Honeywell

Interpret Database Structure

L61521
LCN

TotalPlant®

Notices and Trademarks

This module supports **TotalPlant** Solution (TPS) system network.

TPS is the evolution of TDC 3000X.

Honeywell Inc.
Industrial Automation and Control
Automation College
2820 West Kelton Lane
Phoenix, AZ 85053-3028
1-800 852-3211

TABLE OF CONTENTS

FIGURES AND TABLES

Acronyms Referenced Later In This Module

AM ..... Application Module
APM ..... Advanced Process Module
CDS ..... Custom Data Segment
CL ..... Control Language
CM ..... Computing Module
DEB ..... Data Entity Builder
DSD ..... Data Segment Description
EB ..... Exception Build
GDF ..... Global Definition Forms
GUS ..... Global User Station
HM ..... History Module
IDF ..... Independent Data File
LCN ..... Local Control Network
MC ..... Multifunction Controller
NCF ..... Network Configuration File
NG ..... Network Gateway
PM ..... Process Manager
PSDP ..... Process Status Data Point
UCN ..... Universal Control Network
US ..... Universal Station

MODULE INTRODUCTION

By understanding the global database structure of your system, you can make informed decisions in building your database. By understanding the concept of data binding, you will be able to minimize reconfiguration and troubleshooting efforts.

When a customer-built database is converted from a human-readable (external) form to a Honeywell system-usable (internal) form, a process called data binding occurs. For example, a tag name, such as “TIC100” is the human-readable form of the data we wish to see. At the time the point is loaded, an internal revision or serial number is assigned to that tag name. Thus, we can say that the tag name, TIC100, has been "bound" to an internal system number. Any system display that refers to TIC100 uses the internal system number, and not the tag name.

Module Objective

After completing this module, you should be able to optimize and interpret the LCN-distributed database structure. You should also be able to foresee the effects of changes to your database and more easily troubleshoot database configuration problems.

SYSTEM CONFIGURATION REVIEW (FIGURE 1)

As you will recall, system configuration is defined in the following sequence:

1. Build the LCN network configuration file to define the hardware configuration (nodes and peripherals), and the logical network configuration (units, consoles, logical node mapping to physical nodes).

2. Define the process network configuration to define the Hiway box/slots and the UCN device and point mix.

3. Define Hiway and UCN process points.

4. Define higher level points in the AM and CM. Custom data segments and CL programs should be linked at this time.

5. Finally, configure the history data organization on the HMs, and define the Area database entities, such as groups, logs, and schematics.

> **ATTENTION**
>
> Remember that any database change at a later time may affect configuration data at all levels subsequent to it.

SYSTEM CONFIGURATION TERMS AND CONCEPTS

Database Dependencies

The relationships or "data dependencies" can be conceptualized as shown in Figure 1. The key concept is that the database items in the upper portions of the figure have a dependency on the items in the lower portion of the illustration.

Managing Database Structure

Three concepts are key to managing database structure:

1. Knowing the structure of the database (architecture).

2. Knowing where a database entity "resides."

3. Using the first two concepts to determine what impact a change to the database will have.

Find Names Utility

A utility called "Find Names" exists that can be used to locate references to a database entity in the system, so that the user can know ahead of time the possible consequences of changing the database.

