after viewing

  If you leave the entry blank, the Display Option defaults to NODISP.

Optional Parameter Entry Form



Query results are displayed on the US if the calling schematic remains on screen until the query function completes. If the display option was DISPDEL, the query results file is deleted when you exit the query results display.

If the Display Option was DISP or DISPDEL and you exit the calling schematic before the query function is completed, the query status value indicates that the query completed but could not be displayed. In addition—

- if the display option was DISP—

  The query continues to completion (unless the query is canceled by the Cancel Query actor). Query results are written to the specified output file.

- if the display option was DISPDEL—

  The query continues to completion. Query results are written to the specified output file but they are not deleted in this case.

The query results file can be displayed by using the QRESULTS or DSP_FILE actors.
The query results file can be printed by using the PRT_FILE actor.

NOTE

Display options DISP or DISPDEL are not valid for use with a configurable button. The Button Configurator will accept them but at operating time, an error message results indicating that the query completed but could not be displayed.

This actor is not supported by the Cross_Screen actor.

The Query Descriptor and Pathname are checked at run time. If the Query Descriptor is invalid, the following message appears on the top line of the screen:

Pre-built Query Invalid

If the Pathname is invalid, an appropriate error message appears on the top line of the screen.

Example— QFILE("All entities on UCN 3", "Net>HMV1>Temp1.XX", DISP)

This actor invokes the rebuilt query named "All entities on UCN 3" and stores the results to the file Temp1 in the HM volume named HMV1. It also displays the results on the US.

For additional information, see the Pre-Defined Documentation Tool Query sections in the *Command Processor Operation* manual and in the *Engineer's Reference Manual*.

19.3 INVOKE QUERY WITH PRINTER OUTPUT

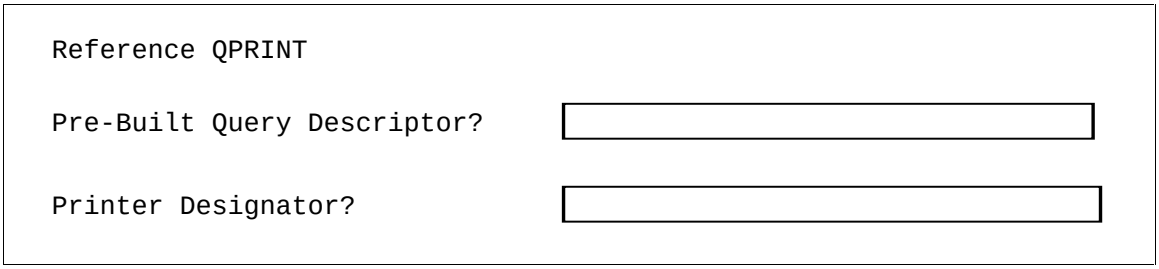
This actor invokes the Documentation Tool Query function to execute a specified rebuilt query and send the output to a specified printer

Calling Sequence— QPRINT("Query Descriptor", "Printer Designation")

where:

- Query Descriptor is the descriptor name of a rebuilt query.
- Printer Designation is the name of the printer.

Optional Parameter Entry Form



Example— QPRINT("All entities on UCN 3", "$P3")

The Query Descriptor and printer designation are checked at run time. If the Query Descriptor is invalid, the following message appears on the top line of the screen:

Pre-built Query Invalid

If the printer designation is invalid, the following message appears on the top line of the screen:

Invalid Printer Designation

Query results are displayed on the US if the calling schematic remains onscreen until the query function completes. If you exit the calling schematic before the query results are displayed, the query continues to completion (unless canceled by the Cancel Query actor). Query results are sent to the specified printer when the query is complete.

NOTE

This actor is not supported by the Cross_Screen actor.

For additional information, see the Pre-Defined Documentation Tool Query sections in the *Command Processor Operation* manual and in the *Engineer's Reference Manual*.

19.4 CANCEL QUERY

This actor cancels a currently active Documentation Tool Query that was invoked from a schematic by one of the Invoke Query actors.

The actor can be configured as a Final Target to automatically cancel a query when exiting the calling schematic (refer to the Picture Editor Define command). You can also configure a button to cancel the currently running query.

If the results data is being written to a file or printed when a Cancel Query actor executes, the following message results: Query cancelled - Partial data exists. The remaining results data is ignored.

Calling Sequence— QCANCEL

> **NOTE**
>
> This actor is not supported by the Cross_Screen actor.

For additional information see the Pre-Defined Documentation Tool Query sections in the *Command Processor Operation* manual, and in the *Engineer's Reference Manual*.

19.5 CLEAR QUERY STATUS

This actor blanks out the query status values (name, status, date/time invoked) in the Query collectors. The purpose is to clear out unwanted residual status data.

Calling Sequence— QCLEAR

> **NOTE**
>
> This actor is not supported by the Cross_Screen actor.

19.6 DELETE QUERY RESULTS FILE

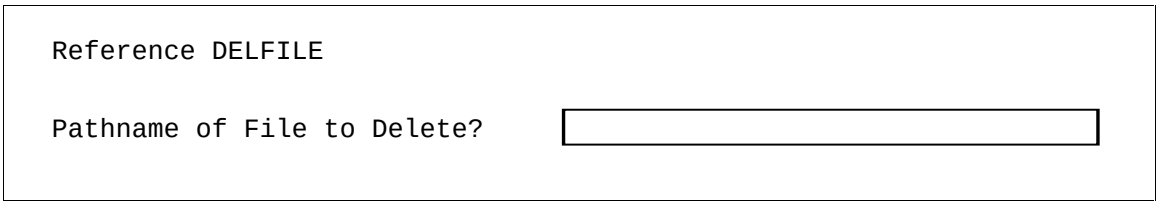
This actor deletes a specified Documentation Tool Query Results file. The purpose is to avoid confusing previous data with current data.

Calling Sequence— DELFILE("Pathname")

where:

Pathname is the pathname of the Query Results file to be deleted.

Optional Parameter Entry Form



Example— DELFILE("NET>HMV1>TEMP.XX")

The Pathname is checked at run time. If it is invalid, an error message appears on the top line of the screen.

19.7 DISPLAY QUERY RESULTS

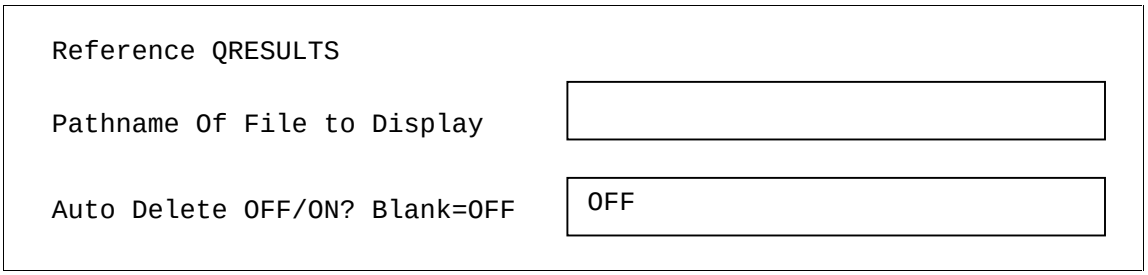
This actor causes the Results File from the last Documentation Tool Query to be displayed on this Universal Station. The Query Results File can optionally be deleted after the display is removed from the screen.

Calling Sequence— QRESULTS(Pathname, Auto-Delete Mode)

where:

- Pathname is the pathname to the Query Results file.
- Auto-Delete Mode is one of the following:
  OFF - The Results File is left unchanged after viewing the data
  ON - The Results File is automatically deleted by exiting the results display.
  Blank = OFF

Optional Parameter Entry Form



Example— QRESULTS("NET>HMV1>TEMP.XX", OFF)

The pathname is checked at run time. If it is invalid, an error message appears on the top line of the screen.

APPENDIX A DISPLAY DATABASE VARIABLES

Standard and user-defined Display Databases (DDB) provide variables that are used only for display purposes. Many of the examples in this appendix show how DDB variables are used. Actors used with configurable buttons cannot reference these databases. Variants used in Free Format Logs cannot reference these databases.

A.1 STANDARD LOCAL DDB VARIABLES

Whenever a Custom Graphic Display is called up, a predefined set of standard Local DDB Variables are created. These variables are used to hold values for that display only. When the display is cancelled, the Local DDB Variables terminate. The standard Local Display Database Variables currently available are listed below.

REAL—You can reference up to 20 Real Display Database (DDB) variables. These are named REAL01 through REAL20.

INTEGER—You can reference up to 20 Integer Display Database variables. These are INT01 through INT20.

ENUMERATIONS— You can reference up to 10 Enumeration DDB variables. These are named ENM01 through ENM10. They contain SET:MEMBER enumerations. Enumerations are handled as strings. You can use the actors SS_ENM and GS_ENM (defined elsewhere) to store and retrieve a **Point.Parameter** enumerated variable.

For Enumerated types that are not point related (like Unit_ID), the actors S_SENM and G_SENM should be used.

Actor RS_LOC (Read and Store Local General) must be used only for standard enumerations, not self-defining enumerations.

BOOLEAN—You can reference up to 20 Boolean Display Database variables. These are named BOOL01 through BOOL20.

STRING—You can reference up to 20 String Display Database variables. These are named STRING01 through STRING20.

ENTITY—You can reference up to 20 Entity Display Database variables. These are named ENT01 through ENT20. These variables can contain entity names and can be used in indirection in displays built with the Picture Editor.

VARIABLE—You can reference up to 20 Variable Display Database variables. These are named VAR01 through VAR20. These variables contain a combination of an entity name and parameter, and can be used in indirection in displays built with the Picture Editor.

DATE TIME—You can reference up to four Date/Time Display Database variables. These are named DATIME1 through DATIME4.

The date and time are entered differently while building the display than at run time. When building the display, the date format is separated by dashes (-), and the time format is separated by a colon (:). At run time, the date and the time formats are separated by blanks.

Internal date and time are stored as a 48-bit binary time value. A time representation consists of six bytes. The first four bytes contain the seconds elapsed since January 1, 1979 and the last two contain the tenth of milliseconds in the second.

