Honeywell

Equipment List Reference Manual

SW27-560

Honeywell

Implementation
Engineering Operations - 3

Equipment List Reference Manual

SW27-560
Release 500
9/95

Copyright, Notices, and Trademarks

Revision 01 – September 1, 1995

This document was prepared using Information Mapping® methodologies and formatting principles.

Honeywell
Industrial Automation and Control
Automation College
2820 West Kelton Lane
Phoenix, AZ 85023

(602) 313-5669

About This Publication

This publication describes the Equipment List function and explains how to use it. The Equipment List function is a set of enhancements introduced in software release R400. You should consider using the Equipment List function if you have a number of similar equipments, such as reactors, performing similar functions. The Equipment List function allows you to write a generic CL sequence control program, and build a generic schematic or schematics. In doing so, you avoid having to generate individual CL programs and schematics for each of the similar equipment units.

This publication introduces the concepts of Equipment Unit and Equipment Unit Class. It shows you how to define a virtual, or generic, Equipment Unit Class model using virtual, or “alias” names instead of specific tag names and constant names. You incorporate this model in an Equipment List source file, in which you also include a list of the actual tag names and constant values for each specific equipment unit (reactor, in this example). You build the Equipment List source file using the Equipment List Builder, which is similar to a compiler. You then use the resulting object file in generic CL programs and generic schematics.

The Equipment List function is applicable to CL/MC, CL/AMC, CL/PM, and CL/APM sequence programs and associated schematics.

The Equipment List function requires software release R400 or later.

Table of Contents

SECTION 1 – EQUIPMENT LIST INTRODUCTION 1

1.1 Overview 1
1.2 Example of a Virtual Model 2
1.3 Example of an Equipment List Source File 4
1.4 Example of Equipment List Use in CL 6
1.5 Example of Equipment List Use in Schematics 8

SECTION 2 – CONFIGURING FOR EQUIPMENT LIST 11

2.1 Overview 11
2.2 What the Procedure Does 13
2.3 Procedure to Configure a US 14
2.4 Example of Configuring a US for Equipment List 15

SECTION 3 – EQUIPMENT LIST SOURCE FILE 17

3.1 Overview 17
3.2 Structure of the Equipment List Source File 18
3.3 Equipment Unit Class (EUC) Header 20
3.4 Equipment Unit Class (EUC) Alias Names List 22
3.5 Unit Instance List 26
3.6 Identifiers 28
3.7 Reserved Words and Predefined Identifiers 29
3.8 The DDB Attribute 30

SECTION 4 – BUILDING THE EQUIPMENT LIST 33

4.1 Overview 33
4.2 Invoking the Equipment List Builder 34
4.3 Example of an ELB Listing File 35

SECTION 5 – USING EQUIPMENT LIST IN CL 37

5.1 Overview 37
5.2 Preparing a Generic CL Source File 38
5.3 Compiling a Generic CL Source File 39
5.4 New CL Compiler Command Options 41
5.5 Loading Generic CL 43
5.6 Example of a CL Listing File 44

SECTION 6 – USING EQUIPMENT LIST IN SCHEMATICS 47

6.1 Overview 47
6.2 Introduction 48
6.3 New Picture Editor Commands 50
6.4 The EQ_LIST Actor 52

Figures and Tables

Figure 1-1 Virtual Model Illustration 3
Figure 1-2 Generic Schematic Example 9
Figure 1-3 Using the Generic Schematic 10
Figure 2-1 Example of NCF Changes for Equipment List Support 14
Figure 3-1 Equipment List Source File Structure 18
Figure 3-2 Equipment List Syntax Diagram 19
Figure 3-3 EUC Header Syntax Diagram 20
Figure 3-4 Network Name Syntax Diagram 21
Figure 3-5 Class Attribute Syntax Diagram 21
Figure 3-6 Alias Names List Syntax Diagram 22
Figure 3-7 Virtual Object Definition Syntax Diagram 24
Figure 3-8 Object Attribute Syntax Diagram 24
Figure 3-9 Unit Instance List Syntax Diagram 27
Figure 3-10 DDB Attribute Syntax Diagram 30

Table 2-1 External Load Modules Required for Each Personality 12
Table 2-2 US Configuration Procedure 13
Table 3-1 Type Expressions Available for Virtual Tags 25
Table 3-2 Type Expressions Available for Virtual Constants 25

Acronyms

AMC.....Advanced Multifunction Controller
APM..... Advanced Process Manager
CL..... Control Language
DDB.....Display Database
ELB ..... Equipment List Builder
EUC..... Equipment Unit Class
MC.....Multifunction Controller
PM..... Process Manager

References

Publication Title	Publication Number	Binder Title	Binder Number
Control Language/ Process Manager Reference Manual	PM27-510	Implementation/ Process Manager - 2	TDC 3040-2
Control Language/Advanced Process Manager Reference Manual	AP27-510	Implementation/ Advanced Process Manager-2	TDC 3042-2
Control Language/Multifunction Controller Reference Manual	PC27-510	Implementation/ Hiway Gateway - 2	TDC 3034-2
Picture Editor Reference Manual	SW09-550	Implementation/ Engineering Operations - 2	TDC 3032-2

Section 1 – Equipment List Introduction

1.1 Overview

Purpose	This section introduces The Equipment List function and defines the concepts of Equipment Unit and Equipment Unit Class.
What is the Equipment List function?	The Equipment List function is a group of related software enhancements introduced in software release R400. One of these enhancements is a new compiler-like tool, called the Equipment List Builder (ELB). Also, major enhancements were made to the CL Compiler and the Picture Editor to enable them to support the Equipment List function.
Where is it used?	The Equipment List function is applicable to schematics, and to CL programs in the Multifunction Controller, the Advanced Multifunction Controller, the Process Manager, and the Advanced Process Manager.
When is it used?	The Equipment List function is useful when you have a number of similar sets of process equipment performing similar functions. In order to be more specific, we will discuss the concepts of Equipment Unit and Equipment Unit Class.
Equipment Unit definition	An Equipment Unit is a specific set of process apparatus, operating as an integral piece of equipment, performing a specific function in the manufacturing of a product. Typically, an Equipment Unit is a vessel and its associated control devices (for example—a reactor, a premixer, or a dosing tank).
Equipment Unit Class definition	Equipment Units can be grouped together into classes based on their structure and function. An Equipment Unit Class is a set of identical or similar Equipment Units that perform the same or similar function, have the same complement of control devices, and utilize the same control strategy. For example, your plant might have three similar reactors. You could define an Equipment Unit Class of REACTOR, and you could identify the three specific Equipment Units as REACT1, REACT2, and REACT3.
What the Equipment List function will do	If you have an Equipment Unit Class with similar or identical Equipment Units as described above, the Equipment List function will allow you to develop a generic CL sequence program and one or more common schematics that will apply to all of the individual units. This process begins with the construction of a virtual model of the Equipment Unit Class, which will be covered next.

1.2 Example of a Virtual Model

Introduction

If you have an Equipment Unit Class with similar or identical Equipment Units as we have described, you can construct a generic or virtual model of the equipment in the class. You can assign virtual or "alias" names for the tag names and constants in the model. These virtual names correspond on a one-to-one basis with the specific tag names in the individual Equipment Units.

An example

Figure 1-1 illustrates this concept. The figure shows a virtual model of the class TANK, which represents the common structure of the individual units TANK1, TANK2, and TANK3. Virtual or alias ("dummy") tag names are assigned in the model, whereas the actual tag names are used in the specific Equipment Units (which are also called Unit Instances).

Note: The example in Figure 1-1 is not a complete system. It includes a vessel, and a fill valve and totalizer point to control filling the vessel. This highly simplified example will be used throughout this introductory section to illustrate the concepts of the Equipment List function.

Alias entity names in the virtual model

The model itself is not a real piece of equipment—it simply represents the actual Equipment Units. The alias entity (tag) names in the model are likewise not the names of actual control elements—they are "dummy" names that represent the actual physical control element names in each of the real units. The table in Figure 1-1 shows the correspondence between the alias names and the actual tag names. For example, the valve in the virtual model is assigned the alias name "VALVE." In the actual Equipment Unit TANK1, the corresponding valve has the real tag name "VALVE1." Similarly, in TANK2 the valve has the tag name "VALVE2" and in TANK3 it is "VALVE3."

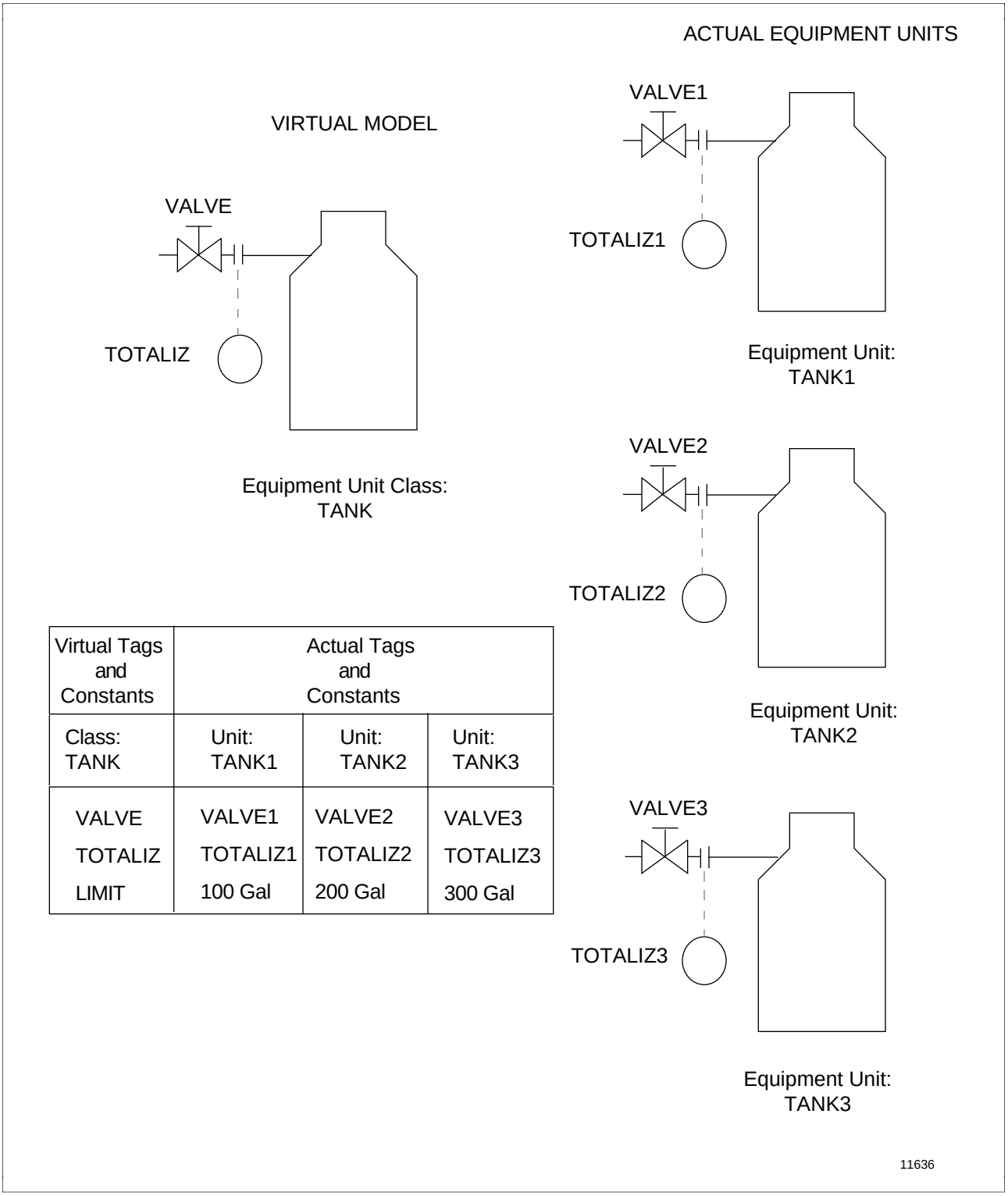
VALVE1, VALVE2, and VALVE3 are real points that are configured in the system, whereas VALVE is an artificial point that is used as a substitute for VALVE1, VALVE2, and VALVE3 in the virtual model.

Alias constant names in the virtual model

Virtual constants can also be assigned in the model and they correspond to actual constants in the real units. The actual constants can have different values in each of the real units. In the example, Figure 1-1, assume we wish to charge each of the tanks with a different amount of liquid. As shown in the table in Figure 1-1, the alias name "LIMIT" is used in the virtual model to represent the desired fill amount. In actual unit TANK1, the value 100 is assigned to LIMIT. In TANK2, the value is 200, and in TANK3 the value is 300. The assignment of actual values is done in the Equipment List source file, which will be covered next.

Continued on next page

Figure 1-1 Virtual Model Illustration



Virtual Tags and Constants	Actual Tags and Constants		
Class: TANK	Unit: TANK1	Unit: TANK2	Unit: TANK3
VALVE	VALVE1	VALVE2	VALVE3
TOTALIZ	TOTALIZ1	TOTALIZ2	TOTALIZ3
LIMIT	100 Gal	200 Gal	300 Gal

1.3 Example of an Equipment List Source File

What is the Equipment List Builder?