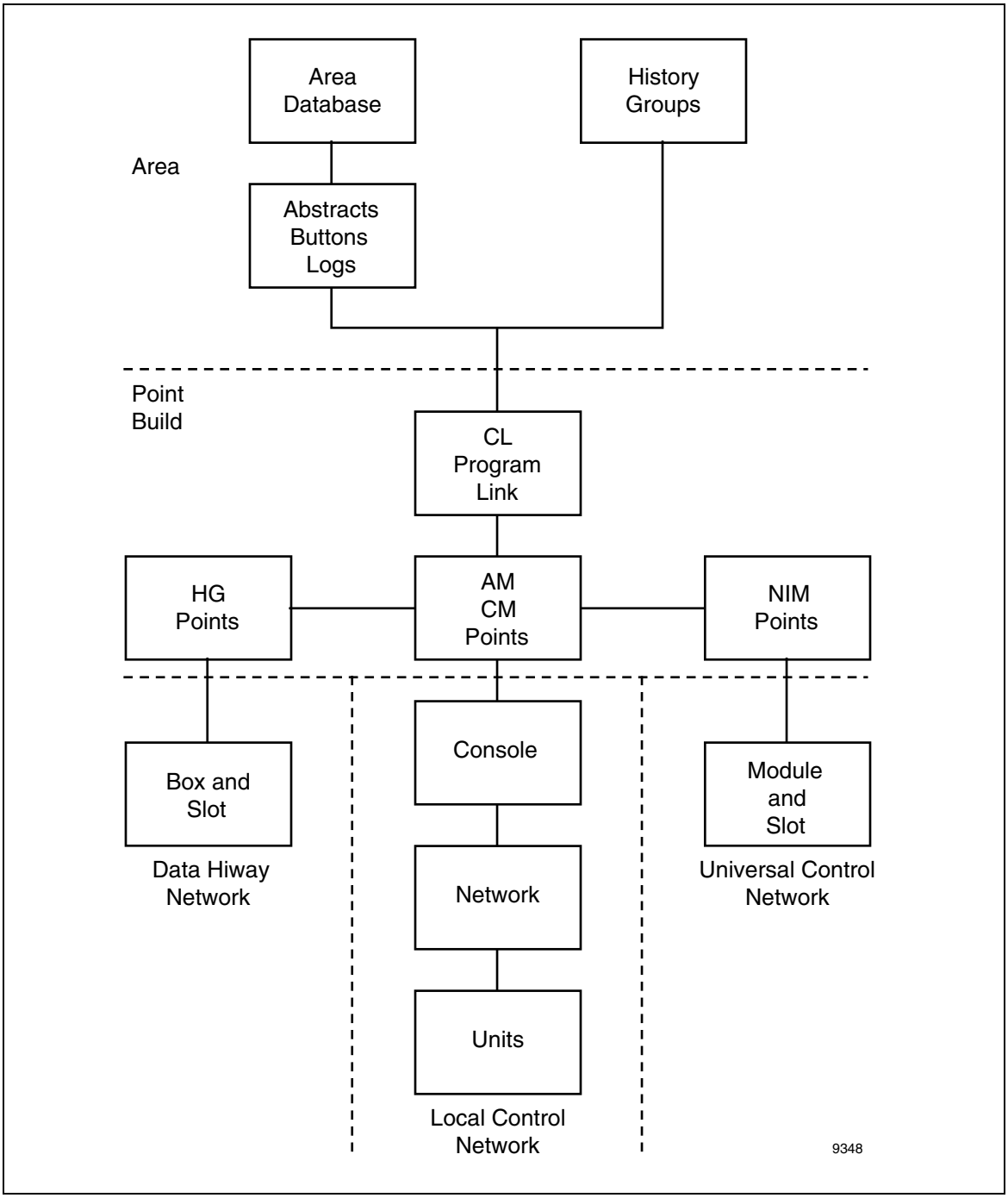

Figure 1 - System Configuration Concepts

DATABASE RESIDENCY

Database Descriptions

This section describes each item in a TPS Network system database:

- How the database item is created.
- Where the database item resides after loading (startup).
- Where the database item is stored.

Table 1 summarizes the following text.

Customer Entity Names and Definitions

- Created in the DEB or using TPS Builder, then loaded to the system.
- Once loaded, they reside in the memory of the data owner.
- Stored in checkpoint files (retain internal identifiers) and also in IDFs or EB files on the HM.

PSDP Reserved Entity Names and Definitions

- The PSDP points are created in each LCN node at that node's startup.
- Once the node is loaded, the PSDPs reside in memory of that node.
- The PSDP information is part of the personality files of the nodes (but the parameters contain real-time data only).

Message Dictionaries

- Message Dictionaries are the NIM and HG libraries.
- They are created upon loading of the system box points at the US/GUS.
- Once loaded, they reside in the memory of the NIM or HG.
- Updated when PM/CL or MC/CL programs are compiled with the -UL option from a US or GUS.
- Stored in the appropriate NIM or HG checkpoint files on the HM.

Standard Parameter Descriptors

- Standard Parameter Descriptors are the parameter definitions (such as type, default values, and key access level) for the different point types that reside in a node.
- These definitions were defined at personality creation time for each data owner and are created in each node at startup.
- They reside in memory of the data owner at node startup.
- These definitions are compiled and stored as part of that node's personality files. The global descriptor files used during point building define groupings (or segments) of parameters.

> **NOTE**
>
> Points are composed of segments. Segments are composed of parameters. The standard Global Descriptor files define the parameters in a segment, and also define what segments go on a particular point.

Custom Parameter Descriptors

- Custom Parameter Descriptors are definitions such as type, default values, and key access levels. They are created when custom data segment CL files are compiled with the -UL option at the US/GUS.
- Custom Global Descriptor files are created that contain the custom parameters and their definitions.
- They reside in memory of the owning node. The custom segment information becomes part of the point structure when a point is loaded to the AM or CG.
- They are stored in the checkpoint files and custom global descriptor files on the HM.

Standard Parameter and Enumeration Names

- These names were compiled as part of each node's personality and are therefore created at personality creation time.
- They are created in node memory at node startup.
- Nodes (AM, CG, US/GUS, NG) that can retrieve point.parameter data by performing external to internal name conversions have this information in memory.
- They are stored in the personality files on the HM.