Because one variable holds both the date and time, the type of this variable is just that, date or time. When storing or reading a DATIME variable, the type can be either a Date or Time.

Note that if the actor RS_LOC is used to store a date time DDB variable, the type is defaulted to Date. If you want the type to be time, use actor R_TIME in conjunction with SYS_TIME.

TREND ID—You can reference up to 12 Trend-Record Display Database variables. These are named TREND01 through TREND12. They are used by the Trend Actors and Application Subpictures TREND/TREND_AX as explained in Section 13 of this manual.

A.1.1 Initial State of Standard Local DDB Variables

A standard local Display Database variable exists from the time a display that uses it is called up until the display is canceled. The initial state of the standard local Display Database variables is shown in Table A-1.

Table A-1 – Initial State of Local Display Database Variables

Variable	Startup State	Initialized Value
INTO1–INT20	initialized	0
REAL01–REAL20	initialized	0.0
BOOL01–BOOL20	initialized	FALSE
STRING01–STRING20	initialized	BLANKS
ENT01–ENT20	initialized	@@@@@@@@ (NULL)
ENM01–ENM10	initialized	* * * * * * * *
VAR01–VAR20	initialized	@@@@@@@@
DATIME1–DATIME4	initialized	Current date/time (i.e., time The display is called up)
TREND01–TREND12*	initialized	No trend traces

* Use DDB variables TREND01-TREND12 only with Application Subpictures TREND or TREND_AX.

A.2 STANDARD GLOBAL DISPLAY DATABASE VARIABLES

A Standard Global Display Database variable exists from the time the station starts up until the station is reset. These variables are used to retain values that can be used on all the Custom Graphic Displays. The same actors that store to the Local Display Database can be used to store to the Global Display Database. Actors used with targets and actors used with the Button Configurator, can access the Global Display Database.

The initial state of the Standard Global Display Database variables is shown below.

Table A-2 — Initial State of Standard Global Display Database Variables

Variable	Startup State	Initialized Value
INT01G–INT20G	initialized	0
REAL01G–REAL20G	initialized	0.0
BOOL01G–BOOL20G	initialized	FALSE
STR01G–STR20G	initialized	BLANKS
ENT01G–ENT20G	initialized	@@@@@@@@ (NULL)
ENM01G–ENM10G	initialized	* * * * * * * *
VAR01G–VAR20G	initialized	@@@@@@@@
DATIME1G–DATIME4G	initialized	The date and time this Universal Station was loaded

A.3 USER-DEFINED DDB VARIABLES

Unlike the standard Display Data Base that contains variable names defined by Honeywell, user-defined Display Data Base variables can be defined by the individual display builders. User-defined DDB variables are declared in a text file for use with the Picture Editor's Load command. The same variable types that are used in the standard Display Data Base are available for user-defined DDBs.

The user DDB declare file contains a declaration of the DDB variable names and must be created before executing the Load command. You can create the file with the Text File Editor. The procedure is—

- Enter the Command Processor by selecting it from the Engineering Main Menu, or by pressing the ESC key while using the Picture Editor.
- Enter **EDIT nnnn.df**
  where **nnnn** is the pathname of the file. The file extension **df** is required.
- Enter the statements. The general format is **name: type, index, disposition;**
  For example—

  BOOL50: Boolean, 500, Local;

Figure A-1 shows an example of a DDB file. The file is created with the Text Editor. The *Text Editor Operation* manual explains how to use the Text Editor.

```
{                                   }
{ This is a sample DDB File         }

{ Note that blank lines are legal  }

SEL_ENT : ENTITY , 200,  G ;        { spaces are allowed  }
SEL_ENT2: ENT,200,L;                {  but not necessary  }

my_int: int, 100, local;            { Lower case is okay  }

REAL01: R,  500,  L;                { This will not be recognized }
                {  because it is a duplicate of }
                {  a standard name               }

BOOL50: Boolean, 500, L;            { This is okay }
BOOL50: Boolean, 250, L;            { This will not be recognized }
                {  because it is a duplicate name }

XYZZY : STRING,  400, G;            { This is okay }
PLOVER: STRING,  400, G;            { This is an alias of XYZZY }

My_Date:Date_Time, 700, L;    My_Int3:Integer, 5, GLOBAL;

{  This is okay  }    My_Enum :  ENUM:MODE,  450,  LOCAL;

{   All of the above are syntactically correct. }
{   All of the following have a syntax error.    }

_Int: INTEGER, 180, G;              { Cannot start with underbar }
10Real: REAL,  190,  Glob;          { Cannot start with numeric }
STR__5: S,  230,  G;                { Consecutive underbars are illegal }
INTEGER20 : INTEGER, 260, L;        { Name is longer than 8 characters }
My_Var_1 : VAR,  330,  G            { Missing semicolon }
My_Ent :  ENTITY,  Loc;             { Missing index }
My_Bool: LOGICAL, {  } 400, G;      { Comments within a statement }
My_Real: REAL,  50,   G;            { Index out of range   }
```

Figure A-1 — Example DDB file

A.3.1 DDB Statement Syntax

The syntax for statements in a user created DDB file is—

VARIABLE NAME: **VARIABLE TYPE**, **INDEX**, **DISPOSITION**;

For example:

DCFI: INTEGER, 100, GLOBAL;

Variable Name

The variable name is—

- one to eight ASCII characters
- Legal characters are A-Z/a-z, 0-9, and underbar. The first character must be A-Z/a-z
- The underbar cannot be the first or last character and there cannot be two consecutive underbars

Variable Type

The variable type is one of the following:

VariableType	Synonyms		
INTEGER	INT	I	
NUMBER	REAL	NUM	R
LOGICAL	BOOLEAN	L	B
STRING	S		
ENUM:<enum_type>	E		
DATE_TIME	DATE/TIME	DATE	D
ENTITY_ID	ENTITY	ENT	
VARIABLE	VAR	V	

Index

- An integer from 100 to 32767
- A maximum of 500 indexes per user created DDB file

Note that user-defined global DDB variables are not initialized and there is a memory limit of 1000 words. Table A-3 shows the user-defined global DDB variables.

D**isposition**

Disposition is one of the following:

Word	Synonyms
GLOBAL	GLOB G
LOCAL	LOC L

Table A-3— User-defined Global Display Database Variables, Memory Word Size

Data Type	Word Size (Characters)	
Entity	9	
Real	7	
Integer	6	
Boolean	6	
Enumeration	7	
Variable	15	
String	n + 6	(n = the number of characters in the string)
	2	

User-defined DDB variables are not initialized; therefore, reading from these variables before writing to them returns a BAD VALUE status.

For example—

Standard DDB variable, INT01, initial value = 0
User-defined DDBvariable, MY_INT, initial value = @@@@@@@@@

A.3.2 Rules

Note the following rules for user-defined DDBs. You may also want to refer to the syntax diagrams and the example DDB file while reading these rules.

- Name duplication within a DDB file is not detected; the first name encountered takes precedence.
- Duplication of names in the DDB file and standard DDB names is not detected. The standard name takes precedence.
- Index duplication within a data type and a disposition are not detected. This causes the names to be aliases for the same location.
- Only indices from 100 to 32767 are valid.
- The maximum number of DDB Variable names per file is 500. Exceeding this number of DDB Variable names causes an error when the file is loaded.

- More than one statement can appear on a line.
- Spaces can appear anywhere within a statement except within identifiers.
- A statement must begin and end on the same line.
- There is a memory limit of 1000 words for user-defined global DDB variables. The size of each DDB variable is shown in Table A-3. If the memory limit is exceeded, the message DDB ERROR appears at runtime.

A.3.3 Comments

DDB files can also contain comments. The following rules apply to comments—

- Comments can appear on the same line with the DDB declaration statement, before the statement or after the statement, or both, but not within the statement.
- A comment must begin and end on the same line.
- Comments must be enclosed in braces { } (see the comment syntax diagram).

Graphically, the DDB matrix can be visualized as shown in Figure A-2.