The Equipment List Builder (ELB) is used to build Equipment List object files from Equipment List source files. These source files have a suffix of ".QS" and are generated with the Text Editor. The ELB is similar in function and execution to the CL Compiler.

The ELB source file

A simple example of an ELB source file is shown on the next page. The source file corresponds to the example shown in Figure 1-1.

Note that the ELB source file contains an Alias Names definition section. Alias names are preceded by an "at" symbol (@). The Alias Names section contains a series of declaration statements that give the alias names of the class elements and identify the data type for each.

Following the Alias Names section, there is a Unit Instance section for each Equipment Unit. Each of these contains a list of assignment statements that assign to each alias name a specific tag name or value appropriate for that Equipment Unit.

Note that each Unit Instance section contains exactly the same elements that appear in the Alias Names section, and that the elements in the Alias and Unit Instance sections are listed in the same order.

This example illustrates the general structure of the ELB source file. The ELB source file will be covered in more detail in Section 3.

In the Unit Class definition, any alias name that will be used in a schematic must be followed by a DDB attribute (display database address), as illustrated in the example.

Build points

Prior to compiling the Equipment List, all real entities (points) used in the Equipment List source file must be built. One of the functions of the ELB is to verify that real entities referenced in the Unit Instances exist in the system. The ELB will generate an error for any referenced point that does not exist.

Build Equipment List

Note: An NCF change is required to run the ELB—see Section 2. You may run the ELB when the source file is completed, the points are built, and the NCF is changed. See 4.2 for the command syntax for the ELB. If there are errors, the builder will generate an error listing with a ".QE" suffix. If there are no errors, the builder will generate a listing with a ".QL" suffix and an object file with a ".QO" suffix. For example, if the source file is TANK.QS, the builder will generate TANK.QE, or TANK.QL and TANK.QO.

Continued on next page

1.3 Example of an Equipment List Source File, Continued

ELB source file example

The following is an example of an ELB source file. It corresponds to the virtual model shown in Figure 1-1.

```
UNIT_CLASS @TANK : UCN            -- ALIAS TANK DESCRIPTION
  ALIAS
    @PMDP         :  PROCMODL     -- ALIAS PROCESS MODULE DATA POINT
    @VALVE        :  DIGCOM       -- ALIAS FLOW VALVE
       DDB(300)
    @TOTALIZ      :  REGPV        -- ALIAS TOTALIZER POINT
       DDB(301)
    @LIMIT        :  NUMBER       -- ALIAS TARGET LEVEL FOR TANK
       DDB(302)
    @WHICH        :  NUMBER       -- IDENTIFIES WHICH TANK
       DDB(303,INTEGER)           -- FORCES “WHICH” TO BE AN INTEGER
  END ALIAS

  UNIT_INSTANCE @TANK  =  "TANK1" -- SPECIFIC DESCRIPTION FOR TANK1
    @PMDP      =    FILL1         -- TANK1 PROCESS MODULE DATA POINT
    @VALVE     =    VALVE1        -- TANK1 FLOW VALVE
    @TOTALIZ   =    TOTALIZ1      -- TANK1 TOTALIZER POINT
    @LIMIT     =    100           -- TANK1 FILL LIMIT
    @WHICH     =    1             -- NUMBER IDENTIFIES TANK 1
  END UNIT_INSTANCE

  UNIT_INSTANCE @TANK  =  "TANK2" -- SPECIFIC DESCRIPTION FOR TANK2
    @PMDP      =    FILL2         -- TANK2 PROCESS MODULE DATA POINT
    @VALVE     =    VALVE2        -- TANK2 FLOW VALVE
    @TOTALIZ   =    TOTALIZ2      -- TANK2 TOTALIZER POINT
    @LIMIT     =    200           -- TANK2 FILL LIMIT
    @WHICH     =    2             -- NUMBER IDENTIFIES TANK 2
  END UNIT_INSTANCE

  UNIT_INSTANCE @TANK  =  "TANK3" -- SPECIFIC DESCRIPTION FOR TANK3
    @PMDP      =    FILL3         -- TANK3 PROCESS MODULE DATA POINT
    @VALVE     =    VALVE3        -- TANK3 FLOW VALVE
    @TOTALIZ   =    TOTALIZ3      -- TANK3 TOTALIZER POINT
    @LIMIT     =    300           -- TANK3 FILL LIMIT
    @WHICH     =    3             -- NUMBER IDENTIFIES TANK 3
  END UNIT_INSTANCE

END UNIT_CLASS
```

1.4 Example of Equipment List Use in CL

Prepare a generic CL sequence program

After the Equipment List is successfully built, it can be used in a generic CL sequence program that is based on the virtual model. Compiling this generic program results in a separate object file for each specific Unit Instance.

Including the Equipment List

In the sequence program, include the Equipment List object file as the first statement that is not a comment. The new CL statement for this purpose is:

```
%INCLUDE_EQUIPMENT_LIST <name>
```

Use alias names

In the generic CL program, use the alias names (preceded by an @) that were defined in the Unit Class portion of the Equipment List source file. The alias names are used in place of the specific tag names and constants.

Compiling Generic CL

Note: An NCF change is required to compile a CL program that has an Equipment List included—see Section 2. Compile the CL program in the usual way—just as you would if it did not include an Equipment List. You will see multiple executions (one for each Unit Instance) of Pass 2 and Pass 3 . If the compile is error-free, a CL object file is generated for each Unit Instance.

Load the CL

Copy all of the object files generated by the compile into the &Enn volume or directory (nn is the Hiway or UCN number). Call up the detail status display for the Process Module Data Point of each Unit Instance (FILL1, FILL2, and FILL3 in our example) and load the sequence program in each.

Continued on next page

1.4 Example of Equipment List Use in CL, Continued

CL source file example

The following example shows a generic CL sequence program that could be used to control the three tanks in our example. The items of particular interest are:

- Use of the `%INCLUDE_EQUIPMENT_LIST` statement.
- Use of alias names for all tags and constants.

```
%INCLUDE_EQUIPMENT_LIST TANK

SEQUENCE  FILL  (APM; POINT @PMDP)

EXTERNAL  @VALVE, @TOTALIZ

PHASE ONE

  SET   @TOTALIZ.COMMAND  =   STOP       -- INITIALIZE TOTALIZER
  SET   @TOTALIZ.COMMAND  =   RESET
  SET   @TOTALIZ.AVTV     =   @LIMIT     -- DESIRED TANK FILL AMOUNT
  SET   @VALVE.MODATTR    =   PROGRAM

  SEND: "CHARGING TANK WITH ",@LIMIT," GALLONS"

  SET   @TOTALIZ.COMMAND  =   START      -- START TOTALIZER
  OPEN  @VALVE                           -- BEGIN FILLING TANK
  WAIT  @TOTALIZ.AVTVFL   =   ON         -- WAIT FOR DESIRED FILL AMOUNT
  CLOSED   @VALVE                        -- CLOSE THE FILL VALVE
  SET   @TOTALIZ.COMMAND  =   STOP       -- AND STOP THE TOTALIZER

  SEND: "TANK CHARGED WITH ",@LIMIT," GALLONS"

END FILL
```

1.5 Example of Equipment List Use in Schematics

Set up the NCF

First, you must configure the Universal Station for generic schematics. This requires an on-line NCF change. The procedure involves configuring some external load modules and the memory they require, and is covered in detail in Section 2.

Three new commands

There are three new Picture Editor commands to support the Equipment List function. They are:

- **LOADEQ**—Load Equipment List. Brings the alias names of the Equipment List object file into the schematic. This allows generic (alias) names to be used in the schematic.
- **UNLOADEQ**—Unload Equipment List. Used to delete alias names of an Equipment List object file from the schematic.
- **LISTEQ**—List Equipment Lists. Lists the Equipment Lists already loaded in the schematic.

Load the Equipment List

When you enter the Picture Editor to build a new generic schematic or to edit an existing generic schematic, you must load the Equipment List object file using the LOADEQ command. For our example, the command would be:

```
LOADEQ TANK
```

Build the schematic

Build the schematic, using alias names (without the "@") for tag names and constants that you reference. Remember that each alias name used in a schematic requires a DDB attribute in the Unit Class definition in the Equipment List source file (see 1.3).

The EQ_LIST actor

A new runtime actor is used in schematics to replace the alias names used with the actual names for a specific Unit Instance. (It does this by updating the DDB locations.) The actor has the syntax:

```
EQ_LIST (Equip_List_Object_File_Name, Unit_Instance_Name)
```

Use of alias names in schematic example

Figure 1-2 shows a generic schematic for our example (the system of Figure 1-1). The schematic was prepared as follows:

- The value following the text "VALVE" is the alias VALVE.PV (an enumeration shown as "EEEEEE").
- The value following the text "LEVEL" is the alias TOTALIZ.PV (a real shown as "RRRRRR").
- The value following the text "TANK" is the alias WHICH (an integer constant shown as "I"—see 3.8).

Continued on next page

Use of the EQ_LIST actor in schematic example

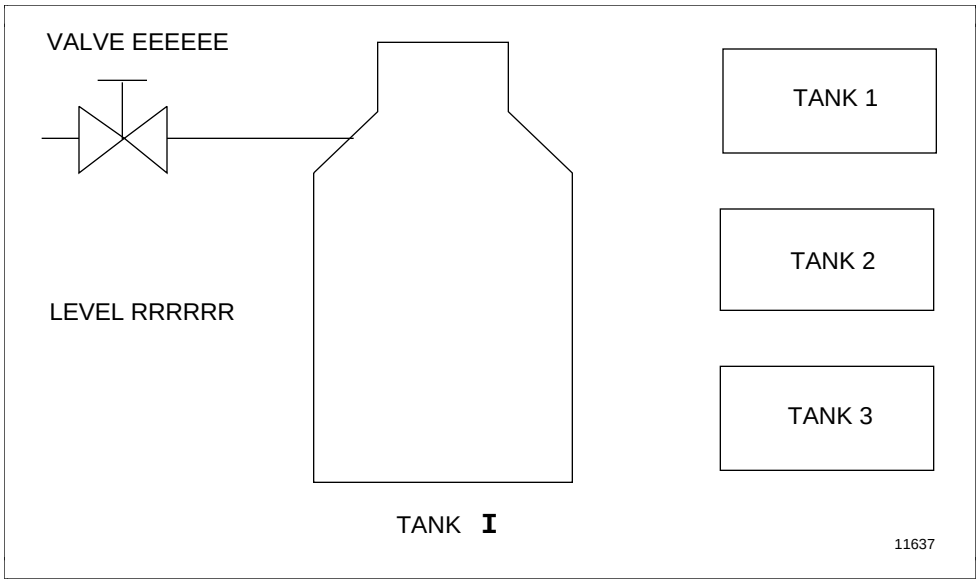
The three targets, labeled TANK 1, TANK 2, and TANK 3, use the actor EQ_LIST as follows:

- Target TANK 1 is assigned the value EQ_LIST (“TANK”,”TANK1”)
- Target TANK 2 is assigned the value EQ_LIST (“TANK”,”TANK2”)
- Target TANK 3 is assigned the value EQ_LIST (“TANK”,”TANK3”)

The generic schematic

The following figure shows the generic schematic after it is compiled in the Picture Editor.