Custom Parameter and Enumeration Names

- They are created when certain AM/CL files are compiled with the -UL option at the US/GUS.
- They reside in the memory of nodes that need custom parameter and enumeration information.
- They are stored on the HM under directory &ASY as the PARAMETR.SP, ENM_SETS.SE, and PAR_LIST.SE files.

Custom Data Segment Names

- SEGMENTS.SP is updated upon compilation at a US/GUS of certain AM/CL files with the
  -UL option.
- They reside in the memory of nodes (AM, CG, US/GUS, NG) that perform external to internal name conversions.
- Names of custom Global Descriptor files (custom data segments) are stored in the SEGMENTS.SP file on the HM under directory &ASY.

History files

- They are created at the DEB when History Group definition takes place.
- They reside in each HM configured to have history.
- They are stored in the APL*.MM files in each history volume.

Area Database Entities

- They are created at the DEB when Area Database building takes place.
- They reside in US/GUS memory, if loaded with the operator, universal, or workstation personality.
- They are stored in file AREAnn.DA on the HM in directory &Dnn (nn = area number).

AM/CL programs

- They are created upon compilation of certain AM/CL files at the US/GUS.
- Once the object code is linked to an AM point, the program resides in node memory.
- The program information is stored in the AM checkpoint files on the HM.

MC /CL and UCN/CL programs

- They are created upon compilation of MC/CL or UCN/CL files at the US/GUS.
- Once loaded to a Process Module data point, they reside in node memory of the HG or NIM.
- The object files are stored in the &Enn (nn = network number) directory on the HM.

Schematics, Free Format Logs, and Buttons

- They are created at the Picture Editor, the Free Format Log builder, or the Button Configurator from a US/GUS.
- The button file resides in US/GUS memory once an Area load is complete. Classic schematics and Free Format Logs may also reside in US/GUS memory, if specified as memory-resident in the Area Database file; otherwise, they reside in HM user-defined directories.
- Source and object code are stored as user files in a user-defined directory on the HM.

Matrix on Database Residency

Table 1 summarizes where the database functionality resides.

Table 1 - Matrix of Database Residencies

Functionality		AM	HG/NIM	CG	HM	US	NG
Customer Entity Names and Definitions	Created					DEB	
	Resides	Node Memory	Node Memory	Node Memory			
	Stored				IDF,EB, Checkpoint		
PSDP Reserved Entity Names and Definitions	Created	At Startup	At Startup	At Startup	At Startup	At Startup	At Startup
	Resides	Node Memory	Node Memory	Node Memory	Node Memory	Node Memory	Node Memory
	Stored				Personality Files		
Message Dictionaries (NIM and HG libraries)	Created					DEB Compile MC or PM/APM CL w/ -UL	
	Resides		Node Memory				
	Stored				Checkpoint		
Standard Parameter Descriptors (Type, default values, access level)	Created	At Startup	At Startup	At Startup			
	Resides	Node Memory	Node Memory	Node Memory			
	Stored				GDFs and Personality Files		
Custom Parameter Descriptors	Created					Compile AM CL w/ -UL	
	Resides	Node Memory		Node Memory			
	Stored				Custom GDFs, Checkpoint		
Standard Parameter and Enumeration Names	Created	At Startup		At Startup		At Startup	At Startup
	Resides	Node Memory		Node Memory		Node Memory	Node Memory
	Stored				Personality Files		

Table 1 Matrix Of Database Residencies, *continued*

Functionality		AM	HG/NIM	CG	HM	US	NG
Custom Parameter and Enumeration Names	Created					Compile AM CL w/ -UL	
	Resides	Node Memory[1]		Node Memory		Node Memory	Node Memory
	Stored				&ASY> parameter.SP enm_sets.SE par_list.SE		
Custom Data Segment Names	Created					Compile AM CL w/ -UL	
	Resides	Node Memory[1]		Node Memory		Node Memory	Node Memory
	Stored				&ASY> segments.SP		
History Group Definition	Created					DEB	
	Resides				Node Memory		
	Stored				APL*.MM		
Area Database	Created					DEB	
	Resides					Node Memory[2]	
	Stored				&Dnn		
CL/AM Programs	Created					CL Files	
	Resides	Node Memory					
	Stored				Checkpoint		
CL/MC & CL/PM/APM/ HPM programs	Created					CL Files	
	Resides		Node Memory				
	Stored				&Enn		
Picture Editor schematics, FFLs, & Button Files	Created					PE, FFLB, Button Config	
	Resides				User Directories	Node Memory	
	Stored				User Files		

NOTES:

1. Custom names reside in AM memory if that AM is configured to have the CONV external load module.
2. The Area Database is in US/GUS node memory if the US/GUS is loaded with the operator, universal, or workstation personality.

MATRIX ON DATABASE CHANGES

Table 2 summarizes the effects of database changes.