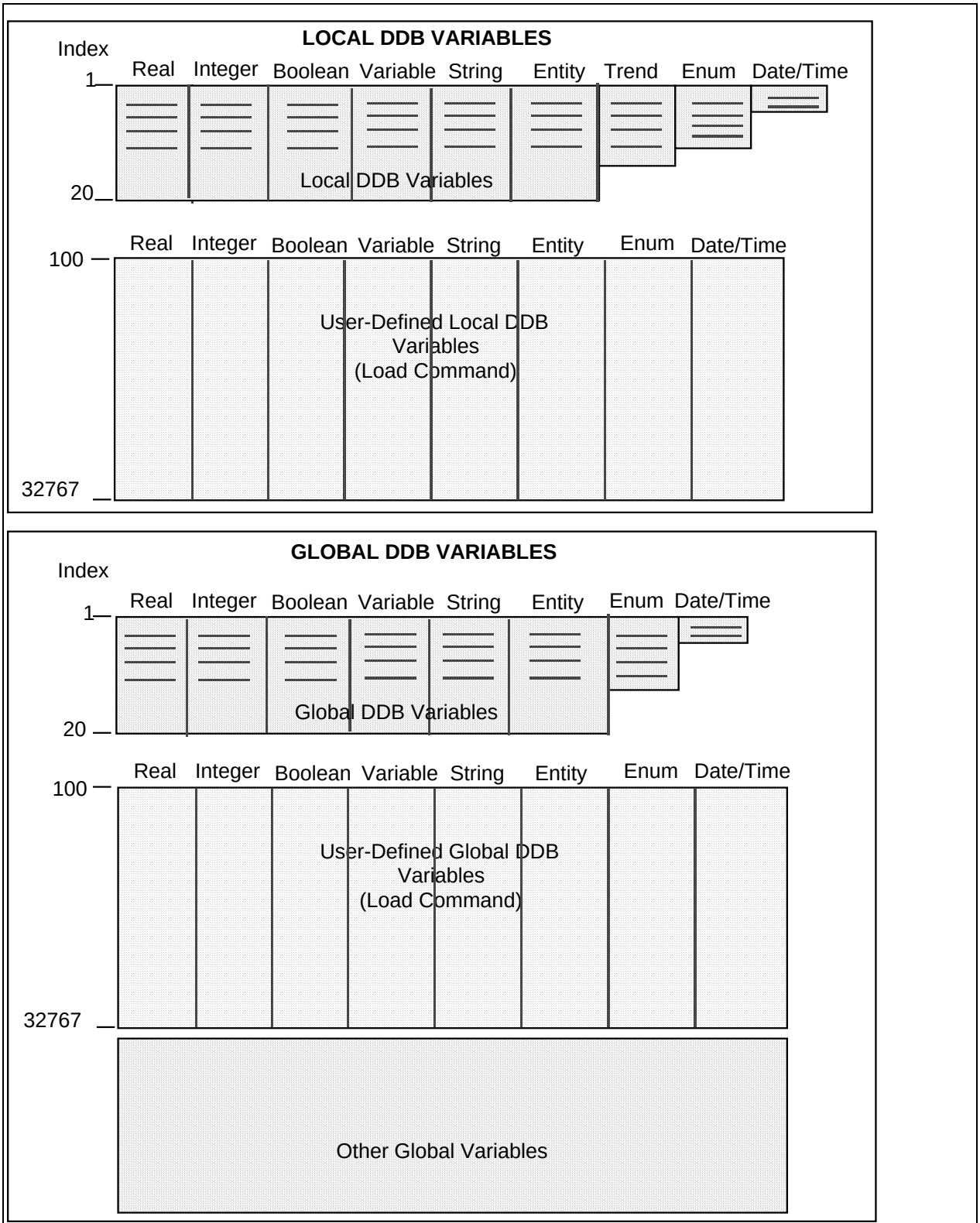


Figure A-2 — Standard and User-defined Display Database Variables 11290

A.4 DISPLAY DATABASE–INDIRECTION

The following examples show how to indirectly specify a **point.parameter**:

Example 1

First use the actor S_ENT to store an Entity ID in the Display Database memory such as:

S_ENT(ENT02,TIC100)

As a result an Actor that requires a variable ID like TIC100.PV will accept:

ENT02.PV or ENT02.NAME

Example 2

First use the actor S_VAR to store a Variable ID in the Display Database memory such as:

S_VAR(VAR01,TIC100.PV)

As a result an actor that requires a variable ID like TIC100.PV will accept VAR01 instead**.**

A.5 SPECIAL DISPLAY DATABASE VARIABLES

Variables in the following table are accessible for read-only use.

Table A-4— Special System Display Database Variables

Variable	Description
$AL_ENTY	An entity variable that provides the name of a point that is selected on the Alarm Summary, Unit Alarm Summary, Alarm Annunciation or Organizational Summary displays. Whenever a point is selected on one of these displays, the point name in $AL_ENTY is overwritten with the newly selected point name.
$ALMCOLR	An integer that specifies the alarm priority color option selected in the NCF: 0 = two color option (red, yellow), 1 = 3 color option (user selected colors).
$ARAID	An 8-character string name for the current area.
$ARAID01 – $ARAID10	An 8-character string name for each indicated area as configured in the NCF.
$ARADSC	A 24-character string description of the current area.
$ARADS01 – $ARADS10	A 24-character string description for each indicated area as configured in the NCF.
$CONDSC	A 24-character string description of the current console.
$CONDS01–$CONDS10	A 24-character string description of the indicated console as configured in the NCF.

Table A-4 continued on next page

Table A-4 — Special Display Database Variables , continued

$CONNUM	An integer that represents the current console number.
$CZ_ACT	A Boolean. True if the Change Zone is Active, false if the Change Zone is inactive.
$CZ_ENTY	An entity ID that contains the point name that is provided as a parameter to the CHG_ZONE actor or to the USER_CZ actor.
$DT_ENTY	This global DDB contains the name of the last point for which a Detail Display was invoked. It is initialized to NULL_ID.
$EALMCLR	An integer that specifies the Emergency Alarm priority color option selected in the NCF (see Table A-5).
$GRPBASE	An entity ID that can be used to access the group definitions.* (see Notes to Table A-4)
$HALMCLR	An integer that specifies the High Alarm priority color option selected in the NCF (see Table A-5).
$KEYLEVL	An enumeration that contains the internal keyswitch level for this US. The values are: VIEW, OPR, SUP, and ENGR.
$LALMCLR	An integer that specifies the Low Alarm priority color option selected in the NCF (see Table A-5).
$MY_AREA	An integer that represents the area number on which the custom display is running.
$MY_PNA	An integer that represents the node number on which the custom display is running.
$REDYEL	An enumeration that contains the minimum alarm priority indicated by the color red. It contains the members: LOW, HIGH, and EMERGNCY
$STNNUM	An integer that represents the station number on which the custom display is running.

Notes to Table A-4

*GRPBASE is used to access group definitions in the following manner:

$GRPBASE(group #).TITLE—returns a 24-character string that contains the indicated group title.

$GRPBASE(group #).EXTIDLST(slot #)—returns an 8-character string that contains the indicated point name.

DDB entity IDs cannot be indexed as used in the example above, but the name can be built by using the Combine Variable ID actor (C_VAR). You must also specify the internal entity ID of the point to be configured on the group slot. For example, if you want to configure the second slot of group 391 to contain the point A100, you can configure a target or button with the following:

SS_ENT(C_VAR(G_ENT($GRPBASE),391,.INTIDLST,2), EI_ENT("A100"))

or

SS_STR(C_VAR(G_ENT($GRPBASE),391,.EXTIDLST,2),"A100")

Note that INTIDLST and EXTIDLST are system parameters for the group points.

- INTIDLST stores the internal (data access) representation of the point name.
- EXTIDLST stores the ASCII name printed when the group is invoked.
- KEYWORD stores the ASCII keyword printed when the group is invoked. Include the actors that fetch the point's real keyword value from the data owning node and save this to a local string parameter. Use the local string parameter in the actor that stores to the group keyword value. This point.parameter was added in R530.

Table A-5—Color Values for **$EALMCLR**, **$HALMCLR**, **$LALMCLR**

Color	Integer
Red	1
Green	2
Yellow	3
Blue	4
Magenta	5
Cyan	6
White	7

A new R520 boolean Node Global DDB is shown in the table below. This value is read/write and is different from the existing read only values named in Table A-4.

Table A-6— Special System Display Database Variables

Variable	Description
$STRT_UP	This DDB can be referenced and/or updated in a custom schematic. Its primary purpose is to allow you to determine if a Universal Station has been restarted. This boolean value is set to false when the node is loaded and started up. Schematics may interrogate and reset this value to do custom startup initialization.

A.6 HONEYWELL DEVELOPERS GLOBAL DISPLAY DATABASE

A separate Display Database exists for use by Honeywell Developers of Custom Graphic Displays. The information in this section is provided to inform both customers and developers of its existence and general structure; however, **customers must not use the Honeywell Developers DDB**.

The Honeywell Developers DDB is used with the Load command in the same way and with the same syntax rules as explained for the User-defined Display Database. There is a memory limit of 10,000 words for Honeywell Developer Globals. In order to differentiate between the User-defined DDB and the Honeywell Developer's DDB, a special directive is required.

A.6.1 Avoiding Conflicts

To avoid failure of critical functions or, at the very least, to avoid causing unpredictable results, it is extremely important that schematics developed at any Honeywell site do not conflict with those developed at other sites. To avoid any possible conflict, different indices for DDB Variables have been assigned.

A.6.2 Assignments

DDB Variables used in custom schematics that are developed at the various Honeywell sites must not conflict. The following table shows how indices are assigned for custom global, and standard and custom local DDB variables. Within each site, a DDB Administrator must be appointed to assure that there is no conflict within the limits assigned to that site.

Index Range	Assigned To	Comments
-32767 to 99	Standard DDB Variables (administered by Phoenix Engineering).	The Load Command does not allow declaration of variables with indices in the range -32767 to 99.
100 to 1999	Yamatake Honeywell Developers	
2000 to 3999	Honeywell Phoenix Custom Systems Developers	
4000 to 5999	Honeywell Canada Developers	
6000 to 7999	Honeywell Pace Developers	
8000 to 9999	Honeywell Fort Washington Developers	
10000 to 11999	Honeywell Icotron Developers	
12000 to 13999	Honeywell/third party projects and joint ventures	
14000 to 15999	Honeywell Toolkit Developers	
16000 to 17999	Honeywell Customer Enhancement Engineering	
18000 to 18999	Honeywell Germany	
19000 to 32767	Reserved, **DO NOT USE**	

Graphically, the Honeywell Developers DDB matrix can be visualized as shown in Figure A-3.