Figure 1-2 Generic Schematic Example



Using the generic schematic

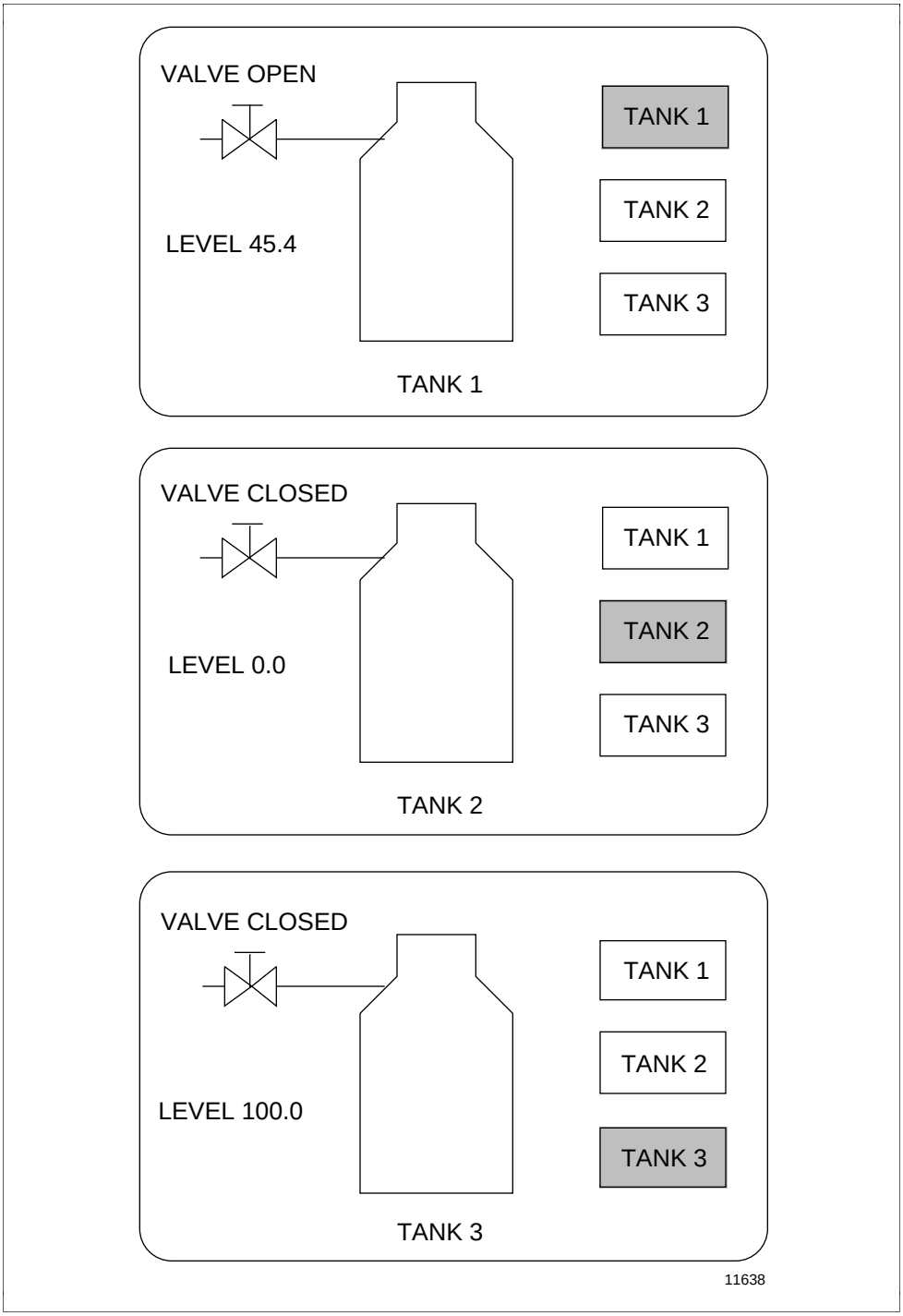
Figure 1-3 on the next page shows how the generic schematic can be used to display the status of each of the three Unit Instances in our example. The figure shows three separate displays that can be obtained by selecting the three targets. When the TANK 1 target is selected, values for that Unit Instance are displayed. Similarly, when the TANK 2 or TANK 3 targets are selected, their respective values are displayed, as indicated in the figure.

Continued on next page

Examples of generic schematic use

The following figure shows how the generic schematic can be switched to display values for any Unit Instance.

Figure 1-3 Using the Generic Schematic



Section 2 – Configuring for Equipment List

2.1 Overview

Introduction

Equipment List support options may be configured during installation of R400. This is covered in subsection G.3.4.15 of the *Customer Release Guide*. If you did not install this option at that time, or wish to add Equipment List support to one or more Universal Stations now, use the procedure in this section.

Each US must be changed to use Equipment List

If Equipment List support was not configured during installation of R400, you must make an on-line NCF change to each Universal Station that will be used for any of the following activities:

- Building Equipment Lists with the Equipment List Builder.
- Compiling generic CL programs that include an Equipment List object file (the CL source file uses the `%INCLUDE_EQUIPMENT_LIST` statement).
- Building or compiling a generic schematic (one that has an Equipment List object file loaded by using the `LOADEQ` command). Requires Equipment List support in Engineering Personality or Universal Personality.
- Displaying a generic schematic. Equipment List support in Operator Personality is all that is required, although you may prefer to use Universal Personality.

Preparation

You will need the following information and media to perform the procedure:

- Prepare a list of the USs you wish to modify.
- List the node number of each US to be modified.
- List the personality or personalities that will need to support Equipment List for each US to be modified. For example, a US that will only be used to display generic schematics may only need Equipment List support in Operator Personality. Compiling Equipment Lists or generic CL sequences, or building generic schematics, would require support in Engineering or Universal Personality.
- Have available the removable media NCF backup.
- Be sure that the NCF backup path is set to one of the removable media drives on the US at which you will perform the modification.

If you have a History Module

Copy the files required for Equipment List support to the History Module. Note: This may have been done when your software was installed or upgraded. The following procedure checks for the files and how to copy them if required.

Using the Command Processor, determine if the directories &OVG and &CUS exist on the History Module:

```
LSV NET
```

If the directories are present, verify that the directory &OVG contains the file ELB.OV. Also verify that the directory &CUS contains the files shown under Name in Table 2-1 (these files have the extension .LO):

```
LS NET>&OVG
LS NET>&CUS
```

If the directories are not present, create them:

```
CD NET>vol>&OVG     (vol is an existing volume on the HM)
CD NET>vol>&CUS
```

If you had to create the directories or if they did not contain the required files, copy the required files from the &C1 (68000) or &C2 (68020) cartridges:

```
CP $Fx>&OVG>ELB.OV  NET>&OVG>=
CP $Fx>&CUS>*.LO  NET>&CUS>=
```

Set the default pathname for the EXT LOAD MODULE to NET>&CUS and the default pathname for GENERIC OVERLAYS to NET>&OVG.

Floppy-based systems

The floppy &OVG contains the overlay file ELB.OV, and also contains a directory &CUS that contains the files listed in Table 2-1 with the extension of .LO. Mount this floppy and set the default pathname for the EXT LOAD MODULE to $Fx>&CUS and the default pathname for GENERIC OVERLAYS to $Fx>&OVG.

Cartridge-based systems

The cartridge &C1 (68000) or &C2 (68020) contains the directory &OVG with the overlay file ELB.OV, and also contains a directory &CUS that contains the files listed in Table 2-1 with the extension of .LO. Mount this cartridge and set the default pathname for the EXT LOAD MODULE to $Fx>&CUS and the default pathname for GENERIC OVERLAYS to $Fx>&OVG.

2.2 What the Procedure Does

Configure optional external load modules

As you perform the procedure in Table 2-2, you are configuring optional external load modules that are required to support Equipment List. You will also be configuring additional memory that is required for these modules.

You will select the modules to configure based on the personality or personalities that you will use at that Universal Station.

Which modules to configure

The following table shows the modules that must be configured to support Equipment List for each personality. In the example in Figure 2-1, modules are configured for all three personalities.

Table 2-1 External Load Modules Required for Each Personality

IF you will load…	THEN configure …	
	Name	**Pers.**
Operator Personality	OPBASE OPEQLT	OPR OPR
Engineering Personality	MSF EPBASE EPEQLT	EP EP EP
Universal Personality	MSF UPBASE UPEQLT	UP UP UP

2.3 Procedure to Configure a US

Perform this procedure

Perform the following procedure for each US that is to be configured to support Equipment List. Refer to Figure 2-1 for an example showing how the display appears.

Table 2-2 US Configuration Procedure

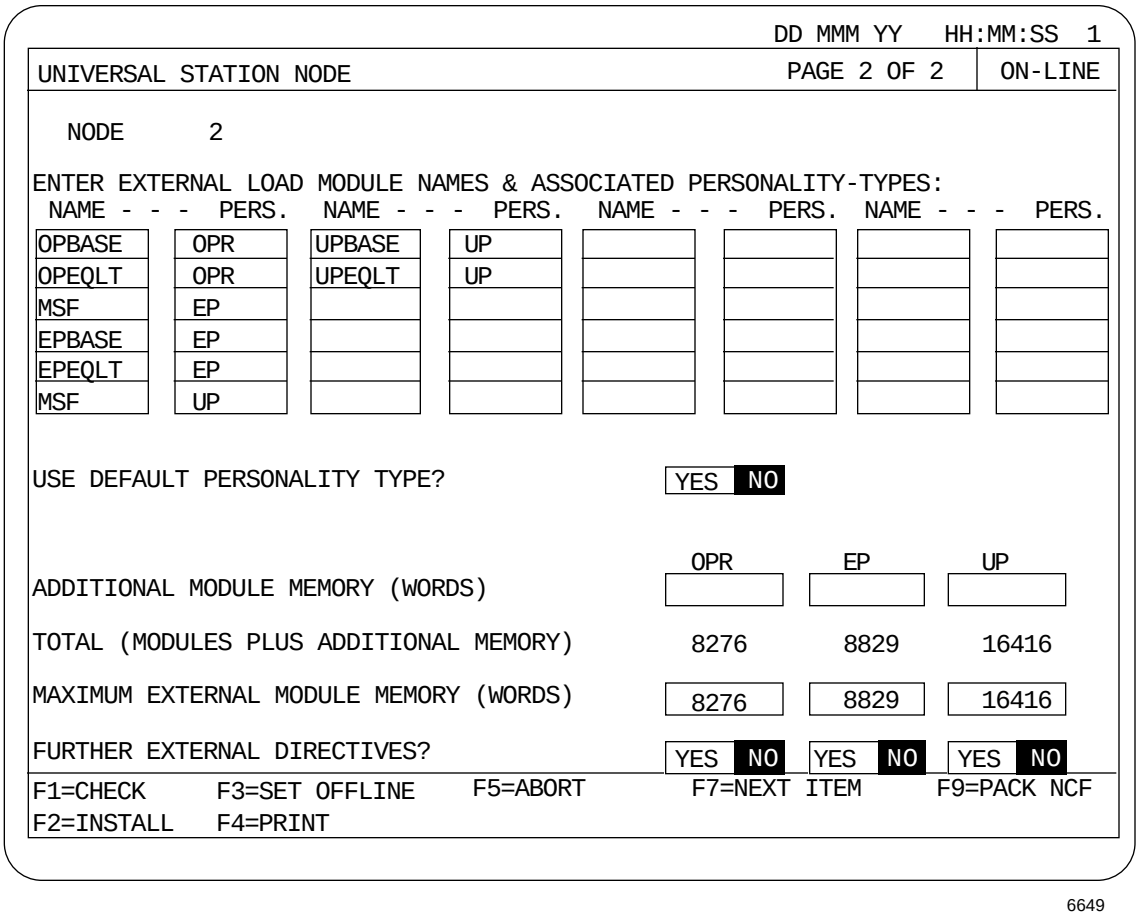
Step	Action
1	Bring up the Engineering Main Menu from the Engineering or Universal Personality.
2	Select the LCN NODES target.
3	Select the node number of the US that you wish to modify.
4	Select the MODIFY NODE target.
5	Enter the file names and personality type from Table 2-1 for the modules you wish to configure. See the example in Figure 2-1.
6	Select NO for "USE DEFAULT PERSONALITY TYPE?"
7	Press [ENTER].
8	Enter the numbers from the line "TOTAL (MODULES PLUS ADDITIONAL MEMORY)" into the entry ports below them on the line "MAXIMUM EXTERNAL MODULE MEMORY (WORDS)"—see Figure 2-1 for an example.
9	Press [ENTER].
10	Press [CTL F1] to run checker.
11	Mount the backup NCF in $Fd (the NCF backup path specified in the pathname catalog).
12	Press [CTL F2] to install.
13	Press [ENTER].
14	Shut down, and then load the US.
15	Repeat for all desired Universal Stations.

2.4 Example of Configuring a US for Equipment List

Screen used to configure US

Figure 2-1 shows Page 2 of the Universal Station Node display. This figure shows a node that is configured to support Equipment List in Operator, Engineering, and Universal Personality. You may omit the external load modules for any personalities that you will not use in that node.

Figure 2-1 Example of NCF Changes for Equipment List Support



6649

Section 3 – Equipment List Source File

3.1 Overview

Introduction

This section covers the structure and preparation of the Equipment List source file.

The Equipment List source file is a text file prepared with the Text Editor. It has a .QS suffix. It serves to define the relationship between the alias names in the virtual model and the actual names in the specific equipment units. The Equipment List Builder generates object code that is used with a generic CL sequence program and a generic schematic to control and monitor the individual equipment units.

Preparation Rules

The rules that control the preparation of an ELB source file are closely related to those that control the writing of CL programs. This manual assumes that the user has been trained in the writing of CL programs and in the preparation of TDC 3000^X schematic displays.

This publication will not attempt to duplicate information that is the same for CL and ELB. For rules not covered here that are the same as for CL, refer to the appropriate CL reference manual (MC, PM, or APM).