Table 2 - Matrix for Database Changes

Effect of Changes: X = Up to severe impact if certain nodes not reloaded. I = HM must be initialized. U = US(s) must be reloaded/area change. O = Affects only owning node, or file change.	Type of Data Change							
	NCF	NVCF	Area Data	Node Data	Schematics	Groups, Trends	Logs, Reports	CL Programs
Unit Names	X	I	U					
Area Names	X							
Console Names	X							
Systemwide Values	X							
System or User Volumes	X	I						
Entity Data			U	O	O	O	O	O
Delete Entity References			U	O	O	O	O	O
Groups, Trends, Logs, Reports			U			O	O	
Area Pathname Catalog			U					
Picture Editor Schematics			U		O			
Button Configuration			U					
CL Programs								O

Node data is the owning node's database.

9349-A

WRITTEN EXERCISE

Complete the following exercise to reinforce your understanding and progress
This section is intended to be completed with your colleagues.

Using Figure 1, Table 1, and Table 2, summarize the effects of a change and what corrective action (if any) is required:

1. Unit names:
   Change affects the following:_____

   _____

   Type of corrective action:_____

   _____

2. Area names:
   Change affects the following:_____

   _____

   Type of corrective action:_____

   _____

3. System or user volumes:
   Change affects the following:_____

   _____

   Type of corrective action:_____

   _____

4. Entity data:
   Change affects the following:_____

   _____

   Type of corrective action:_____

   _____

5. Delete entity references:
Change affects the following:_____

_____

Type of corrective action:_____

_____

6. Groups, trends, logs, reports:
Change affects the following:_____

_____

Type of corrective action:_____

_____

7. Area pathname catalog:
Change affects the following:_____

_____

Type of corrective action:_____

_____

8. Picture Editor schematics:
Change affects the following:_____

_____

Type of corrective action:_____

_____

9. Button configuration:
Change affects the following:_____

_____

Type of corrective action:_____

_____

10. CL programs:
Change affects the following:_____

_____

Type of corrective action:_____

_____

SOLUTIONS

The following solutions are possible answers to the preceding exercise. Note that other correct answers may exist.

Using Figure 1, Table 1, and Table 2, summarize the effects of a change and what corrective action (if any) is required:

1. Unit names:

   Change affects the following: *NCF database. May also require HM initialization, if a new unit was added.*

   Type of corrective action: *Install new NCF. Initialize HM.*

2. Area names:

   Change affects the following: *NCF database*

   Type of corrective action: *Install new NCF.*

3. System or user volumes:

   Change affects the following: *NCF database, HM volume configuration*

   Type of corrective action: *Install new NCF, Initialize HM.*

4. Entity data:

   Change affects the following: *Area data, node data, schematics, groups, trends, logs, reports, CL programs*

   Type of corrective action: *Perform area change at US/GUS. May require recompiling or reinstalling schematics, groups, trends, logs, reports, CL programs.*

5. Delete entity references:

   Change affects the following: *Area data, node data, schematics, groups, trends, logs, reports, CL programs*

   Type of corrective action: *Perform area change at US/GUS. May require recompiling or reinstalling schematics, groups, trends, logs, reports, CL programs.*

6. Groups, trends, logs, reports:

   Change affects the following: *Area database*

   Type of corrective action: *Perform area change at US/GUS.*

7. Area pathname catalog:

   Change affects the following: *Area database*

   Type of corrective action: *Perform area change at US/GUS.*

8. Schematics:

   Change affects the following: *Area database*

   Type of corrective action: *Area change at US/GUS if schematic is memory resident.*

9. Button configuration:

   Change affects the following: *Area database*

   Type of corrective action: *Perform area change at US/GUS.*

10. CL programs:

    Change affects the following: *AM, MC, or PM/APM/HPM object code on volume &Enn must be updated for MC or PM/APM/HPM.*

    Type of corrective action: *AM - Recompile, relink. HPM/APM/PM/MC - Recompile, reload program to device.*

DISCUSSION EXERCISE

This section is intended to be completed with your colleagues.

If the following controller points are in your database, can you determine what parameters reside in the gateway versus what parameters reside in the controller? Refer to Figure 2 through Figure 8 for this discussion.