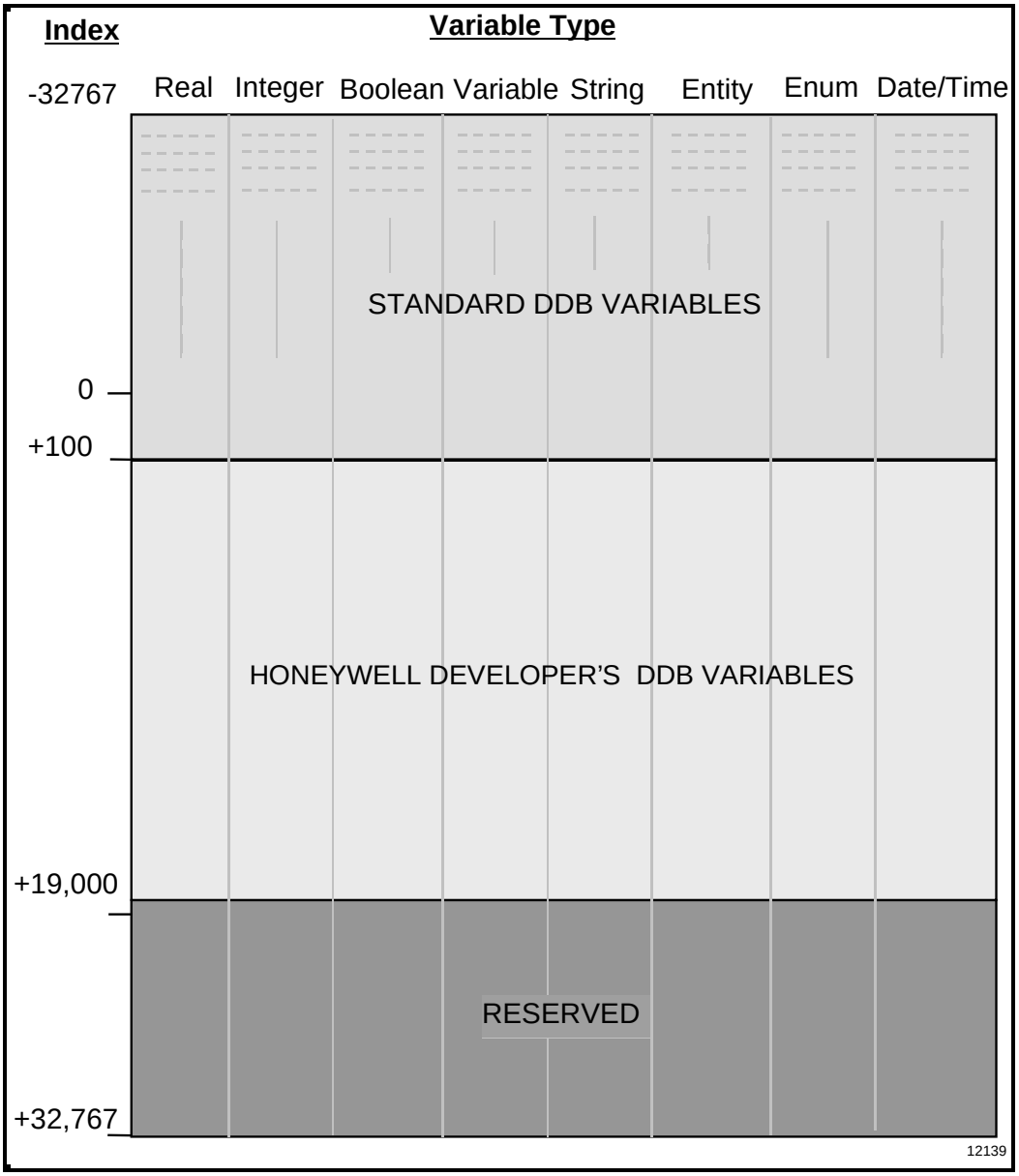


Figure A-3 — Honeywell Developer's Global Display Database Variables

For complete information on the Honeywell Developer's Display Database, contact:

Manager, Software Systems Engineering
Honeywell IAC
HED Address AZ15-2E8
16404 N. Black Canyon Highway
Phoenix, AZ 85023

APPENDIX B CUSTOM CHANGE ZONE CONSTRUCTION

This supplement provides an advanced example of how actors can be used. Before attempting this construction, you should be totally familiar with subsection 3.3.10 (Subpictures) in the Picture Editor Reference Manual, and Section 1 in this Actors Manual.

B.1 CUSTOM CHANGE ZONE EXAMPLE

Problem—Create a set of subpictures so that an engineer can construct displays that contain custom-change zones. The change zones must allow an operator to observe and change the Setpoint (SP), Process Variable (PV) Output (OP) and Mode (Mode) parameters for any point that has these parameters.

Operation of the Change Zone is to be as follows:

1. The Operator selects a point by touching a target on a display.
2. The Change Zone is presented at the bottom of the display. The Change Zone shows the name of the point, the description of the point and the values for each of the above mentioned parameters.
3. To change any of the numeric parameters (e.g., SP, PV, or OP), the operator uses a touch target to select the parameter. After selection, he/she types the new value in and, when the "Enter" key is pressed, the parameter value is changed.

For the Mode Parameter, the operator selects the new value by first touching one of three targets. Each of these targets represents a value of the Mode parameter (i.e., Manual, Auto, or Cascade). Following selection of the value, an EXECUTE target appears. When this target is touched, the Selected value is stored into the Mode parameter.

4. The Change Zone is to contain a target that will allow the operator to call up the Detail display for the selected point.
5. Once a point has been selected, and the Change Zone for that point displayed, the Change Zone is to remain on the screen. If a subsequent point is selected, the Change Zone is updated to reflect the new point.
6. The Change Zone is to contain a target that will allow the operator to remove it from display.

Solution—The solution is described in three parts.

- The first part (Table B-1) describes the Display Database variables used.
- The second part describes construction of the subpictures, including the actors used.
- The third part describes construction of a display that uses the subpictures.

B.1.1 Part 1—Display Database Variables Used

Table B-1 — Display Database Variables

Variable Name	Type	Use
ENT01	ENTITY	Contains the Name of the selected entity
VAR01	VARIABLE	Contains the names of the selected entity and the SP parameter
VAR02	VARIABLE	Contains the names of the selected entity and the PV parameter
VAR03	VARIABLE	Contains the names of the selected entity and the OP parameter
VAR04	VARIABLE	Contains the names of the selected entity and the MODE parameter.
BOOL01	BOOLEAN	Controls the visibility of the Change Zone
STRING01	STRING	Contains the external name of the value for the Mode Parameter (i.e., "Man", "Auto", "Cas")

B.1.2 Part 2—Subpictures

Four subpictures are used in the solution: SELECT, CZ, CZ_IMAGE, and EXEC. Each is described in the following paragraphs.

B.1.2.1 Subpicture SELECT

This subpicture is used to select an entity for the change zone. The subpicture consists of a single invisible target. Action procedures for the target contain parameters and are as follows:

S_ENT(ENT01,&P1);S_VAR(VAR01,&P1.SP);S_VAR(VAR02,&P1.PV);

S_VAR(VAR03,&P1.OP);S_VAR(VAR04,&P1.MODE);

S_STR(STRING01," ");S_BOOL(BOOL01,TRUE);UPDATE(0,0)

The function of each actor is discussed below:

S_ENT(ENT01,&P1) - This actor stores the name of the selected point into the Display Database variable "ENTO1" (&P1 represents a parameter).

S_VAR(VAR01,&P1.SP) - This actor combines the name of the selected point and the parameter name "SP" and stores the result in Display Database variable "VAR01."

S_VAR(VAR02,&P1.PV) - This actor combines the name of the selected point and the parameter name "PV" and stores the result in Display Database variable "VAR02."

S_VAR(VAR03,&P1.OP) - This actor combines the name of the selected point and the parameter name "OP" and stores the result in Display Database variable "VAR03."

S_VAR(VAR04,&P1.MODE) - This actor combines the name of the selected point and the parameter name "MODE" and stores the result in Display Database variable "VAR04."

S_STR(STRING01," ") - This actor stores a blank string in the Display Database variable STRING01. This variable is used to retain the selected value for the Mode parameter of the selected point. Blanks indicate no parameter value is selected.

S_BOOL(BOOL01,TRUE) - This actor sets the Boolean location "BOOL1" to true. This Boolean is used to drive a variant in the Change Zone subpicture. The variant controls the visibility of the change zone (see "Subpicture CZ_IMAGE," below).

UPDATE(0,0) - This actor requests a demand update of the display. This makes the Change Zone subpicture appear.

B.1.2.2 Subpicture CZ

This subpicture implements the custom change zone for entities that have SP, PV, OP, and MODE parameters.

The subpicture consists of a single Variant. This variant controls the visibility of the Change Zone image.

The Variant statement is as follows:

```
IF BOOL01
THEN SUBPICTURE CZ_IMAGE
```

Note that the statement IF BOOL01 is equivalent to IF BOOL01 = TRUE (reference the *Picture Editor Reference Manual*, Appendix E).

B.1.2.3 Subpicture CZ_Image

This subpicture implements the image and actions of the Change Zone. Its construction is discussed below in two sections. First the Picture Editor commands used to build the image are discussed, then the targets within the subpicture are described.

Figure B-1 shows the image of the subpicture for the example entity: 03PC003.

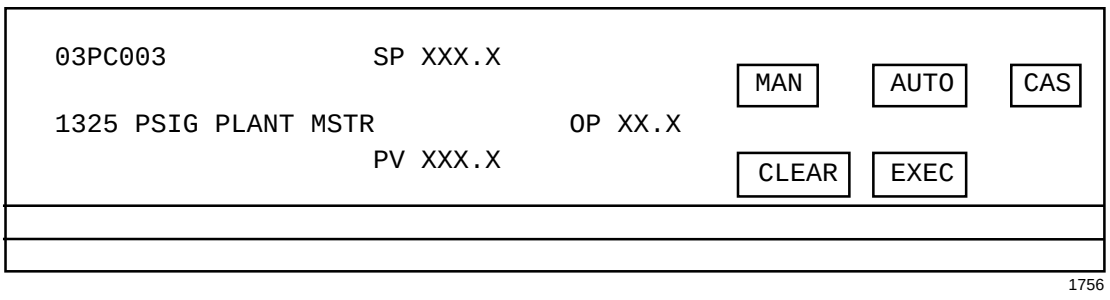


Figure B-1 — Custom Change Zone

B.1.2.3.1 Picture Editor Commands for Image

This section describes the Picture Editor commands used to create the image for the CZ subpicture.

1. The name of the selected point appears in the upper-left corner of the subpicture. It is added using the Picture Editor's Add Value command. The expression used to retrieve the value is "ENT01.NAME."

2. The description of the point appears below the name. It is added by using the Picture Editor command Add Value. The expression used to retrieve the value is "ENT01.PTDESC."

3. The TEXT objects are SP, PV, OP, MAN, AUTO, CAS and CLEAR. These are added with the ADD TEXT command.

4. The values of the SP, PV, and OP parameters are added with the Add Value command. The expressions used to retrieve the values are

   VAR01. (for the SP); ENT01.SP is an alternate form.

   VAR02. (for the PV); ENT01.PV is an alternate form.

   VAR03. (for the OP); ENT01.OP is an alternate form.

5. The text objects MAN, AUTO, and CAS are under conditional behavior. This allows the operator to see the current value of the Mode parameter. For example, if the Mode of the selected point is Man, then the MAN text will be white and blinking. The CAS and AUTO text will be Green, Non-Blink.