Elements of the Equipment List source file

An Equipment List source file is made up of many of the same elements as a CL program. The following list of ELB elements indicates their similarities and differences with respect to CL:

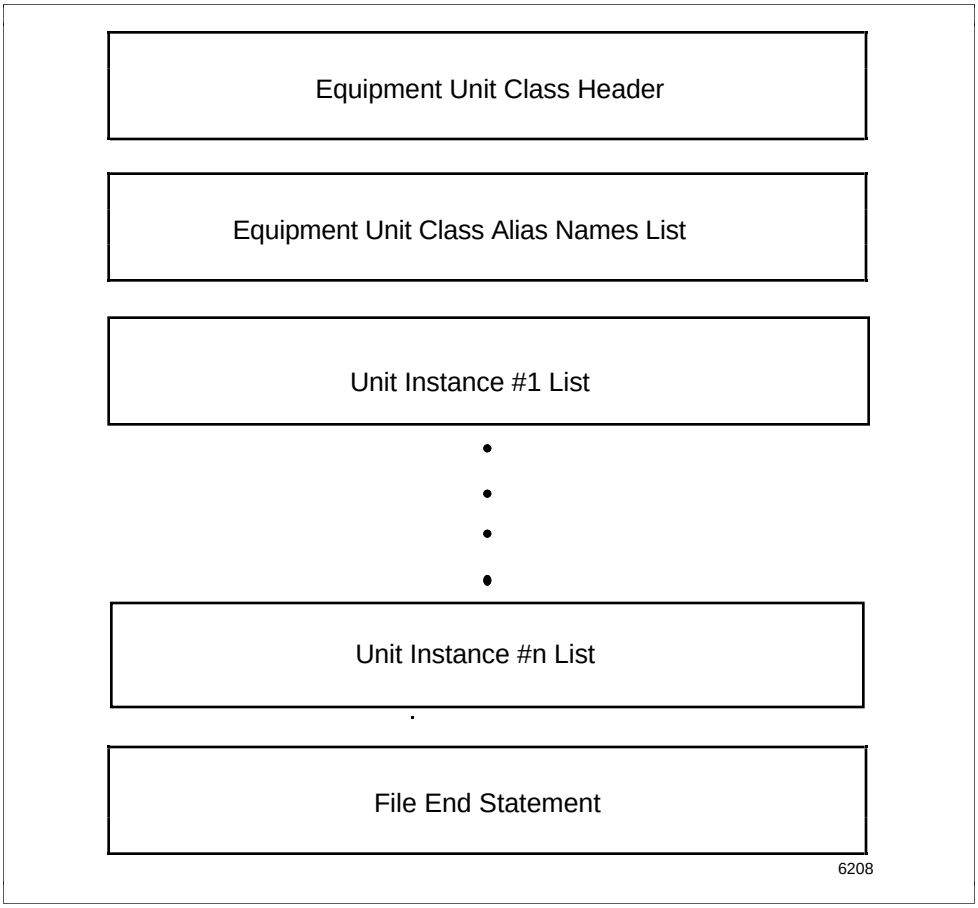
- **Characters**—The ELB character set is made up of the 95 printable ASCII characters (including blank). This is the same set used in CL—refer to a CL reference manual for details.
- **Lines**—Rules for lines are the same as for CL. This includes the use of "&" for continuation of lines, and the use of "--" to identify comments. Refer to a CL reference manual for details.
- **Reserved Words and Predefined Identifiers**—See 3.7.
- **Identifiers**—These are names that identify objects. See 3.6 for details.
- **Data Types**—See Table 3-1 and Table 3-2 in this publication.
- **Expressions**—Not used in ELB.
- **Statements**—The ELB source file uses special header and end statements. They will be defined in this publication. The ELB also uses definition statements (in the Alias Names list) and assignment statements (in the Unit Instance lists). They will also be covered in this publication.

3.2 Structure of the Equipment List Source File

Introduction

Figure 3-1 shows the structure of the Equipment List source file in block form. Each of the blocks will be covered in detail in the material that follows. We suggest that you refer to the ELB source file example in 1.3 as you read about the ELB source file structure.

Figure 3-1 Equipment List Source File Structure



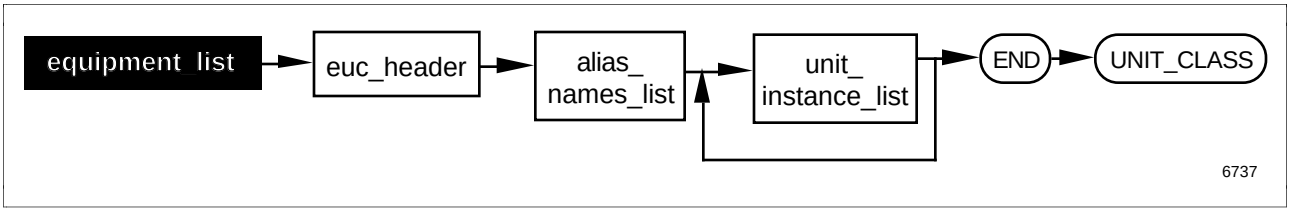
Continued on next page

Equipment List Syntax

An Equipment List source file consists of the

- **euc_header** (Equipment_Unit_Class header),
- followed by an **alias_names_list** (199 alias names maximum),
- followed by from 1 to 32 **unit_instance_lists**,
- and closes with the reserved words **END UNIT_CLASS**

Figure 3-2 Equipment List Syntax Diagram



3.3 Equipment Unit Class (EUC) Header

EUC Header Syntax

The EUC Header consists of the reserved word

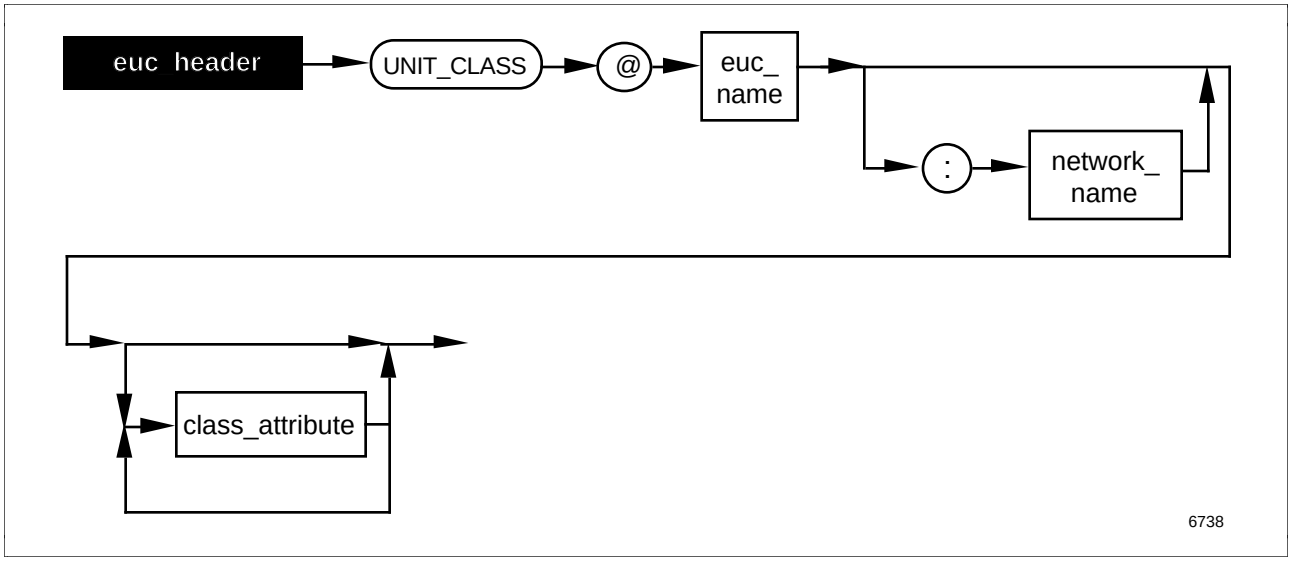
- **UNIT_CLASS**, followed by
- **@ euc_name**, followed by the optional
- **: network_name**, followed by optional
- **class_attributes**.

The euc_name must be same as the Equipment List source file's name. The network_name specifies whether the physical tags are located on a UCN or on a Data Hiway. If the network_name is not defined in the header (it is optional), the default is HWY.

The class_attributes are

- DDB—Display database index. This is required if the euc_name is to be used in a schematic. See 3.8.
- DESC—EUC descriptor. This is a string of up to 24 characters. It is reserved for future use.

Figure 3-3 EUC Header Syntax Diagram



EUC Header Example

The following shows an example of a Unit Class Header that might be used in an Equipment List source file REACTOR.QS.

```
UNIT_CLASS @reactor : UCN
  DDB(150)
  DESC "Reactor Phase of Batch"
```

Continued on next page

EUC Name Syntax

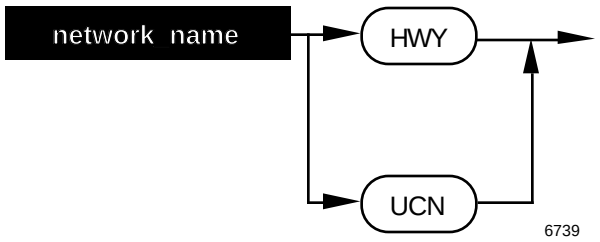
The euc_name (identifier)

- Must be the same as the name given to the Equipment List source file.
- Must be no longer than eight characters long. The "@" character is not included in the eight-character limit.
- Must follow the CL identifier rules (see 3.6).

Network Name Syntax

The network_name element defines whether the points being referenced are on a TDC 3000^X Universal Control Network (UCN) or on a Data Hiway (HWY). If no network_name is specified, HWY is assumed.

Figure 3-4 Network Name Syntax Diagram

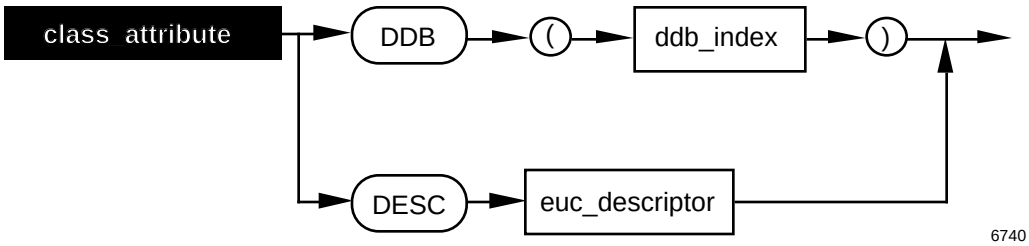


Class Attribute Syntax

The class_attributes that can be assigned to an EUC Header are

- DDB—Index into the display database. Required if the euc_name is to be used in a schematic. See 3.8.
- DESC—String of up to 24 characters that is reserved for future use.

Figure 3-5 Class Attribute Syntax Diagram



3.4 Equipment Unit Class (EUC) Alias Names List

Alias Names List Description

An alias_names_list consists of the reserved word

- **ALIAS**, followed by one or more
- **virtual_object_definitions**, and ending with the reserved words
- **END ALIAS**

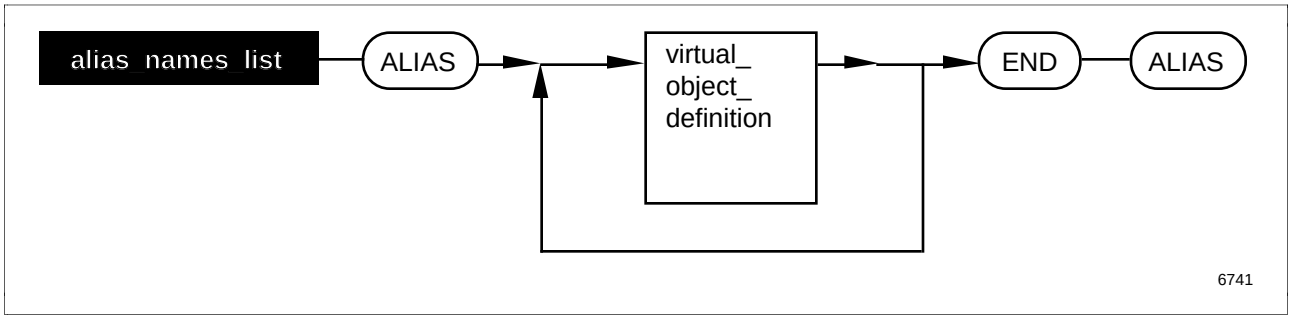
Alias Names List Example

```
ALIAS
  @PMDPA    : procmodl  --process module data point
    DESC “Bound Data Point”
  @LEVEL    : analgin   --analog input point
    DDB(200)
  @VALVE    : digin     --digital input point
  @BDP      : boxdp     --box data point
  @CAPACITY : number
    DDB(230, integer)
  @OFFSET               --type undefined—default is number
    DDB(240)
  @MESSAGE  : string
    STRING_SIZE 16      --the default string size is 8
    DDB(320)
  @VALVEPOS : logical
    DDB(300)
  @NEWPOS   : SDEnum
END ALIAS
```

Alias Names List Syntax

Figure 3-6 shows the syntax diagram for the Alias Names List.

Figure 3-6 Alias Names List Syntax Diagram



Continued on next page

Virtual_object_ definitions

A virtual_object_definition defines either a virtual tag (data point name) or a virtual constant. Each virtual tag or virtual constant definition consists of the

- **virtual_tag_name** or **virtual_constant_name**, followed by an optional
- **: type_expression**, and an optional
- **object_attribute**.

The virtual_tag_name or virtual_constant_name identifies the virtual tag or virtual constant. The maximum length is eight characters. The "@" character is not included in the eight-character limit. The maximum number of virtual (alias) names is 199.