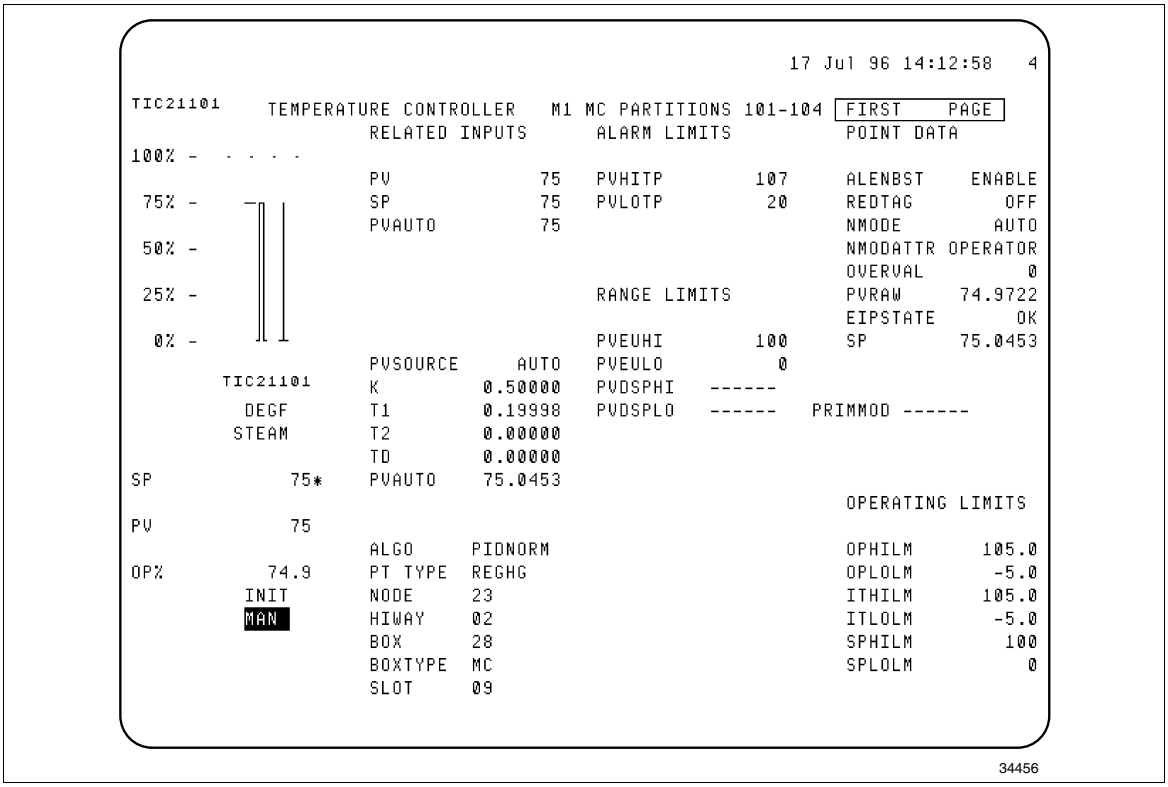

Figure 2 - Data Hiway Detail Display Example, REGHG Point (First Page)