   Conditional Behavior for the MAN text is

   IF VAR04. = MAN
   Then Set White Blink
   Else Set Green

   For the AUTO text

   IF VAR04. = AUTO
   Then Set White Blink
   Else Set Green

   For the CAS text

   IF VAR04. = CAS
   Then Set White Blink
   Else Set Green

6. The boxes around the MAN, AUTO and CAS text are targets (either solid or hollow). These targets have conditional behavior that show the current operator selection. The conditions are

   For the MAN Target

   IF STRING01 = "MAN"
   Then Set Red Full
   Else Set Red Half

   For the AUTO Target

   IF STRING01 = "AUTO"
   Then Set Red Full
   Else Set RED Half

   For the CAS Target

   IF STRING01 ="CAS"
   Then Set Red Full
   Else Set Red Half

   Thus, when the operator selects a particular value for the Mode parameter, the target box for that parameter appears in full-intensity red; otherwise, it appears in half-intensity red.

7. The box around the CLEAR text is a target (either solid or a box).

8. The EXECUTE box is a variant with the Variant body statement

   IF STRING01<>" "
   Then Subpicture EXEC

   The subpicture EXEC contains a single target and is described later. If a value for the Mode parameter is selected, the EXECute target is visible, otherwise it is not.

B.1.2.3.2 Targets

The targets in subpicture CZ_Image are described below.

1. An invisible target is added such that it covers the selected point's name and descriptor (upper-left corner). This target allows the operator to call the detail display for the selected point.

   The action procedure for this target is

   ```
   DETAIL(IE_ENT(ENT01))
   ```

   When this target is touched, the actor "IE_ENT" executes and retrieves the selected Entity ID from the location ENT01. This value is returned and passed to the DETAIL actor. The DETAIL actor then calls up the Detail display for the selected point.

2. An invisible target is added such that it covers the SP (setpoint) text and value. When this target is touched, a Text Input Port is opened next to the SP value. This allows the operator to enter the new setpoint value in the Text Input Port. When the operator presses the enter key, the value is read and stored in the SP of the selected point. The action procedure for this target is

   ```
   RS_SYS(VAR01,10,2,6,"ENTER SETPOINT",False,0);Update(0,0)
   ```

   The Update actor is to assure that the change is seen immediately on the change zone.

3. Invisible targets are added such that they cover the PV and OP text and values. The respective action procedures are

   ```
   RS_SYS(VAR02,10,2,6,"ENTER PV",False,0);Update(0,0)

   RS_SYS(VAR03,10,2,6,"ENTER OP",False,0);Update(0,0)
   ```

4. The Manual target allows the operator to select the value of MAN for the Mode parameter of the selected point. This target does not store the value to the point, but saves the value as a Display Data Base variable. The action sequence for this target is

   ```
   S_STR(STRING01,"MAN");UPDATE(0,0)
   ```

   This action sequence first stores the selected value of MAN for the Mode parameter of the selected point. It then updates the screen. The update assures that the EXEC target becomes visible when any Mode value has been selected, and that the MAN, AUTO, and CAS targets reflect the current selection.

5. The AUTO and CAS targets allow the operator to select the value of AUTO or CAS as the value for Mode parameter of the selected point. This target does not store the value in the point, but saves the value to be stored in the Display Data Base variable STRING01.

   The action sequences are

   S_STR(STRING01,"AUTO");UPDATE(0,0)

   S_STR(STRING01,"CAS");UPDATE(0,0)

   These action sequences store the selected value of AUTO or CAS for the Mode parameter of the selected point. It then updates the screen. The update assures that the EXEC target becomes visible when any Mode value has been selected, and that the MAN, AUTO, and CAS targets reflect the current selection.

6. The CLEAR target allows the operator to clear the change zone. The action sequence for this target is

   S_STR(STRING01," ");S_BOOL(BOOL01,FALSE);UPDATE(0,0)

   This sequence clears the value of the Mode parameter (so that on the next invocation the EXEC target is not visible) and then sets BOOL01 to FALSE to make the Change Zone disappear.

B.1.2.4 Subpicture EXEC

This subpicture consists of a single target and the TEXT EXEC. This target's action sequence is

SS_ENM(G VAR(VAR04),G_STR(STRING01));

S_STR(STRING01," ");UPDATE(0,0)

When this target is touched, the actor G_VAR retrieves the value of VAR04 (i.e., the names of the selected point and the mode parameter). Thus, if the selected entity was 03PC003, VAR04 would contain 03PC003.MODE. Actor G_STR retrieves the selected value for the Mode parameter, thus, if MAN had been selected, STRING01 would contain MAN.

These values are then input to the SS_ENM actor, which stores the selected value in the Mode parameter of the selected point.

The S_STR actor sets the value for the MODE parameter to blanks, and the UPDATE actor causes the display to update. This results in the EXEC target disappearing, and causes the MAN, AUTO and CAS targets to reflect the current selection.

B.1.3 Part 3—Constructing the Display

This section covers the construction of a display that uses the subpictures just described.

1. For each point you want to be able to operate using the custom change zone, add the SELECT subpicture. The parameter to this subpicture (&P1) is the name of the entity to be selected.

2. You can add the subpicture CZ in a location of your choice on the display.

   Building the display is much easier than constructing the subpictures. This suggests two steps (or more) in display construction. The first step consists of creating the general subpictures. The second step consists of using the general subpictures as building blocks to create displays.

APPENDIX C KEY/BUTTON IDENTIFIERS

Keyboard keys and buttons have the following IDs for use with the Cross Keyboard, Key or Queue Key actors. Refer to the Actor descriptions in Section 3 of this manual.

The Alias Key/Button column contains the name of any keys that produce the same internal code as the referenced key. Note that when two keys produce the same internal code, any alias of the specified key will cause the same action.

Key/Button	ID	Alias Key/Button
A	A	Configurable Button No. 71
B	B	Configurable Button No. 72
C	C	Configurable Button No. 73
D	D	Configurable Button No. 74
E	E	Configurable Button No. 75
F	F	Configurable Button No. 76
G	G	Configurable Button No. 77
H	H	Configurable Button No. 78
I	I	Configurable Button No. 79
J	J	Configurable Button No. 80
K	K	Configurable Button No. 81
L	L	Configurable Button No. 82
M	M	Configurable Button No. 83
N	N	Configurable Button No. 84
O	O	Configurable Button No. 85
P	P	Configurable Button No. 1
Q	Q	Configurable Button No. 2
R	R	Configurable Button No. 3
S	S	Configurable Button No. 4
T	T	Configurable Button No. 5
U	U	Configurable Button No. 6
V	V	SYST STATS
W	W	CONS STATS
X	X	RECRD
Y	Y	FAST
Z	Z	CANCL PRINT

Key/Button	ID	Alias Key/Button
a	a	DETAIL
b	b	UNIT TREND
c	c	TREND
d	d	BATCH
e	e	SLOT
f	f	SCHEM
g	g	HELP
h	h	DISP SET
i	i	HOUR AVG
j	j	PRIOR DISP
k	k	ASSOC DISP
l	l	DISP BACK
m	m	DISP FWD
n	n	PAGE BACK
o	o	PAGE FWD
p	p	ENTER
q	q	ACK
r	r	SIL
s	s	MSG SUMM
t	t	ALM SUMM
u	u	ALM ANNC
v	v	MSG CONFM
w	w	MSG CLEAR
x	x	UNIT ALM SUMM
y	y	MAN
z	z	AUTO
ACK	ACK	q
ALM ANNC	ALM_ANNC	u
ALM SUMM	ALM_SUMM	t
AM STAT	AMSTATUS	
ASSOC DISP	ASC_DISP	k
AUTO	AUTO	z
BACK SPACE	BK_SPACE	_
BATCH	BATCH	d
BKGND	NORM_BKG	
BKGND	REV_BKG	
BLACK	BLACK	
BLINK (off)	BLINK_OF	
BLINK (on)	BLINK_ON	
BLUE	BLUE	
BREAK	BREAK	
CANCEL	CANCEL	
CANCL PRINT	CNCL_PRT	Z
CENTER	CENTER	
CLR ENTR	CLR_ENT	~
COMM NETWK STAT	COMMNTWK	
COMMAND	COMMAND	
CONS STATS	CON_STAT	W
CURSOR DOWN	CURSOR_D	
CURSOR LEFT	CURSOR_L	
CURSOR RIGHT	CURSOR_R	

Key/Button	ID	Alias Key/Button
CURSOR UP	CURSOR_U	
CYAN	CYAN	
DEL CHAR	DEL_CHAR	
DEL LINE	DEL_LINE	
DELETE	DELETE	
DETAIL	DETAIL	a
DISP BACK	DISP_BK	l
DISP FWD	DISP_FWD	m
DISP SET	DISP_SET	h
ENG keylock level	KEY_ENG	
ENTER	ENTER	p
ERASE	ERASE	
ESCAPE	ESCAPE	
FAST	FAST	Y
FAST LOWER (press)	FAST_LOW	
FAST LOWER (release)	FSTLWRL	#, *and* Configurable Button No. 41
FAST RAISE (press)	FSTRAISE	
FAST RAISE (release)	FSTRASRL	
GREEN	GREEN	
GROUP	GROUP	‘
HELP	HELP	g
HOME	HOME	
HOUR AVG	HOUR_AVG	I
INS CHAR	INS_CHAR	
INS LINE	INS_LINE	
INTEN (low)	INT_LOW	
INTEN (norm)	INT_NORM	
LF (LINE FEED)	LF	
LOAD	LOAD	^
LOWER (press)	LOWER	
LOWER (release)	LOWER_RL (see Note)	

NOTE

For Flag, Digital In, and Digital Out points:

When configuring a button on the configurable keyboard to simulate the LOWER key on the operator’s keyboard, LOWER_RL must be entered into the QUE_KEY actor parameter entry to get the desired effect.