The choices available for type_expression depend on whether the "object" is a tag name or a constant, and whether the data is located on a UCN or a Data Hiway. Table 3-1 and Table 3-2 show all possible type_expression assignments. If a type_expression is not included, "Number" is assumed.

The object_attribute is optional and shown in Figure 3-8.

Use of the type Entity_Id

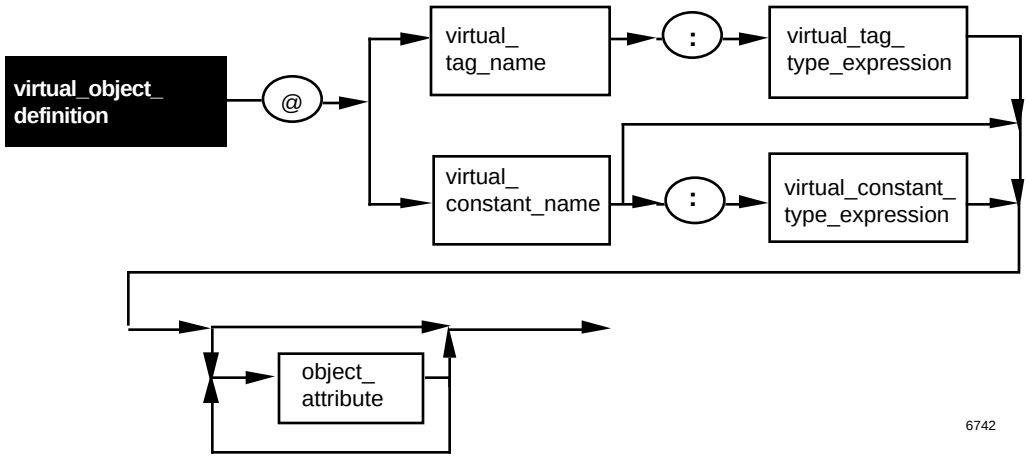
The type Entity_Id, included in Table 3-1, is general and can represent any type of data point. As an example, it could be used when an alias referenced in a schematic is a different type of point in various unit instances, such as an APM regulatory control point in one Unit Instance, and an AM regulatory control point in a second Unit Instance. For CL sequences, however, aliases would normally use one of the specific types listed in Table 3-1.

Continued on next page

Elements of virtual_ object_definitions

Figure 3-7 shows the syntax diagram of the virtual object definition, and indicates that the virtual object can be either a virtual tag name or a virtual constant name. Table 3-1 shows the types available for virtual tags, and Table 3-2 shows the types available for virtual constants.

Figure 3-7 Virtual Object Definition Syntax Diagram

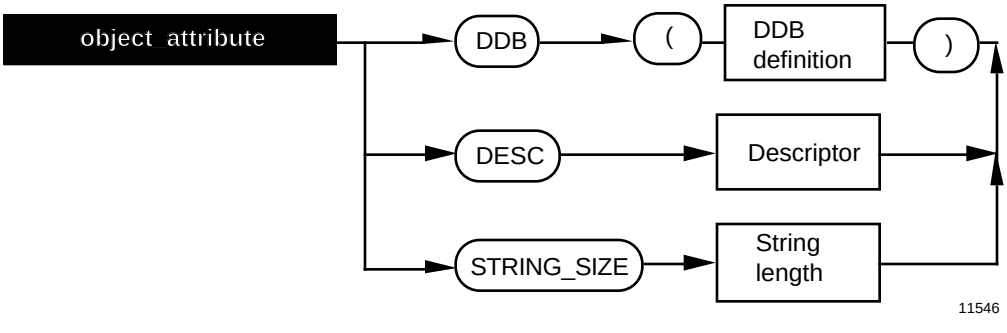


Object_attribute

The object_attributes that can be assigned to a virtual object are:

- DDB—Index into the display database. Required if the virtual name is to be used in a schematic. See 3.8.
- DESC—String of up to 24 characters that is reserved for future use.
- STRING_SIZE—Can only be used if the virtual name is a string type constant. String length must be in the range 1-64.

Figure 3-8 Object Attribute Syntax Diagram



Continued on next page

Type_expressions for virtual objects

Table 3-1 shows the types that can be used with virtual tags, and Table 3-2 shows the types that can be used with virtual constants. If no type is specified, the default type Number is assumed. In CL, Number is always a real. In schematics, Number can be forced to real or integer—see 3.8.

Table 3-1 Type Expressions Available for Virtual Tags

Type_ID	Description	UCN	Hwy
AnalgCom	Analog Composite data point		X
AnalgIn	Analog Input data point	X	X
AnalgOut	Analog Output data point	X	X
Array	Array point	X	
Boxdp	Box data point	X	X
Counter	Counter data point		X
DevCtrl	Device Control Point	X	
DigCom	Digital Composite data point	X	X
DigIn	Digital Input data point	X	X
DigOut	Digital Output data point	X	X
Entity_Id	Any type of data point	X	X
Flag	Flag data point	X	X
LogicBlk	Logic Block or Logic data point	X	X
Numeric	Numeric data point	X	X
Procmodl	Process Module data point	X	X
RegCtl	Regulatory control data point	X	
Reglatry	Regulatory data point		X
RegPV	Regulatory PV data point	X	
Timer	Timer data point	X	X

Table 3-2 Type Expressions Available for Virtual Constants

Type_ID	Description	UCN	Hwy
Number	Real constant (default type)	X	X
String	String constant	X	X
Logical	Logical constant	X	X
SDEnum	Enumeration constant	X	X

3.5 Unit Instance List

Unit Instance List Definition

You may have a maximum of 32 Unit Instance Lists in an Equipment List source file. Each Unit Instance List begins with the reserved word

- **UNIT_INSTANCE**, followed by
- **@ euc_name**, followed by
- **= " unit_instance_name"**, followed by one or more
- **@ virtual_name = instance_value**, and concluding with

  END UNIT_INSTANCE

The euc_name must be same as the Equipment List source file's name.

The unit_instance_name must be eight characters or less, not including the "@" character. Space characters must not be used. The unit_instance_name identifies the specific Unit Instance (Equipment Unit) whose value definitions follow in the Unit Instance list. It is used with the Picture Editor, in conjunction with the new EQ_LIST actor, to identify the specific Equipment Unit whose values you wish to display in a generic schematic (see 6.4).

Each virtual_name identifies either a tagname or a constant previously defined in the Alias Names List. Virtual_names must be listed in the same order that they appear in the Alias Names List, and none may be omitted.

Each instance_value corresponds with the virtual_name preceding, must be of the type identified in the Alias Names List, and gives the actual point name or value for the instance.

Tag names may be 8 or 16 characters

Tag names can be 8 characters maximum, or 16 characters maximum, depending on whether the SHORT or LONG tag name option is selected in the TAG NAME OPTIONS menu, under the SYSTEM WIDE VALUES menu, which is accessed from the ENGINEERING PERSONALITY MAIN MENU.

Points on other LCNs

Tag names cannot reference points on other LCNs.

String truncation

In the Alias Names List, a virtual name can be defined to be of type string. The string length can be specified, or will default to eight characters. (See the Alias Names List Example in 3.4.) In the Unit Instance List, a string constant is assigned to the virtual name. If the string constant is longer than the string length defined in the Alias Names List, the ELB will truncate the name and will include a warning message in the listing file.

Continued on next page

Box point

A box data point is valid in an Equipment List, but can be used only in CL programs. Box data points cannot be used in schematics. The ELB includes a warning message in the listing file when a box point is specified.

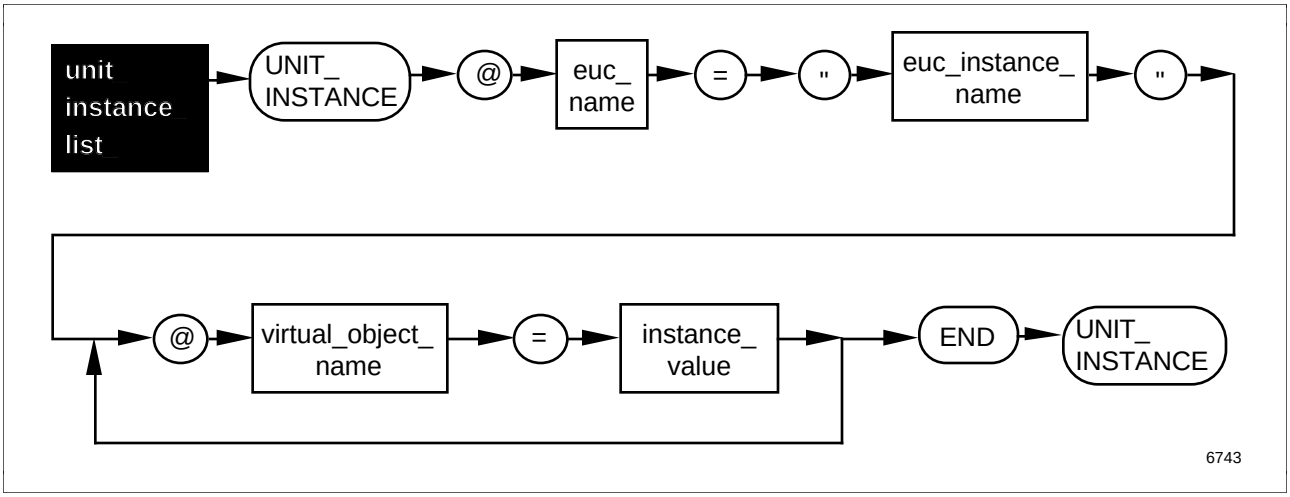
Unit Instance List Example

```
UNIT_INSTANCE @reactor = “reactor2”
  @PMDPA     = PMDP60
  @LEVEL     = LVL21
  @VALVE     = !DI06S01    --Cannot use in schematics
  @BDP       = $NM01N03    --!Box name form also supported
  @CAPACITY  = 1000
  @OFFSET    = 1.5E-2
  @MESSAGE   = “Unit 2 PM Point”
  @VALVEPOS  = ON
  @NEWPOS    = OPEN
END UNIT_INSTANCE
```

Unit Instance List Syntax

Figure 3-9 shows the syntax diagram of the Unit Instance List.

Figure 3-9 Unit Instance List Syntax Diagram



3.6 Identifiers

Identifiers Definition

Identifiers are used as the names of objects in ELB, just as they are in CL. Some of the identifiers used in ELB are:

- Equipment Unit Class Names
- Predefined identifiers of ELB (see the next page)
- Alias (Virtual) Names
- Physical Tag Names
- Logical states (ON/OFF)
- SD ENUM states

Identifiers Rules

ELB identifiers are composed of alphabetic characters (A to Z and a to z), numeric characters (0 to 9), the dollar sign ($), the exclamation point (!), and the break (underscore/underbar) character (_).

The builder also will accept a single apostrophe (’) as the first character of an identifier. This enables you to use a reserved word (such as ’END or ’UCN) as an identifier. See the Heading “Reserved Words and Predefined Identifiers” for more information.

An identifier cannot be a single-digit numeric.

There can be no spaces between characters; however, a break character can be used to divide a long identifier to make it easier to read. A break character cannot be the first nor the last character of an identifier. An identifier cannot contain two adjacent break characters.

Except in strings, there is no distinction between uppercase and lowercase alphabetic characters; that is, each lowercase alphabetic character is considered the same as its uppercase counterpart.

In strings, the case is preserved. A lowercase string “fred” is not the same as an uppercase string “FRED.”

3.7 Reserved Words and Predefined Identifiers

Introduction

Reserved words are predefined by ELB and cannot be redefined or used for other than their reserved meanings.

Predefined Identifiers are not reserved and can be redefined in a program. However, we recommend that you avoid redefining any Predefined Identifiers to avoid possible confusion.

Reserved Words

The ELB Reserved Words are:

- ALIAS
- DDB
- DESC
- END
- ENTITY_ID
- HWY
- STRING_SIZE
- UCN
- UNIT_CLASS
- UNIT_INSTANCE

Predefined Identifiers

The ELB Predefined Identifiers are:

- AnalgCom
- AnalgIn
- AnalgOut
- Array
- Boxdp
- Counter
- Devctrl
- DigCom
- DigIn
- DigOut
- Entity_ID
- Flag
- Logical
- LogicBlk
- Number
- Numeric
- Procmodl
- RegCtl
- Reglatry
- RegPV
- SDEnum
- String
- Timer

3.8 The DDB Attribute

What is a DDB attribute?

The DDB attribute is an index into a schematic database. In Equipment List, it must be used with any alias name that will be used in a schematic.

DDB values

The DDB index value is an integer. Its available range is 100 to 32,767. However, to avoid conflicts, Honeywell has preassigned blocks of DDB values for various developers. For more information, see Appendix A of the *Actors Manual* in this binder.