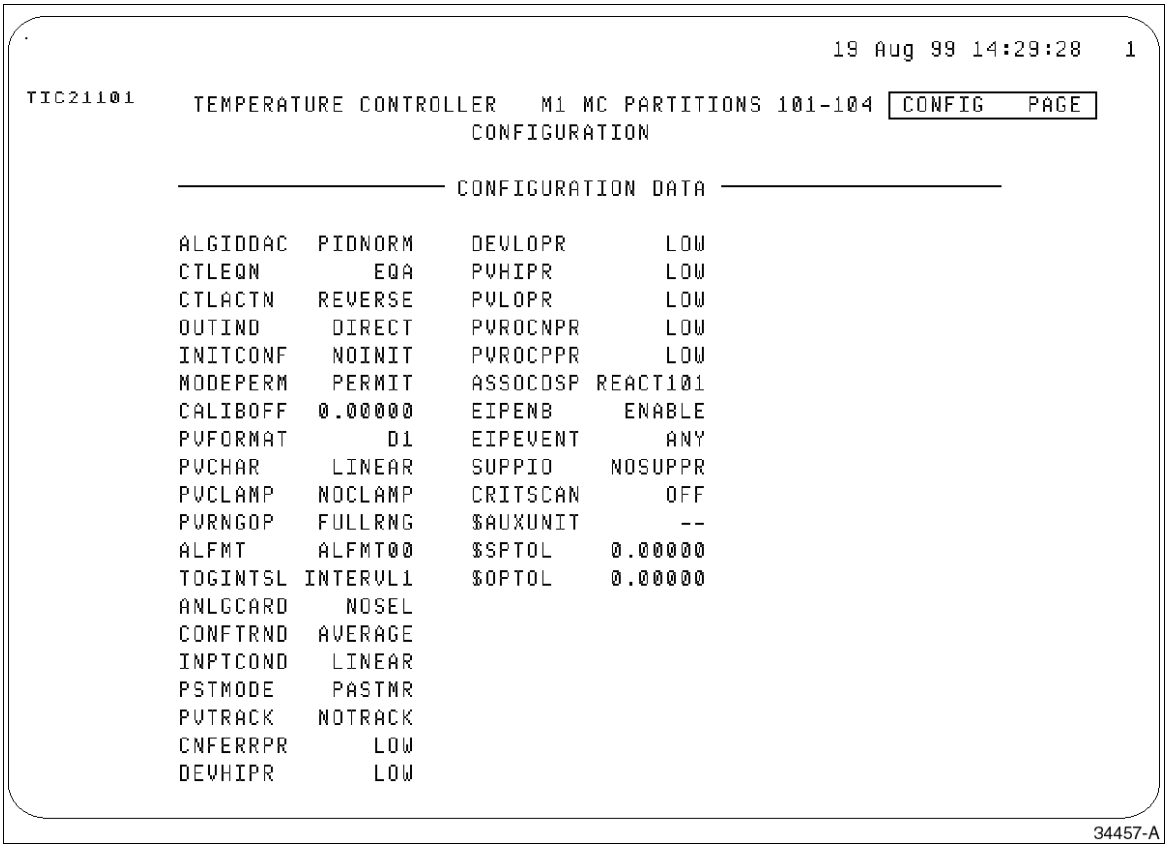

Figure 3 - Data Hiway Detail Display Example, REGHG Point (Config Page)