Key/Button	ID	Alias Key/Button
MAGENTA	MAGENTA	
MAN	MAN	y
MENU	MENU	
MSG CLEAR	MSG_CLR	w
MSG CONFM	MSG_CNFM	v
MSG SUMM	MSG_SUMM	s
NORM	NORM	{
ORG SUMM	ORG_SUMM	
OUT	OUT	}
PAGE BACK	PAGE_BK	n
PAGE FWD	PAGE_FWD	o
PR keylock level	KEY_OPR	
PRINT DISP	PRT_DISP	[
PRINT TREND	PRT_TRND	\
PRIOR DISP	PRR_DISP	j
PROC NETWK STAT	PROCNTWK	
RAISE (press)	RAISE	
RAISE (release)	RAISE_RL (See Note)	

NOTE

For Flag, Digital In, and Digital Out points:
When configuring a button on the configurable keyboard to simulate the RAISE key on the operator's keyboard, RAISE_RL must be entered into the QUE_KEY actor parameter entry to get the desired effect.

RECRD	RECORD	X
RED	RED	
RETURN	RETURN	
SCHEM	SCHEM	f
SELECT	SELECT	
SIL	SIL	r
SLOT	SLOT	e
SP (Set Point)	SP	|
SPACE	SPAC_BAR	Configurable Button No. 38
SUP keylock level	KEY_SUP	
SYST MENU	SYS_MENU	]
SYST STATS	SYS_STAT	V
TAB DOWN	TAB_DOWN	
TAB LEFT	TAB_LEFT	
TAB RIGHT	TAB_RGHT	
TAB UP	TAB_UP	
TREND	TREND	c

Key/Button	ID	Alias Key/Button
UNIT ALM SUMM	U_ALM_SM	x
UNIT ASGN	UNIT_ASG	
UNIT TREND	UNT_TRND	b
WHITE	WHITE	
YELLOW	YELLOW	
0	ZERO	Configurable Button No. 54
1	ONE	Configurable Button No. 55
2	TWO	Configurable Button No. 56
3	THREE	Configurable Button No. 57
4	FOUR	Configurable Button No. 58
5	FIVE	Configurable Button No. 59
6	SIX	Configurable Button No. 60
7	SEVEN	Configurable Button No. 61
8	EIGHT	Configurable Button No. 62
9	NINE	Configurable Button No. 63
_	UNDRSCR	BACK SPACE
.	PERIOD	Configurable Button No. 52
!	EXCLMTN	Configurable Button No. 39
"	DBLQUOTE	Configurable Button No. 40
#	POUND	FAST LOWER (release), *and* Configurable Button No. 41
$	DOLLAR	Configurable Button No. 42
%	PERCENT	Configurable Button No. 43
&	AMPER	Configurable Button No. 44
’	APSTRPHE	Configurable Button No. 45
(	L_PAREN	Configurable Button No. 46
)	R_PAREN	Configurable Button No. 47
=	EQUAL	Configurable Button No. 67
^	CARET	LOAD
~	TILDE	CLR ENTR
‘	QUOTE	GROUP
@	AT_SIGN	Configurable Button No. 70
{	L_BRACE	NORM
[	L_BRCKT	PRINT DISP
+	PLUS	Configurable Button No. 49
;	SEMICOL	Configurable Button No. 65
*	STAR	Configurable Button No. 48
:	COLON	Configurable Button No. 64
}	R_BRACE	OUT
]	R_BRCKT	SYST MENU
|	BAR	SP (Set Point)
\	BKSLASH	PRINT TREND
<	LESS	Configurable Button No. 66
>	GREATER	Configurable Button No. 68
,	COMMA	Configurable Button No. 50
?	QUESTN	Configurable Button No. 69
/	SLASH	Configurable Button No. 53
-	DASH	Configurable Button No. 51

Key/Button	ID	Alias Key/Button
Configurable Button No. 1	CNF1	P
Configurable Button No. 2	CNF2	Q
Configurable Button No. 3	CNF3	R
Configurable Button No. 4	CNF4	S
Configurable Button No. 5	CNF5	T
Configurable Button No. 6	CNF6	U
Configurable Button No. 7	AN_CNF7	
Configurable Button No. 8	AN_CNF8	
Configurable Button No. 9	AN_CNF9	
Configurable Button No. 10	AN_CNF10	
Configurable Button No. 11	AN_CNF11	
Configurable Button No. 12	AN_CNF12	
Configurable Button No. 13	AN_CNF13	
Configurable Button No. 14	AN_CNF14	
Configurable Button No. 15	AN_CNF15	
Configurable Button No. 16	AN_CNF16	
Configurable Button No. 17	AN_CNF17	
Configurable Button No. 18	AN_CNF18	
Configurable Button No. 19	AN_CNF19	
Configurable Button No. 20	AN_CNF20	
Configurable Button No. 21	AN_CNF21	
Configurable Button No. 22	AN_CNF22	
Configurable Button No. 23	AN_CNF23	
Configurable Button No. 24	AN_CNF24	
Configurable Button No. 25	AN_CNF25	
Configurable Button No. 26	AN_CNF26	
Configurable Button No. 27	AN_CNF27	
Configurable Button No. 28	AN_CNF28	
Configurable Button No. 29	AN_CNF29	
Configurable Button No. 30	AN_CNF30	
Configurable Button No. 31	AN_CNF31	
Configurable Button No. 32	AN_CNF32	
Configurable Button No. 33	AN_CNF33	
Configurable Button No. 34	AN_CNF34	
Configurable Button No. 35	AN_CNF35	
Configurable Button No. 36	AN_CNF36	
Configurable Button No. 37	AN_CNF37	
Configurable Button No. 38	AN_CNF38	SPACE
Configurable Button No. 39	AN_CNF39	!
Configurable Button No. 40	AN_CNF40	"
Configurable Button No. 41	AN_CNF41	POUND, *and* FAST LOWER (release)

Key/Button	ID	Alias Key/Button
Configurable Button No. 42	AN_CNF42	$
Configurable Button No. 43	AN_CNF43	%
Configurable Button No. 44	AN_CNF44	&
Configurable Button No. 45	AN_CNF45	’
Configurable Button No. 46	AN_CNF46	(
Configurable Button No. 47	CNF47	)
Configurable Button No. 48	CNF48	*
Configurable Button No. 49	CNF49	+
Configurable Button No. 50	CNF50	,
Configurable Button No. 51	CNF51	-
Configurable Button No. 52	CNF52	.
Configurable Button No. 53	CNF53	/
Configurable Button No. 54	CNF54	0
Configurable Button No. 55	CNF55	1
Configurable Button No. 56	CNF56	2
Configurable Button No. 57	A_CNF57	3
Configurable Button No. 58	B_CNF58	4
Configurable Button No. 59	C_CNF59	5
Configurable Button No. 60	D_CNF60	6
Configurable Button No. 61	E_CNF61	7
Configurable Button No. 62	F_CNF62	8
Configurable Button No. 63	G_CNF63	9
Configurable Button No. 64	H_CNF64	:
Configurable Button No. 65	I_CNF65	;
Configurable Button No. 66	J_CNF66	<
Configurable Button No. 67	K_CNF67	=
Configurable Button No. 68	L_CNF68	>
Configurable Button No. 69	M_CNF69	?
Configurable Button No. 70	N_CNF70	@
Configurable Button No. 71	O_CNF71	A
Configurable Button No. 72	P_CNF72	B
Configurable Button No. 73	Q_CNF73	C
Configurable Button No. 74	R_CNF74	D
Configurable Button No. 75	S_CNF75	E
Configurable Button No. 76	T_CNF76	F
Configurable Button No. 77	U_CNF77	G
Configurable Button No. 78	V_CNF78	H
Configurable Button No. 79	W_CNF79	I
Configurable Button No. 80	X_CNF80	J
Configurable Button No. 81	Y_CNF81	K
Configurable Button No. 82	Z_CNF82	L
Configurable Button No. 83	SP_CNF83	M
Configurable Button No. 84	DSHCNF84	N
Configurable Button No. 85	BK_CNF85	O

Key/Button	ID	Alias Key/Button
F1	F1	
F2	F2	
F3	F3	
F4	F4	
F5	F5	
F6	F6	
F7	F7	
F8	F8	
F9	F9	
F10	F10	
F11	F11	
F12	F12	
PF1	PF1	
PF2	PF2	
PF3	PF3	
PF4	PF4	
PF5	PF5	
PF6	PF6	
PF7	PF7	
PF8	PF8	
PF9	PF9	
PF10	PF10	
PF11	PF11	
PF12	PF12	
PF13	PF13	
PF14	PF14	
PF15	PF15	
PF16	PF16	
PF17	PF17	
SHIFT/PF1	PF1_SHFT	
SHIFT/PF2	PF2_SHFT	
SHIFT/PF3	PF3_SHFT	
SHIFT/PF4	PF4_SHFT	
SHIFT/PF5	PF5_SHFT	
SHIFT/PF6	PF6_SHFT	
SHIFT/PF7	PF7_SHFT	
SHIFT/PF8	PF8_SHFT	
SHIFT/PF9	PF9_SHFT	
SHIFT/PF10	PF10_SHF	
SHIFT/PF11	PF11_SHF	
SHIFT/PF12	PF12_SHF	
SHIFT/PF13	PF13_SHF	
SHIFT/PF13	PF14_SHF	
SHIFT/PF15	PF15_SHF	
SHIFT/PF16	PF16_SHF	
SHIFT/PF17	PF17_SHF	

Key/Button	ID	Alias Key/Button
CTL/PF1	PF1_CTL	
CTL/PF2	PF2_CTL	
CTL/PF3	PF3_CTL	
CTL/PF4	PF4_CTL	
CTL/PF5	PF5_CTL	
CTL/PF6	PF6_CTL	
CTL/PF7	PF7_CTL	
CTL/PF8	PF8_CTL	
CTL/PF9	PF9_CTL	
CTL/PF10	PF10_CTL	
CTL/PF11	PF11_CTL	
CTL/PF12	PF12_CTL	
CTL/PF13	PF13_CTL	
CTL/PF14	PF14_CTL	
CTL/PF15	PF15_CTL	
CTL/PF16	PF16_CTL	
CTL/PF17	PF17_CTL	

NOTE

Some of the keys above are present only on certain keyboards. Refer to the *Button Configuration Form Instructions* manual or the *Keyboards* manual for illustrations of the various keyboards.

In Release 520 and later systems, Button IDs for Configurable Buttons can be changed by a user-built Button Name File.