Syntax

The syntax of the DDB attribute definition is:

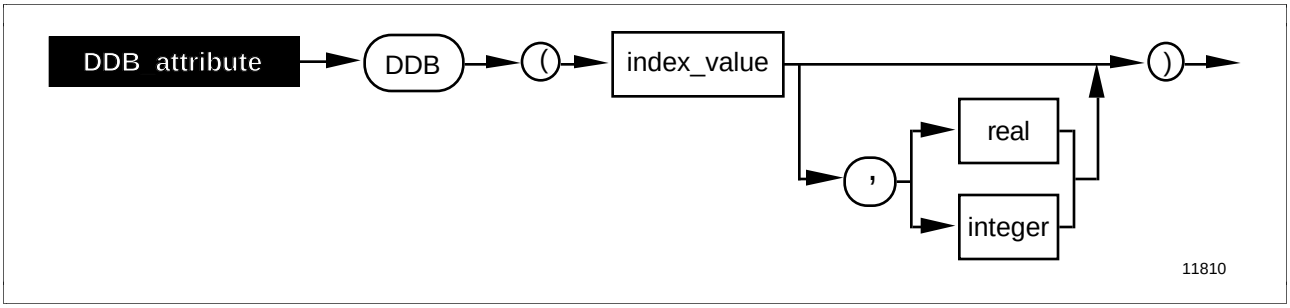
```
DDB(<index_value>)
```

If the DDB is associated with an alias of type NUMBER, there are alternate forms of the DDB definition that allow you to force the alias type to INTEGER or REAL. This allows you to specify whether an integer or a real is displayed in your schematic. The default type is REAL.

```
DDB(<index_value>,INTEGER)

DDB(<index_value>,REAL)
```

Figure 3-10 DDB Attribute Syntax Diagram



Continued on next page

DDB storage type

The DDB is organized by DDB storage type, which in turn is related to the alias types. The following table shows the relationship:

Alias Type	DDB Storage Type
Number (Real)	REAL
Number (Integer)	INTEGER
String	STRING
Logical	LOGICAL
SDEnum	STRING
Entity (tag name)	STRING

DDB index assignment

The index values for one DDB storage type can be the same as for another DDB storage type (duplication is allowed). For example, the following two examples are valid and do not conflict:

```
@VALUE1
  DDB(301, INTEGER)    -- Integer 301 (DDB storage type for
                              -- EUC header is string)

@VALUE2
  DDB(301, REAL)       -- Real 301 (valid—not duplicate)
```

On the other hand, the following examples are not valid because each requires the storage type STRING and they would, therefore, all try to use the same DDB storage space:

```
  UNIT_CLASS @REACTOR  :  UCN
    DDB(301)           -- string 301

ALIAS                  --begin alias section

  @STR1  :  string
    DDB(301)           --string 301 (invalid—duplicate)

  @ENUM1  :  SDEnum
    DDB(301)  :        -- string 301 (invalid—duplicate)

END ALIAS
```

In the above examples, different DDB index values are required for the three DDB attributes because each requires DDB storage type string.

Continued on next page

More examples of usage

Following are some additional examples of DDB usage:

```
UNIT_CLASS @reactor : UCN
  DDB(300)             -- applies to alias “@reactor”

@Valve : Digin
  DDB(300)             -- typical usage (entity)

@LEVEL : NUMBER
  DDB(300)             -- @LEVEL defaults to type REAL

@WHICH : NUMBER
  DDB(300,INTEGER)     -- forces @WHICH to type INTEGER

@WHICH : NUMBER
  DDB(301,REAL)        -- legal but unnecessary because
                          -- default is REAL
```

Section 4 – Building the Equipment List

4.1 Overview

Equipment List Builder

The Equipment List must be built before it can be used in generic CL sequences and generic schematics. The Equipment List Builder is used to build Equipment List object files from Equipment List source files. It is similar to the CL Compiler in many respects. This section will describe its use.

Build points first

Prior to building the Equipment List, all entities (points) used in the Equipment List source file must be built and loaded into the appropriate devices (PMs, APMs, MCs, and AMCs). One of the functions of the ELB is to verify that real entities referenced in the Unit Instances exist in the system. The ELB will generate an error for any referenced point that does not exist.

Files created

The Equipment List source file must have a suffix of .QS. If the ELB build is successful (no errors), the builder will produce a listing file of the same name, but with a suffix of .QL, and it will produce an object file with the same name, but with a suffix of .QO. If the ELB detects errors, it will not produce the .QL and .QO files. It will produce a listing file that includes error statements. This error file has the suffix .QE.

For example, an ELB compile of the file TANK.QS will result in:

TANK.QL and TANK.QO is there were no errors, or

TANK.QE if there were errors.

The files produced by the builder (.QL and .QO, or .QE) are placed in the same directory as the .QS file.

Break key

You can break (abort) out of an Equipment List Builder build with the Break Key. To activate the Break Key function, hold down the CTL key and press the COMND key on the Engineer's Keyboard. ("Break" is printed on the front surface of the COMND key.) If you break out of a build, no object or listing files are created.

4.2 Invoking the Equipment List Builder

Starting the builder

At a TDC 3000X Universal Station, execute the following steps.

Step	Action
1	Invoke the Engineering Main Menu.
2	Select SUPPORT UTILITIES.
3	Select MODIFY VOLUME PATHS.
4	Enter the name of the directory that contains your Equipment List source file(s) in the USER DEFAULT PATH box.
5	Return to the Engineering Main Menu.
6	Select COMMAND PROCESSOR.
7	Enter an ELB command on the command line. Syntax: **ELB <equipment_list_source_file_path>** You do not need to include the .QS suffix in the equipment_list_source_file path. It is optional. Note: If the directory containing Equipment list source files is not as specified by the "User Default Path," you will need to specify the full path name of the source file.
8	Press [ENTER] to begin the build.

Example of an ELB command

To build the Equipment List from the source file TANK.QS, assuming that you have correctly set the USER DEFAULT PATH, you would simply enter the command:

```
ELB   TANK
```

Builder Output Messages

Assuming that you have a source file named TANK with three Unit Instances, a successful ELB build will result in messages like these appearing on the Command Processor display.

```
ELB   TANK
Elapsed time for Alias                    =   6.8 seconds
Elapsed time for UnitInstance TANK1       =   1.6 seconds
Elapsed time for UnitInstance TANK2       =   1.6 seconds
Elapsed time for UnitInstance TANK3       =   1.6 seconds
Elapsed time for Listing                  =  17.6 seconds
Elapsed time for Compilation              =  30.1 seconds
     New ELB object file :
          File TANK.QO created
  Total errors detected : None
```

4.3 Example of an ELB Listing File

Sample listing ELB listing file for the example in Section 1 (see 1.3).

```
ELB V40.6  TANK                              06/03/92  10:28:48:0490   Page    1

Line  Loc  Text

 1         UNIT_CLASS  @TANK  :  UCN       -- ALIAS TANK DESCRIPTION
 2          ALIAS
 3            @PMDP    :  PROCMODL         -- ALIAS PROCESS MODULE DATA POINT
 4            @VALVE   :  DIGCOM           -- ALIAS FLOW VALVE
 5               DDB(300)
 6            @TOTALIZ :  REGPV            -- ALIAS TOTALIZER POINT
 7               DDB(301)
 8            @LIMIT   :  NUMBER           -- ALIAS TARGET LEVEL FOR TANK
 9               DDB(302)
10            @WHICH   :  NUMBER           -- IDENTIFIES WHICH TANK
11               DDB(303,INTEGER)
12          END ALIAS
13
14          UNIT_INSTANCE @TANK = "TANK1" -- SPECIFIC DESCRIPTION FOR TANK1
15            @PMDP     = FILL1            -- TANK1 PROCESS MODULE DATA POINT
16            @VALVE    = VALVE1           -- TANK1 FLOW VALVE
17            @TOTALIZ  = TOTALIZ1         -- TANK1 TOTALIZER POINT
18            @LIMIT    = 100              -- TANK1 FILL LIMIT
19            @WHICH    = 1                -- NUMBER IDENTIFIES TANK 1
20          END UNIT_INSTANCE
21
22          UNIT_INSTANCE @TANK = "TANK2" -- SPECIFIC DESCRIPTION FOR TANK2
23            @PMDP     = FILL2            -- TANK2 PROCESS MODULE DATA POINT
24            @VALVE    = VALVE2           -- TANK2 FLOW VALVE
25            @TOTALIZ  = TOTALIZ2         -- TANK2 TOTALIZER POINT
26            @LIMIT    = 200              -- TANK2 FILL LIMIT
27            @WHICH    = 2                -- NUMBER IDENTIFIES TANK 2
28          END UNIT_INSTANCE
29
30          UNIT_INSTANCE @TANK = "TANK3" -- SPECIFIC DESCRIPTION FOR TANK3
31            @PMDP     = FILL3            -- TANK3 PROCESS MODULE DATA POINT
32            @VALVE    = VALVE3           -- TANK3 FLOW VALVE
33            @TOTALIZ  = TOTALIZ3         -- TANK3 TOTALIZER POINT
34            @LIMIT    = 300              -- TANK3 FILL LIMIT
35            @WHICH    = 3                -- NUMBER IDENTIFIES TANK 3
36          END UNIT_INSTANCE
37
38        END UNIT_CLASS
39
```

Continued on next page

4.3 Example of an ELB Listing File, Continued

Sample listing , continued

Continuation of the ELB listing file for the example in Section 1.

```
*******  NO ERRORS DETECTED
    New EL object file:
           File TANK.QO  created

OPTIONS SELECTED  :  NONE

TIME STAMPS
-----------
Display Compatibility   :  06/03/92  10:28:48:0490
Program Compatibility   :  06/03/92  10:28:48:0490
Equipment List Source   :  06/01/92  11:15:35:0000
Unit Instance TANK1     :  06/03/92  10:28:48:0490
Unit Instance TANK2     :  06/03/92  10:28:48:0490
Unit Instance TANK3     :  06/03/92  10:28:48:0490
```

Time stamps

There are four kinds of time stamps appearing at the end of the ELB listing:

- Display Compatibility—indicates when alias names, their data types, or their DDB locations were created or last changed. Custom schematics that were compiled with this Equipment List object loaded, are impacted when this time stamp changes.
- Program Compatibility—indicates when alias names or their data types were created or last changed. CL programs that were compiled with this Equipment List object included, are impacted.
- Equipment List Source—indicates when the Equipment List source file was created or last modified.
- Unit Instance—indicates when an individual Unit Instance was first created or last modified. There will be one time stamp for each Unit Instance.

Section 5 – Using Equipment List in CL

5.1 Overview

Introduction

As explained by example in Section 1, Equipment List is used with generic CL sequence programs so that you can write a single source program that can be used to control a number of similar Equipment Units. This section explains how to compile generic CL sequences that include Equipment List object files.

Preparation

Before writing a generic CL sequence program, you must perform the following steps:

Step	Action
1	Prepare a virtual model (refer to 1.2).
2	Assign alias names to the virtual tags and constants.
3	Prepare the Equipment List source file.
4	Build the real points in the system.
5	Configure Equipment List support in each US that will be used to compile generic CLs (refer to 2.3).
6	Build the Equipment List (error-free).
7	Determine the control strategy (what the CL sequence will do).

5.2 Preparing a Generic CL Source File

Including the Equipment List

In the sequence program source file, include the Equipment List object file in the first line that is not a comment line. The syntax is as follows:

```
%INCLUDE_EQUIPMENT_LIST    <EL_Object_PathName>
```

You do not need to include the .QO suffix in EL_Object_PathName (it is optional). You may include a full pathname, or only the object file name. If you specify only the object name (TANK in the example), the User Default Path is used. If the full path is specified in the include statement, but you wish to use a different device and volume name at compile time, the Override EL Path (-OEP) switch may be used (see 5.4).

Use alias names

In the generic CL program source file, use the alias names (preceded by an @) that were defined in the Unit Class portion of the Equipment List source file. The alias names are used in place of the specific tag names and constants. Refer to the example below.

Example

The following example is the same as used in Section 1. It illustrates the use of alias names and the use of the %INCLUDE_EQUIPMENT_LIST directive.

```
%INCLUDE_EQUIPMENT_LIST TANK

SEQUENCE  FILL  (APM; POINT @PMDP)

EXTERNAL  @VALVE, @TOTALIZ

PHASE ONE

  SET   @TOTALIZ.COMMAND  =   STOP       -- INITIALIZE TOTALIZER
  SET   @TOTALIZ.COMMAND  =   RESET
  SET   @TOTALIZ.AVTV     =   @LIMIT     -- DESIRED TANK FILL AMOUNT
  SET   @VALVE.MODATTR    =   PROGRAM

  SEND: "CHARGING TANK WITH ",@LIMIT," GALLONS"

  SET   @TOTALIZ.COMMAND  =   START      -- START TOTALIZER
  OPEN  @VALVE                           -- BEGIN FILLING TANK
  WAIT  @TOTALIZ.AVTVFL   =   ON         -- WAIT FOR DESIRED FILL AMOUNT
  CLOSED   @VALVE                        -- CLOSE THE FILL VALVE
  SET   @TOTALIZ.COMMAND  =   STOP       -- AND STOP THE TOTALIZER

  SEND: "TANK CHARGED WITH ",@LIMIT," GALLONS"

END FILL
```

5.3 Compiling a Generic CL Source File

Configure EL support

You must configure Equipment List support on the US that you will use to compile the generic CL. See Section 2.