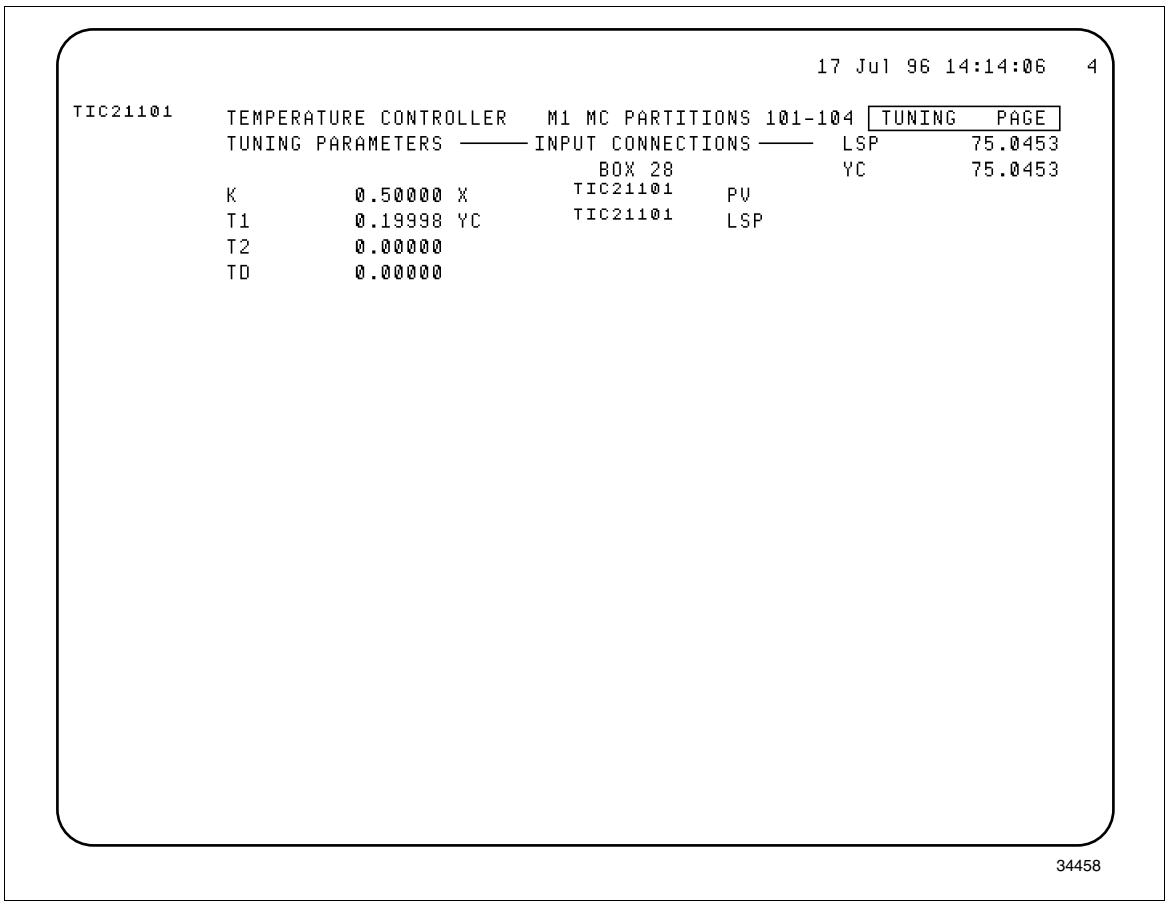

Figure 4 - Data Hiway Detail Display Example, REGHG Point (Tuning Page

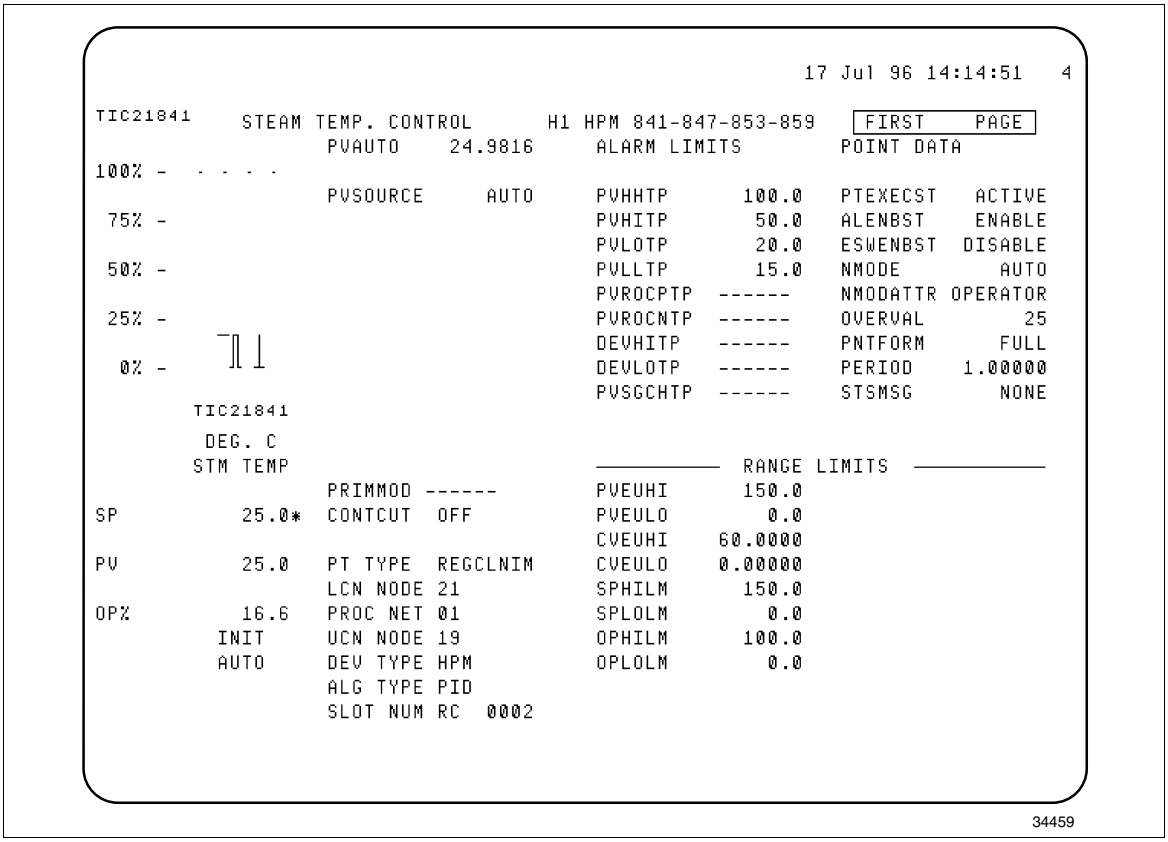

Figure 5 - UCN Detail Display Example, RECCLNIM Point (First Page)