Index

AC_RSS_1 (Actor Example 6)	1.4
Actor Parameters	1.5.1
Actors	
Calling Sequence	1.5
Data Entry	1.5
Error Handling	1.6
Examples	1.4
Testing	1.8
Alarm Annunciator Display, Call Up	2.1
Alarm Priority Color (DDB Variables)	Table A-4
Alarm Summary Display, Call Up	2.2
Area Change Actor	12.1
Area Number (DDB Variables)	Table A-4
Arithmetic Actors	14
Box Status Display, Call Up	2.6
Call Up Actors (see also Display Type)	2.0
Calling Sequence, Actors	1.5.1
Cancel Print	2.44
Centerline Trending	13.2
Change Zone (Standard)	2.8
Change Zone	
Construction, Custom	B.1
Construction, Standard	2.7
User Change Zone	2.57
Check for Variables with Bad Values (R600)	12.2
Check for Non-Existent Variables (R600)	12.3
Clear the Change Zone	2.45
Clear Prompt Region	2.46
Clear the Screen	2.47
Collect History actor	2.48
Combine Variable ID	12.6
Combine Variable (Enumeration Index)	12.5
Comparative Actors	16
Concatenate Strings	12.7
Conditional Actors	17
Console Number (DDB Variables)	Table A-4
Conversion Actors	18
Cross Screen (Conditioning)	2.49
Cross Keyboard (Actor)	3.1
Cursor Positioning	3.8
Custom Change Zone Construction	B
Custom Name Save Display, Call Up	2.7
Database	
Display (Local)	A, C, 1.3
Global	A
System, Actors that Store in	8, 9

The letters A, B, and C indicate Appendices to this publication. When multiple references are shown, bold print is sometimes used to indicate the most often needed information.

Index

Data Entry
Of Actors 1.5
Date/Time, Combining 12.4
DDB Variables Table A-4
Define Data Source Trend Trace 13.3
Delay 12.8
Delete Trace from Trend 13.4
Demand Display Update 2.50
Detail Display, Call Up 2.9
Detail Display, Page n, Call Up 2.10
Display Actors 2
Display Database A, C, 1.3
  Actors that Store in 5
  Honeywell Developers A.6
  Indirection A.4
  User-defined A.3
  Variables Table A-4
Display File (Actor) 2.51
Display Transfer, (Actors) 2.49, 2.52
Display Update (Actors) 2.50, 2.56
Documentation Tool Query Actors 19

Enter 3.2
Enter to Execute 3.3
Equipment List 5.6
Error Handling, Actors 1.6
Event History Retrieval Display, Call Up 2.13
Examples, Actors 1
Extract Entity 12.9
Extract Network ID 12.10

From Screen (Display Transfer) 2.52

General Actors 12
Get Actors 7, 10
  Get Boolean Data 7.1, 10.1
  Get Console Number 12.11
  Get Date/Time Data 7.2, 10.2
  Get Entity ID 7.3, 10.3
  Get Enumeration Data 10.4
  Get Integer Data 7.4, 10.5
  Get Node Number 12.12
  Get Primmod 3.5
  Get Primmod Count 12.13
  Get Primmod Status 12.14
  Get Real Data 7.5, 10.6
  Get Standard Enumeration Data 7.6
  Get Station Number 12.15
  Get String Data 7.7, 10.7
  Get Variable ID 7.8
  System Variable 10.8
  System Variable As A String 10.9

Index

Global Display Database A.2
Group Edit Display, Call Up 2.14
Group Display, Call Up 2.13
Group Trend Display, Call Up 2.16

Hair-Line Cursor Move Actor 13.6
Hardware and Software Keylevels 3.5.1
Hiway Box Point Summary Display, Call Up 2.5
Hiway Status Display, Call Up 2.18

Indirection, Display Database A.4
Input from Operator (Actors) 4.0
Integer To Real Conversion 18.1
Integer To String Conversion 18.2
Internal DDB Entity ID to External Entity ID 18.3
Internal DDB Variable ID to External Entity ID 18.4
Internal DDB Variable ID to External Parameter ID 18.5
Internal DDB Variable ID to External Variable ID 18.6

Key/Button Identifiers C
Key Simulation 3.7
Keylevel Change 3.5
Keylevels - Hardware & Software 3.5.1
Keylevel Reset 3.6
Keylevel Status (DDB) Table A-4

LCN Node Point Summary Display, Call Up 2.23
Local Display Data Bases A.1, 1.3
Logic Manager Diagnostic Display, Call Up 2.19
Logical Actors 15

Message Summary Display, Call Up 2.20
Minimum Alarm Priority for Red (DDB Variable) Table A-4
Move the Cursor 3.8
Multiple Remote Keystroke 3.9
NIM Diagnostic Display, Call Up 2.22
Node Number (DDB Variable) Table A-4
NOP 12.12

Operator Input Actors 4.0
Option 12.13
Organizational Summary Menu, Call Up 2.24
Output a Prompt Message 2.53
Overlay, Call Up (Actors) 2.4, 2.21, 2.25
Overview Display, Call Up 2.26
Overview Edit Display, Call Up 2.27

Palette Actor 2.54
Picture Editor Bad Value Check In Targets 12.3
Point Attribute Summary Display, Call Up 2.3
Point Usage Summary Display, Call Up 2.43
Primmod Actors 3.4, 3.12, 12.11, 12.12
Print File (Actor) 2.55

Index

Process Manager Module Diagnostic Display, Call Up 2.28
Process Manager Status Display, Call Up 2.29
Process Module Display, Call Up 2.30
Process Module Group Display, Call Up 2.15
Process Module Summary Display, Call Up 2.41
Process Variable Retrieval Display, Call Up 2.31
Prompt Message, Output 2.53
Prompt Region 1.7

Queue Key 3.10
Query Actors 19

Range Checking Actors 11
- Range Check Integer Data 11.1
- Range Check Local Data 11.2
- Range Check Real Data 11.3
- Range Check String Data 11.4
- Range Check Variable ID Data 11.5

Read Actors
- Read Data and Store Variable 9.1
- Read Date 4.2
- Read Duration 4.3
- Read Entity 4.5
- Read Entity Name 4.6
- Read Enumeration 4.7
- Read Integer Data 4.8
- Read Parameter ID 4.9
- Read Real Data 4.10
- Read Standard Enumeration Data 4.11
- Read Standard Enumeration Name 4.12
- Read String Data 4.13
- Read Time 4.3
- Read Variable 4.14
- Read and Store Local 6.1

Read Data and Store into Display Database Actors 6
Read Data from Display Database Actors 7
Read Data and Store into System Database Actors 9
Read Data from System Database Actors 10
Real Time Journal Display, Call Up 2.32
Region Number (Actors) 1.7
Region
- Prompt 1.7
- Schematic 1.7
- Screen 1.7

Remote Move 3.11
Removable-Media Initialization Display, Call Up 2.12
Report Menu, Call Up 2.33

Index

Schematic Region 1.7
Schematic, Call Up 2.34
Screen Region Number (Actors) 1.7
Screen Regions 1.7
Set Range for Trend Trace 13.9
Set Scale for Trend 13.10
Set Scroll for Trend 13.11
Set Time Base for Trend 13.12
Standard Change Zone Construction 2.8
Station Number (DDB Variable) Table A-4
Store Actors
- Store Boolean Data 5.1, 8.1
- Store Date 5.2, 8.2
- Store Duration 5.3, 8.3
- Store Entity ID 5.5, 8.5
- Store Enumeration Data 5.8, 8.6
- Store Integer Data 5.7, 8.7
- Store Primmod 3.12
- Store Real Data 5.8, 8.8
- Store String Data 5.10, 8.9
- Store Time 5.4, 8.4

Store to Display Database Actors 5
Store to System Database Actors 8
Store Variable ID 5.11
System Database, Actors that Store in 8, 9
System Database, Actors that Read from 10

Testing Actors 1.8
Text Input Ports (TIP) 6, 9
Time Base, for Trend Actors 13.12
Title Summary Display, Call Up 2.35
Trend Actors 13
Trend Functionality 13.13
Trend Mode Actor 13.5
Trend Overview Display, Call Up 2.37
Trends in Schematics-Construction Example 13.13

UCN I/O Module Diagnostic Display, Call Up 2.38
UCN Slot Hardware Point Summary Display, Call Up 2.17
UCN Status Display, Call Up 2.39
Unit ID To Unit Number Conversion 18.10
Unit Point Summary Display, Call Up 2.40
Unit Trend Display, Call Up 2.42
Update the Display 2.56
User Change Zones 2.57
User-defined Display Database A.4

Variables in Actors 4.14
Variables with Bad Values (R600) 12.2
Variables for Non-Existent Variables (R600) 12.3

ACTORS, ALPHABETICALLY BY CALL NAME

Call Name	Description	Section
ALARMANN	Call Up the Alarm Annunciator Display	2.1
ALARMSUM	Call Up the Alarm Summary Display	2.2
ATTRIBPS	Call Up the Point Attribute Summary Display	2.3
ADD_I	Add Integers	14.1
ADD_R	Add Reals	14.7
AND	And	15.1
AREA_CHG	Area Change	12.1
BAD_VAL	Bad Value	12.2
BACK_OV	Background Overlay	2.4
BOX_PS	Call Up the Hiway Box Point Summary Display	2.5
BOX_STAT	Call Up the Box Status Display	2.6
CANCLPRT	Cancel Print	2.46
CHGZONE	Call Up a Standard Change Zone	2.8
CLEAR_CZ	Clear the Change Zone	2.47
CLR_SCRN	Clear the Screen	2.49
CMP_E	Compare Enumerations	16.4
CMP_I	Compare Integers	16.1
CMP_R	Compare Reals	16.2
CMP_S	Compare Strings	16.3
C_DATTIM	Combine Date/Time	12.4
C_VAR	Combine Variable ID	12.6
C_V_ENUM	Combine Variable (Enumeration Index)	12.5
COLLHIST	Collect History	2.50
CON_NBR	Get Console Number	12.11
CONCAT	Concatenate Strings	12.7
CNV_I	Integer to String Conversion	18.2
CNV_UID	Unit ID To Unit Number Conversion	18.10
CROSSKBD	Cross Keyboard	3.1
CROSSCRN	Cross Screen (Conditioning)	2.51
CUSTSAVE	Call Up the Custom Name Save Display	2.7
DETAIL	Call Up the Detail Display	2.11
DET_PAGE	Call Up the Detail Display, Page n	2.12
DELAY	Delay	12.8
DELFILE	Delete Query Results File	19.6
DIV_I	Divide Integers	14.4
DIV_R	Divide Reals	14.10
DMD_UPD	Demand Display Update	2.52
DSP_FILE	Display File	2.52
EHR	Call Up the Event History Display	2.13
EI_ENT	String to Internal Entity ID	18.9
ELSE	Else	17.2
ENDIF	Endif	17.3
ENTER	Enter	3.2
ENT_EXEC	Enter to Execute	3.3
EQ_LIST	Equipment List Initialize	5.6
EXISTS	Exists	12.2
EXTR_ENT	Extract Entity	12.9
EXTR_ID	Extract Network ID	12.9
FLOAT	Integer to Real Conversion	18.1