Set default paths

Compiling a generic CL sequence program is exactly the same as compiling a program that is not generic. After preparing the CL source file, the next step would normally be to set up default paths for CL SOURCE/OBJ, and USER DEFAULT PATH (for ELB objects). From the Engineering Main Menu, select SUPPORT UTILITIES, and then select MODIFY VOLUME PATHS.

Invoking the compiler

The compiler is invoked from the Command Processor by entering the following command:

```
CL   <filename>   [-<option  switch>]...
```

For further information on the compiler operation, including setting pathnames and the use of compiler command options (switches), refer to the appropriate Data Entry manual (PM, APM, or MC). Three command options have been added—they are covered in this manual (see 5.4) as well as in the respective CL Data Entry manuals.

Multiple objects

When a generic CL program is compiled, the CL Compiler generates multiple object files—one for each Equipment Unit. This can be observed by watching the messages that appear on the Command Processor display during the compile. You will see multiple executions of Pass 2 and Pass 3. The following example shows the Command Processor display as the generic CL program TANK.CL is compiled.

```
CL  TANK   -UL
Elapsed time for Pass 1       =       3.7 seconds
Elapsed time for Pass 2       =       3.1 seconds
Elapsed time for Pass 3/APM   =       4.1 seconds
Elapsed time for Pass 2       =      10.9 seconds
Elapsed time for Pass 3/APM   =       5.8 seconds
Elapsed time for Pass 2       =       3.4 seconds
Elapsed time for Pass 3/APM   =       4.1 seconds
Elapsed time for Listing      =      27.6 seconds
Elapsed time for Compilation  =      63.1 seconds
      Object File 30518994.NO for Instance TANK1    created
      Object File 30519994.NO for Instance TANK2    created
      Object File 30520994.NO for Instance TANK3    created
Errors detected : None
          Notes : 1
```

Continued on next page

Listing file changes

There are a number of differences in the listing (.LS) file when generic CL is compiled:

- If an error exists that affects one or more, but not all, Unit Instances, there will be an error message in the listing for each Unit Instance that is affected. If an error exists that affects all Unit Instances, there will only be a single error message in the listing.
- Alias names used in the CL program will be included in the XREF listing if you request one to be generated by the compiler.
- The listing includes the object file names for each Unit Instance, and the size of each object file.
- The listing includes the following time stamps:
  - CL Source Program
  - Equipment List Source
  - Program Compatibility
  - A time stamp for each object file.
- The Equipment List Object pathname used is specified. This could be the pathname specified in the %INCLUDE_EQUIPMENT_LIST directive, the pathname specified by the User Default Path, or the pathname specified on the command line if you used the Override Equipment List Path option (see 5.4) when compiling.
- The listing includes a list of switches used, including the new switches described in 5.4.

5.4 New CL Compiler Command Options

Three new options

There are three new options in the CL Compiler to support generic CL. The following table shows their availability for AM, MC, AMC, PM, and APM:

	Compiler Option	AM	MC	AMC	PM	APM
–OEP	Overwrite EL Path	N	Y	Y	Y	Y
–UI	Unit Instance	N	Y	Y	Y	Y
–OPT	Optimize	N	N	N	Y	Y

The –OEP option

Override Equipment List Path—The –OEP (or –OVRWRTELP) option overrides the device name and volume name specified in a %INCLUDE_EQUIPMENT_LIST statement and/or the User Default Path. This option does not include and cannot override the object file name. Form:

```
-OEP    <device>volume>
```

Examples:

```
-OEP    NET>TEST
-OEP    $f1>ELB
```

The –UI option

Unit Instance Only—The –UI (or –UNITINST) option causes only the specified Unit Instance of an included Equipment List to be compiled. Form:

```
-UI    <unit_instance_name>
```

If no errors are detected during compilation of the selected Unit Instance:

- A listing (.LS) file, and an object (.PO, .NO, or .MO) file for that Unit Instance, are generated. Any existing listing file, and object file for that Unit Instance, are overwritten.
- No error listing (.LE) file is generated and any existing .LE file is deleted.

If any error is detected during the compilation of the selected Unit Instance:

- An error listing (.LE) file is generated that includes error information only for that Unit Instance. If a .LE file already exists, it is overwritten.
- No listing (.LS) or object (.PO, .NO, or .MO) files are generated; thus, any existing listing or object files are not replaced.

Continued on next page

The –OPT option

Optimize Object Generation—The –OPT (or –OPTIMIZE) option is used to reduce compilation time. When a generic CL is compiled with the –OPT option, the compiler will compile only those Unit Instances that require compilation. Note: The –OPT option is not available for the Multifunction Controller.

When the –OPT option is specified, the compiler determines, for each Unit Instance, whether a new object needs to be generated. It does this by checking time stamps in the CL source file, the Equipment List object file, and the CL object files for each Unit Instance.

It will compile a Unit Instance if:

- **The CL source file has changed.** The time stamp has changed since the last compile.
- **The Equipment List object file has changed for the Unit Instance**. The time stamp for a specific Unit Instance in the Equipment List object file has changed from the value recorded in the CL object file for that Unit Instance.
- **The CL object for the Unit Instance has changed**. The CL object file time stamp for the Unit Instance has changed from the value recorded in the Equipment List object file since the last compile. This could be caused by deleting the object file or by copying an older version into the CL SOURCE/OBJ directory.
- **The Equipment List object file has changed**. The time stamp for program compatibility in the Equipment List object file has changed from the value recorded in the CL object files. All Unit Instances will be compiled even if the –OPT option is used.

If no errors are detected during compilation of one or more Unit Instances:

- A generic CL listing (.LS) file is generated and the information for all Unit Instances is included. Existing .LS file is overwritten.
- No error listing (.LE) file is generated and any existing .LE file is deleted.

If an error is detected during compilation of one or more Unit Instances:

- An error listing (.LE) file is generated that includes error information for those Unit Instances. If a .LE file already exists, it is overwritten.
- No listing (.LS) or object (.PO or .NO) files are generated; thus, any existing listing or object files are not replaced.

5.5 Loading Generic CL

Copy the files to &Enn

Compiling a generic CL program without errors results in a CL object (executable) file for each Unit Instance defined in the Equipment List that is included in the CL source. Conceptually, the CL object files that you obtain using Equipment List in generic CL are the same object files that you would obtain if you did a separate compile without Equipment List for each Unit Instance, using the specific tagnames and constants for each Unit Instance.

All of the object files generated by the compile will be in the same volume/directory as the CL source file. These object files must be copied or moved to your &Enn volume/directory (nn is the HWY or UCN number). Refer to the appropriate Data Entry manual for information about creating and using this volume/directory.

Load the CL

Call up the Detail Status display for the Process Module Data Point that you specified in each Unit Instance in the Equipment List source file. Load the sequence program in each.

5.6 Example of a CL Listing File

Sample Listing CL Listing file for the example in Section 1 (see 1.4).

```
CL V40.30    TANK                                06/03/92 16:08:35:4663  Page  1

Line  Loc   Text

   1        %INLUDE_EQUIPMENT_LIST TANK
   2
   3        SEQUENCE  FILL  (APM; POINT @PMDP)
   4
   5        EXTERNAL  @VALVE, @TOTALIZ
   6
   7        PHASE ONE

**NOTE  **  All enabled abnormal handlers are disabled

   8
   9    1     SET  @TOTALIZ.COMMAND   =   STOP   -- INITIALIZE TOTALIZER
  10    2     SET  @TOTALIZ.COMMAND   =   RESET
  11    3     SET  @TOTALIZ.AVTV      =   @LIMIT -- DESIRED TANK FILL AMOUNT
  12    4     SET  @VALVE.MODATTR     =   PROGRAM
  13
  14    5     SEND: "CHARGING TANK WITH ",@LIMIT," GALLONS"
  15
  16    6     SET  @TOTALIZ.COMMAND   =   START  -- START TOTALIZER
  17    7     OPEN @VALVE                        -- BEGIN FILLING TANK
  18    8     WAIT @TOTALIZ.AVTVFL    =   ON     -- WAIT FOR DESIRED FILL AMOUNT
  19    9     CLOSED @VALVE                      -- CLOSE THE FILL VALVE
  20   10     SET @TOTALIZ.COMMAND    =   STOP   -- AND STOP THE TOTALIZER
  21
  22   11     SEND: "TANK CHARGED WITH ",@LIMIT," GALLONS"
  23
  24        END FILL
  25

********  No errors detected
```

Continued on next page

5.6 Example of a CL Listing File, Continued

Sample listing, continued

Continuation of the CL listing file for the example in Section 1.

```
       6 Blocks of object code out of a maximum of   392 blocks
     Object File 30518994.NO for instance TANK1    oreated
       6 Blocks of object code out of a maximum of   392 blocks
     Object File 30519994.NO for instance TANK2    oreated
       6 Blocks of object code out of a maximum of   392 blocks
     Object File 30520994.NO for instance TANK2    oreated

OPTIONS SELECTED : -UPDATELIB
    Equipment List Object : NET>JIMK>TANK

TIME STAMPS:
CL Source Program       : 06/04/92 13:21:01:0000
Equipment List Source   : 06/01/92 11:15:35:0000
Program Compatibility   : 06/03/92 10:28:48:8730
Unit Change  TANK1      : 06/03/92 10:28:48:8730
Unit Change  TANK2      : 06/03/92 10:28:48:8730
Unit Change  TANK3      : 06/03/92 10:28:48:8730
```

Section 6 – Using Equipment List in Schematics

6.1 Overview

Introduction

As explained by the example in Section 1, Equipment List is used with generic schematics so that you can build a single schematic that can be used to monitor and control a number of similar Equipment Units. This section explains how to build generic schematics that include Equipment List object files.

Preparation

Before building a generic schematic, you must perform the following steps:

Step	Action
1	Prepare a virtual model (refer to 1.2).
2	Assign alias names to the virtual tags and constants.
3	Prepare the Equipment List source file. **IMPORTANT**—Remember that all alias names that will be used in schematics require a DDB definition.
4	Build the real points in the system.
5	Configure Equipment List support in each US that will be used to build, compile, or display generic schematics.
6	Build the Equipment List (error-free).

CAUTION

An alias name cannot be used with an actor or collector that requires a specific entity id as an argument. For example:

```
if ACKSTAT(tag1) = UNAKALRM then s red blink
    else if ACKSTAT(tag1) = AKDALRM then s red no blink
```

In this example, tag1 must be an actual point built on the system—it cannot be an alias name.

6.2 Introduction

Purpose of generic schematics

In a generic schematic, you use alias names in the display in place of actual tag names and constant names. For example, you use alias names when you define parameter values to be displayed. A new Picture Editor runtime actor can be used with initial and normal targets. It allows you to select the actual tag names and constants from a particular Unit Instance and use them in place of the alias values. Using this technique, you can use a single generic schematic with a number of similar Equipment Units. Refer to the simple example in 1.5.

Alias names used in schematics do not include the @ character. This is different from CL usage.

Alias-named DDB

The alias names from Equipment List object files that are loaded into a schematic with the new LOADEQ command (see 6.3) are stored in an area of local DDB called alias-named DDB. If the new runtime actor EQ_List (see 6.4) is used in the initial target, the alias names in the alias-named DDB can be initialized with values from a specified Unit Instance. This means the schematic will display values from the specified Unit Instance when the schematic is first called up. (For information on the initial target, see the DEFINE command in the *Picture Editor Reference Manual*.)

The EQ_List actor can be put in normal targets. This allows the user to change the contents of alias-named DDBs by target action, thereby displaying values from user-selected Equipment Units.

Some limits

- A schematic can be loaded with a maximum of eight Equipment List object files.
- An Equipment List can have a maximum of 200 DDB definitions.

Continued on next page

Picture Editor search order

The id search order for the Picture Editor is as follows:

Order Searched	What is Searched
1	System DDB (Examples: INT01, STRING01)
2	User-defined DDB (This is defined in the .DF file and loaded by the LOAD command.)
3	Alias-named DDB (This is defined in the Equipment List object file and loaded by the LOADEQ command.)
4	System Entity (POINT ID.)

If you define the same name in the Equipment List and as a point id, the Picture Editor searches the alias-named DDB before the point ids are searched. Therefore, the Picture Editor cannot access this system entity name.

If you define the same name in the Equipment List as in the User-defined DDB, the Picture Editor displays an error and the EL object or the User Definition file will not be loaded.

6.3 New Picture Editor Commands

Three new commands

There are three new Picture Editor commands to support the Equipment List function. They are:

- **LOADEQ**—Load Equipment List. Brings the alias names of the Equipment List object file into the schematic. This allows generic (alias) names to be used in the schematic.
- **UNLOADEQ**—Unload Equipment List. Used to delete alias names of an Equipment List object file from the schematic.
- **LISTEQ**—List Equipment Lists. Lists the Equipment Lists already loaded to the schematic.

LOADEQ
Load Equipment List

Syntax: `LOADEQ nnnn` or `LDQ nnnn`
where nnnn is the full or partial pathname of the Equipment List object file. Refer to the Set Pathname command in the *Picture Editor Reference Manual* for acceptable forms of nnnn. The file is assumed to have the suffix of .QO, but this suffix is not included in the command.

Use: The LOADEQ command is used to load the alias names that have DDB locations defined in the Equipment List object file. These alias names are used in the same manner as local DDB names in the schematic. During building of a schematic, a maximum of eight different Equipment List object files can be loaded to the Picture Editor.

Examples:
```
LDQ NET>ELB>REACTOR
```
or
```
LDQ REACTOR
```

Error conditions:

- If eight Equipment List object files were already loaded to the Picture Editor and the user attempts to load a ninth object file, then an error message is displayed and the LOADEQ command is rejected.
- If the name of the Equipment List object file is the same as one that is already loaded (even if the pathname is different), the Picture Editor returns an error and the file is not loaded.
- If duplicate names among the Equipment List files and the user-defined DDB files are encountered, the Picture Editor returns errors and the LOADEQ command is rejected. If a duplicate name between the alias names of an Equipment List file and the Standard DDB is encountered, the Picture Editor uses the Standard DDB variable instead of the alias-named variable.
- If different alias names are assigned to the same location, the Picture Editor returns an error and the LOADEQ command is rejected.

Continued on next page

UNLOADEQ
Unload Equipment List

Syntax: `UNLOADEQ nnnn` or `ULDQ nnnn`
where nnnn is the name of an Equipment List object file that was previously loaded to the Picture Editor by the LOADEQ command.

Use: This command is used to unload (delete) the Equipment List object file that is specified by the argument, from the Picture Editor if it exists there. The alias-named local DDBs of the unloaded Equipment List object file cannot be referenced in the schematic after the UNLOADEQ command. Even if the alias names of the unloaded Equipment List object file are used in the schematic before they are unloaded, an error will be generated at compile time.

When the UNLOADEQ command is used with no argument, the Picture Editor responds with the following prompt:

All EQ files will be deleted. Press ENTER to continue/CANCEL to abort.

Pressing ENTER will delete all loaded Equipment List object files, while pressing CANCEL aborts the command.

Example: `ULDQ REACTOR`

Error condition: If the specified Equipment List object file does not exist in the Picture Editor, an error message is generated.

LISTEQ
List Equipment Lists

Syntax: `LISTEQ` or `LSQ`

Use: This command is used to list all the Equipment List object pathnames currently loaded to the Picture Editor. The list is displayed on the main editing display. Pressing the CANCEL key removes the displayed list.

Example: `LSQ`

LOAD command—new functions

The load command is used to specify user-defined local and global DDBs. This command can be used even if the user uses Equipment List alias-named DDBs in the schematic.

If the Equipment List object is already loaded when the LOAD command is used, the Picture Editor performs the following error checking for conflict:

- Checks for name conflict with alias-named DDBs already loaded.
- Checks for index conflict with alias-named DDBs.

If errors are found, they are displayed and the LOAD command is rejected. The reason for these checks is to anticipate runtime errors because alias-named DDBs are assigned values at runtime.

6.4 The EQ_LIST Actor

Purpose

This actor allows you to initialize or change the alias-named DDBs to values from a specific Unit Instance. This allows you to specify the Unit Instance whose values will be displayed when the schematic is invoked, and it allows you to use targets to switch to other Unit Instances without recalling the schematic.

Calling EQ_LIST

Calling Sequence:

```
EQ_LIST(equipment_list_object_filename, unit_instance_name)
```

Examples:

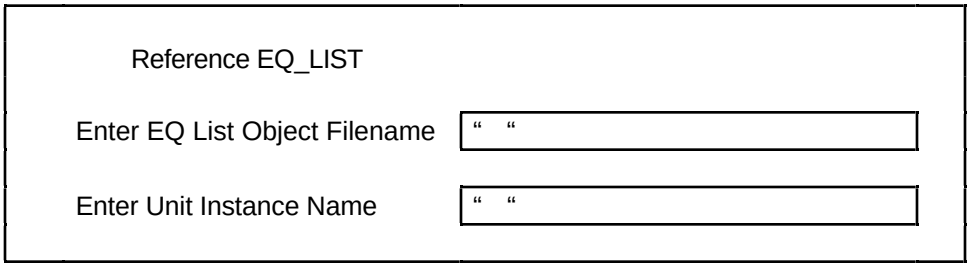
A string type global DDB is usually used to specify the Unit_Instance name if this actor is put in the initial target. The global DDB has to be set before invoking the generic schematic. An example of this usage is:

```
EQ_LIST("REACTOR", G_STR(STR01G))
```

You can also specify the Unit_Instance directly, as shown in this example:

```
EQ_LIST("REACTOR","REACTOR1")
```

If you enter the actor name without parameters, the Optional Parameter Entry Form will appear for you to enter data:



Parameters

The Equipment List object file name is the name that was used in the LOADEQ command to load the object file to the Picture Editor.

The Unit Instance name is the name of a specific Unit Instance in the Equipment List. This Unit Instance provides the specific tag names and constant names that are placed in the alias-named DDB, and therefore become the actual values displayed on the schematic.

Continued on next page

Configuring the pathname

When you call up a generic schematic, the schematic must have access to the Equipment List object file or files that were loaded when the schematic was built and compiled. You can put the Equipment List object file or files in the same volume/directory as the schematic object file. Alternatively, you can place the Equipment List objects in a different volume/directory, but that volume/directory must be defined in SCHEMATIC PATHNAMES in the Pathname Catalog in the Area Database. Note, however, that the search for Equipment List objects is done in reverse order. Therefore, for maximum performance, you should configure the schematic path at the beginning of the search list, and the Equipment List object file at the end of the list.

Using the actor

The EQ_LIST actor may be used in the initial target in order to initialize alias names in the schematic to values for a specific Unit Instance. This is highly recommended, because if the initial target is not used, the alias name values will indicate bad values when the schematic is invoked. For more information on the initial target, refer to the DEFINE command in the *Picture Editor Reference Manual* in this binder.

The actor may also be used in normal targets, allowing you to change the Unit Instance that is displayed without recalling the schematic. Refer to the simple example in 1.5.

The actor may not be used in button configuration.

For more information

For additional information on the EQ_LIST actor, refer to the *Actors Manual* in this binder. The material includes information on runtime error messages that can occur when using Equipment List with schematics.

Index

A

Actor Parameters *50*
Alias Names *2*
Alias Names List *20*
Alias Names List Syntax *20*
Alias-named DDB *46*

B

Break Key *31*
Building an Equipment List *31*

C, D

Class Attribute Syntax *19*
Compiling Generic CL *6*, *37*
Configuring for Equipment List *11*
    Each US must be changed *11*
    External load modules *12*
    Preparation *11*
    Procedure *13*
    Screen used to configure US *14*
    Which modules to configure *12*
DDB attribute *28*

E, F, G, H, I, J, K

EQ_LIST Actor *50*
Equipment List
    Definition *1*
    Introduction *1*
    What is the Equipment List function? *1*
    What the Equipment List function will do *1*
    When is it used? *1*
    Where is it used? *1*
Equipment List Builder
    Build Equipment List *4*
    Build points *4*
    Source file example *5*
    The ELB source file *4*
    What is the Equipment List Builder? *4*
Equipment List Source File
    Alias Names List
        Description *20*
        Elements of virtual_ object_definitions *22*
        Example *20*
        Object attribute syntax diagram *22*
        Object_attribute *22*
        Syntax *20*
        Syntax diagram *22*
        Type_expressions for virtual objects *23*
        Use of the type Entity_Id *21*
        Virtual_object_ definitions *21*
    Building
        Break key *31*
        Build points first *31*
        Builder Output Messages *32*
        Equipment List Builder *31*
        Example of an ELB command *32*
        Files created *31*
        Sample listing *33*
        Starting the builder *32*
        Time stamps *34*
    Elements of *15*
    Equipment Unit Class (EUC) Header *18*
        Class Attribute Syntax *19*
        Name Syntax *19*
        Network Name Syntax *19*
    Identifiers
        Definition *26*
    Introduction *15*
    Predefined Identifiers *27*
    Preparation Rules *15*
    Reserved Words *27*
    Structure Introduction *16*
    Syntax *17*
    The DDB Attribute
        DDB values *28*
        Syntax *28*
        What is a DDB attribute? *28*
    Unit Instance List
        Definition *24*
        Example *25*
        Rules *26*
        Syntax *25*
        Tag names *24*
Equipment List Syntax *17*

Index

Equipment List Use in CL
- Compiling Generic CL *6, 37*
  - Configure EL support *37*
  - Invoking the compiler *37*
  - Listing file changes *38*
  - Multiple objects *37*
  - Set default paths *37*
- Example *7, 36*
- Example of a CL Listing File
  - Sample Listing *42*
- Including the Equipment List *6, 36*
- Introduction *35*
- Load the CL *6*
- Loading Generic CL
  - Copy the files to &Enn *41*
- New CL Compiler Command Options
  - Load the CL *41*
  - Three new options *39*
  - –OEP option *39*
  - –OPT option *40*
  - –UI option *39*
- Preparation *35*
- Prepare a generic CL sequence program *6*
- Source file example *7*
- Use alias names *6, 36*

Equipment List Use in Schematics
- Actor Parameters *50*
- Alias-named DDB *46*
- Build the schematic *8*
- Calling EQ_LIST *50*
- Configuring the pathname *51*
- Example *8*
- Examples of generic schematic use *10*
- Introduction *45*
- List Equipment Lists *49*
- LISTEQ *49*
- LOAD command—new functions *49*
- Load Equipment List *48*
- Load the Equipment List *8*
- LOADEQ *48*
- Picture Editor search order *47*
- Preparation *45*
- Purpose of generic schematics *46*
- Set up the NCF *8*
- The EQ_LIST actor *8, 50*
- The generic schematic *9*
- Three new commands *8, 48*
- Unload Equipment List *49*
- UNLOADEQ *49*
- Use of alias names in schematic example *8*
- Use of the EQ_LIST actor in schematic example *9*
- Using the actor *51*
- Using the generic schematic *9*

Equipment Unit Class
- Definition *1*

Equipment Unit Class (EUC) Header *18*
Equipment Unit Class Header Syntax *18*

L, M

LISTEQ Command *49*
LOAD Command *49*
LOADEQ Command *48*

N

Network Name Syntax *19*

O

Object Attribute Syntax *22*
OEP option *39*
OPT option *40*

P, Q, R

Picture Editor Search Order *47*

S

Syntax
- Alias Names List *20*
- Class Attribute *19*
- Equipment List *17*
- Equipment Unit Class Header *18*
- Network Name *19*
- Object Attribute *22*
- Unit Instance List *25*
- Virtual Object Definition *22*

T

Time Stamps *34*

U

UI option *39*
Unit Instance List *24*
Unit Instance List Syntax *25*
UNLOADEQ Command *49*

V, W, X, Y, Z

Virtual Model
    Alias constant names in *2*
    Alias entity names in *2*
    An example *2*
    Illustration *3*
    Introduction *2*
Virtual Object Definition Syntax *21*

Index

READER COMMENTS

Honeywell IAC Automation College welcomes your comments and suggestions to improve future editions of this and other publications.

You can communicate your thoughts to us by fax, mail, or toll-free telephone call. We would like to acknowledge your comments; please include your complet name and address

BY FAX: Use this form; and fax to us at (602) 313-4108

BY TELEPHONE: In the U.S.A. use our toll-free number 1*800-822-7673 (available in the 48 contiguous states except Arizona; in Arizona dial 1-602-313-5558).

BY MAIL: Use this form; detach, fold, tape closed, and mail to us.

Title of Publication: **Equipment List Reference Manual** Issue Date: **9/95**

Publication Number: **SW27-560**

Writer: **Maria Nelson**

COMMENTS: _____

_____

_____

_____

_____

_____

_____

_____

RECOMMENDATIONS:_____

_____

_____

_____

_____

_____

_____

NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

TELEPHONE _____ FAX _____

(If returning by mail, please tape closed; Postal regulations prohibit use of staples.)

Communications concerning technical publications should be directed to:

Automation College
Industrial Automation and Control
Honeywell Inc.
2820 West Kelton Lane
Phoenix, Arizona 85023

FOLD FOLD

From: _____

_____

_____

NO POSTAGE
NECESSARY
IF MAILED
IN THE USA

BUSINESS REPLY MAIL

FIRST CLASS **PERMIT NO. 4332** PHOENIX, ARIZONA

POSTAGE WILL BE PAID BY ....

Honeywell

Industrial Automation and Control
2820 West Kelton Lane
Phoenix, Arizona 85023

Attention: Manager, Quality

FOLD FOLD

Cut Along Line

Additional Comments:

Honeywell

Industrial Automation and Control
Honeywell Inc.
16404 North Black Canyon Highway
Phoenix, Arizona 85023-3033

Helping You Control Your World