ACTORS, ALPHABETICALLY BY CALL NAME

Call Name	Description	Section
FRM_SCRN	From Screen (Display Transfer)	2.54
G_BOOL	Get Boolean Data from Local Display Database	7.1
G_DATIME	Get Date/Time Data from Local Display Database	7.2
G_ENT	Get Entity ID from Local Display Database	7.3
G_INT	Get Integer Data from Local Display Database	7.4
G_PRIM	Get Primmod from Button Lamp Specific Data	3.4
G_REAL	Get Real Data from Local Display Database	7.5
G_SENM	Get Standard Enumeration Data from Local DDB	7.6
G_STR	Get StrinG_Data from Local Display Database	7.7
G_VAR	Get Variable ID from Local Display Database	7.8
GROUP	Call Up the Group Display	2.15
GRP_EDIT	Call Up the Group Edit Display	2.16
GRPTREND	Call Up the Group Trend Display	2.18
GS_BOOL	Get Boolean Data from System Database	10.1
GS_DATIM	Get Date/Time from System Database	10.2
GS_ENM	Get Enumeration Data from System Database	10.4
GS_ENT	Get Entity ID from System Database	10.3
GS_INT	Get Integer Data from System Database	10.5
GS_REAL	Get Real Data from System Database	10.6
GS_STR	Get String Data from System Database	10.7
GS_VAR	Get System Variable from System Database	10.8
GS_VAR_S	Get System Variable as String from System Database	10.9
HWPTSUM	Call Up UCN Slot Summary Display	2.19
HWY_STAT	Call Up the Hiway Status Display	2.20
IE_ENT	Internal DDB Entity ID to External Entity ID	18.3
IE_VAR_E	Internal DDB Variable ID to External Entity ID	18.4
IE_VAR_P	Internal DDB Variable ID to External Parameter ID	18.5
IE_VAR	Internal DDB Variable to External Variable ID	18.6
IF	If	17.1
INITFLOP	Call Up the Removable Media Initialization Display	2.14
IOM_DIAG	UCN IO Module Display	2.40
KEY	Key Simulation	3.7
$KEYCHG	Keylevel Change	3.5
$KEYRST	Keylevel Reset	3.6
LM_DIAG	Call Up Logic Manager Diagnostic Display	2.21
MOD_I	Modulo Integers	14.5
MODGROUP	Call Up the Process Module Group Display	2.17
MOVE	Move the Cursor	3.8
MSGSUM	Call Up the Message Summary Display	2.22
MUL_I	Multiply Integers	14.3
MUL_R	Multiply Reals	14.9
MULT_OV	Call Up Multiple Serial Overlays	2.23
NEG_I	Negate Integer	14.6
NEG_R	Negate Real	14.11
NIM_DIAG	Call Up the NIM Diagnostic Display	2.24
NODE_NBR	Get Node Number	12.12
NODE_PS	Call Up the LCN Node Point Summary Display	2.25
NOP	No Operation	12.16
NOT	Not	15.4

ACTORS, ALPHABETICALLY BY CALL NAME

Call Name	Description	Section
OR	Or	15.2
OPTION	Option	12.17
OSUMMENU	Call Up the Organizational Summary Display	2.26
OVERLAY	Call Up Overlay	2.27
OVERVIEW	Call Up the Overview Display	2.28
OVW_EDIT	Call Up the Overview Edit Display	2.29
PALETTE	Palette	2.56
PMM_DIAG	Process Manager Module Diagnostic Display	2.30
PM_STAT	Process Manager Status Display	2.31
PRT_FILE	Print File	2.57
PRIMCNT	Get Primmod Count	12.13
PRIMSTS	Get Primmod Status	12.14
PROCMOD	Call Up the Process Module Display	2.32
PROMPT	Output a Prompt Message	2.55
PROMPT_C	Clear the Prompt Region	2.48
PVR	Call Up the Process Variable Retrieval Display	2.33
QUE_KEY	Queue Key	3.10
QCANCEL	Cancel Query	19.4
QCLEAR	Clear Query Status	19.5
QDISPLAY	Invoke Query With Display Output	19.1
QFILE	Invoke Query With File Output	19.2
QPRINT	Invoke Query With Printer Output	19.3
QRESULTS	Display Query Results	19.7
R_BOOL	Read Boolean Data through Text Input Port	4.1
R_DATE	Read Date through Text Input Port	4.2
R_DUR	Duration	4.4
R_ENM	Read Enumeration through Text Input Port	4.7
R_ENT	Read Entity through Text Input Port	4.5
R_ENT_N	Read Entity Name through Text Input Port	4.6
R_INT	Read Integer Data through Text Input Port	4.8
R_PAR	Read Parameter ID	4.10
R_REAL	Read Real Data through Text Input Port	4.11
R_SENM	Read Standard Enumeration Data through TIP	4.12
R_SENM_N	Read Standard Enumeration Name through TIP	4.13
R_STR	Read String Data through Text Input Port	4.14
R_TIME	Read Time through Text Input Port	4.3
R_VAR	Read Variable through Text Input Port	4.15
REM_MOVE	Remote Move	3.11
REM_STR	Multiple Remote Keystroke	3.9
REP_MENU	Call Up the Report Menu	2.35
ROUND	Real to Integer Conversion by Rounding	18.7
RNG_INT	Range Check Integer Data	11.1
RNG_LOC	Range Check Local Data	11.2
RNG_REAL	Range Check Real Data	11.3
RNG_STR	Range Check String Data	11.4
RNG_VAR	Range Check Variable Data	11.5
RS_LOC	Read and Store Local	6.1
RS_SYS	Read Data and Store Variable	9.1
RTJ	Call Up the Real Time Journal Display	2.34

ACTORS, ALPHABETICALLY BY CALL NAME

Call Name	Description	Section
SCHEM	Call Up Schematic	2.36
S_BOOL	Store Boolean Data in Local Display Database	5.1
S_DATE	Store Data in Local Display Database	5.2
S_DUR	Duration	5.3
S_ENT	Store Entity ID in Local Display Database	5.5
S_INT	Store Integer Data in Local Display Database	5.7
S_PRIM	Store Primmod to Button Lamp Specific Data	3.12
S_REAL	Store Real Data in Local Display Database	5.8
S_SENM	Store Standard Enumeration Data in Local DDB	5.9
S_STR	Store String Data in Local Display Database	5.10
S_TIME	Store Time in Local Display Database	5.4
S_VAR	Store Variable ID in Local Display Database	5.11
SS_BOOL	Store Boolean Data in System Database	8.1
SS_DATE	Store Date in System Database	8.2
SS_DUR	Store Duration Date in System Database	8.3
SS_ENM	Store Enumeration Data in System Database	8.6
SS_ENT	Store Entity ID in System Database	8.5
SS_INT	Store Integer Data in System Database	8.7
SS_REAL	Store Real Data in System Database	8.8
SS_STR	Store String Data in System Database	8.9
SS_TIME	Store Time in System Database	8.4
STN_NBR	Get Station Number	12.15
SUB_I	Subtract Integers	14.2
SYS_JRNL	System Status Journal	2.38
SUB_R	Subtract Reals	14.8
TITLESUM	Call Up the Title Summary Display	2.37
TR_ADD	Add Trace to Trend	13.1
TR_CLINE	Centerline Trending	13.2
TR_DATA	Define Data Source for Trend Trace	13.3
TR_DEL	Delete Trace from Trend	13.4
TRENDOVW	Call Up the Trend Overview Display	2.39
TRUNK	Real to Integer Conversion by Truncation	18.8
TR_MODE	Trend Mode Actor	13.5
TR_RSTR	Display (Restore) Saved Trend Settings	13.8
TR_RANGE	Set Range for Trend Trace	13.9
TR_SAVE	Save Set of Trend Settings	13.7
TR_SCALE	Set Scale for Trend	13.10
TR_SCRLL	Set Scroll for Trend	13.11
TR_TIME	Set Time Base for Trend	13.12
UCN_STAT	UCN Status Display	2.41
UNIT_PS	Call Up the Unit Point Summary Display	2.42
UNITSUM	Call Up the Process Module Summary Display	2.43
UNITTRND	Call Up the Unit Trend display	2.44
UPDATE	Update the Display	2.58
USAGE_PS	Call Up the Point Usage List Display	2.45
USER_CZ	User Change Zone	2.59
XOR	Xor (Exclusive OR)	15.3

FAX Transmittal **FAX No.: (602) 313-4212**

TO: Information Development **Total FAX pages:** _____
(including this page)

Reader Comments

Title of Document: **Actors Manual**

Document Number: **SW09-655 R620** Issue Date: **12/00**

Comments: _____

Recommendations: _____

FROM:

Name: _____ **Date:** _____

Title: _____

Company: _____

Address: _____

City: _____ **State:** _____ **ZIP:** _____

Telephone: _____ **FAX:** _____

FOR ADDITIONAL ASSISTANCE:

WRITE	CALL
Honeywell Industrial Automation and Control 16404 N. Black Canyon Highway Phoenix, AZ 85053	You may also call 800-343-0228 (available in the 48 contiguous states).

Honeywell

Industrial Automation and Control

16404 N. Black Canyon Highway

Phoenix, AZ 85053