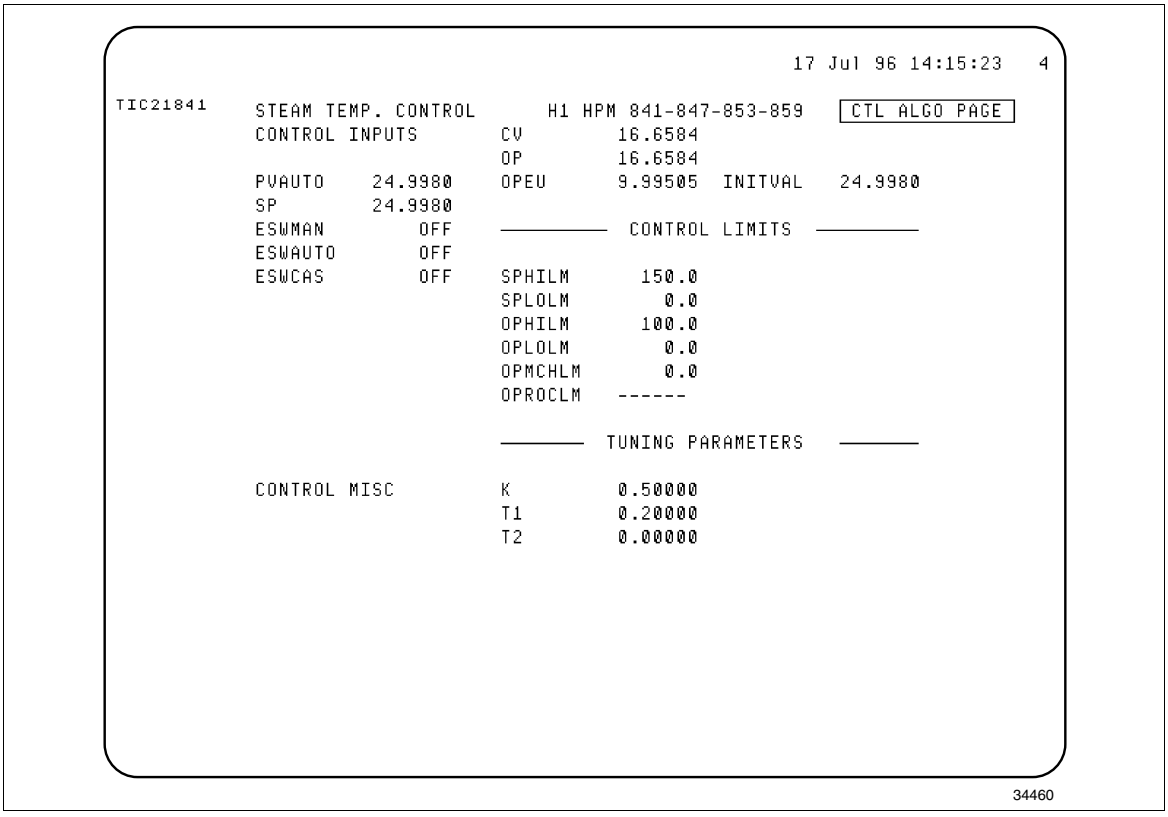

Figure 6 - UCN Detail Display Example, RECCLNIM Point (Ctl Algo Page)

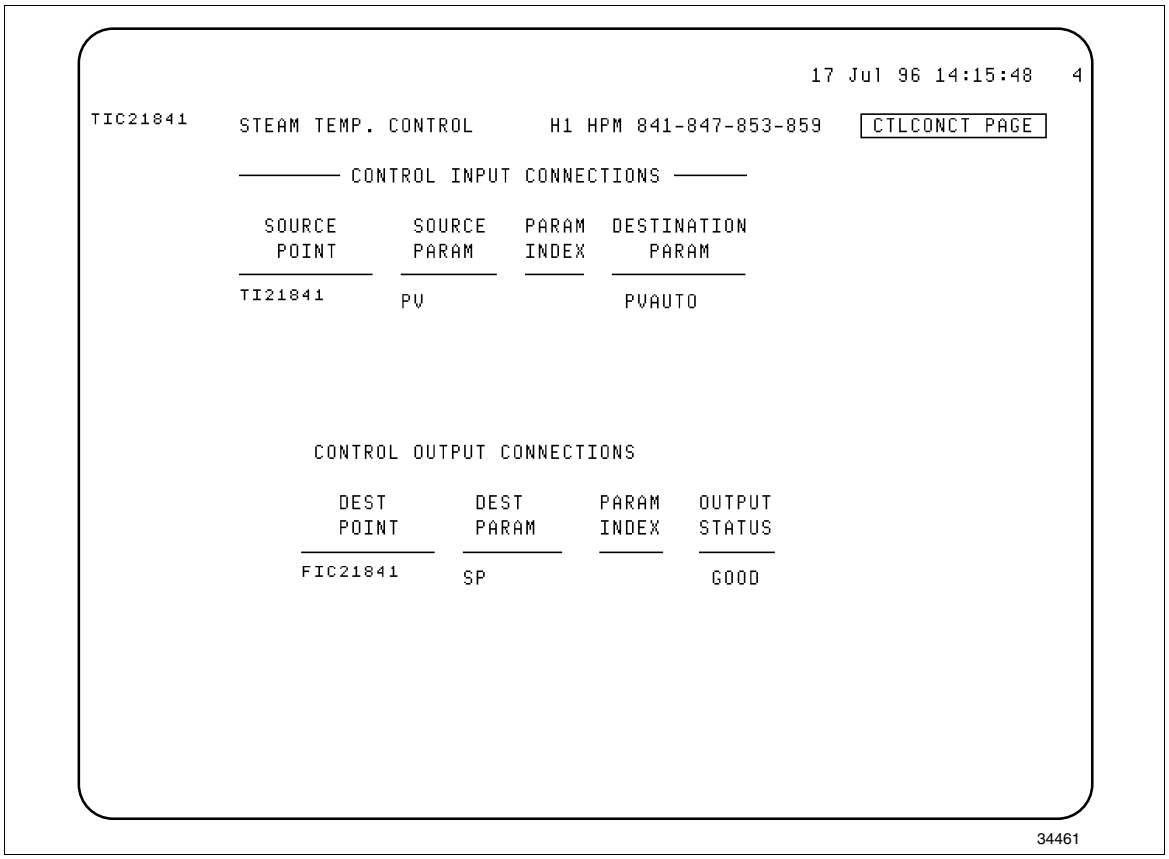

Figure 7 - UCN Detail Display Example, RECCLNIM Point (CtlConct Page)

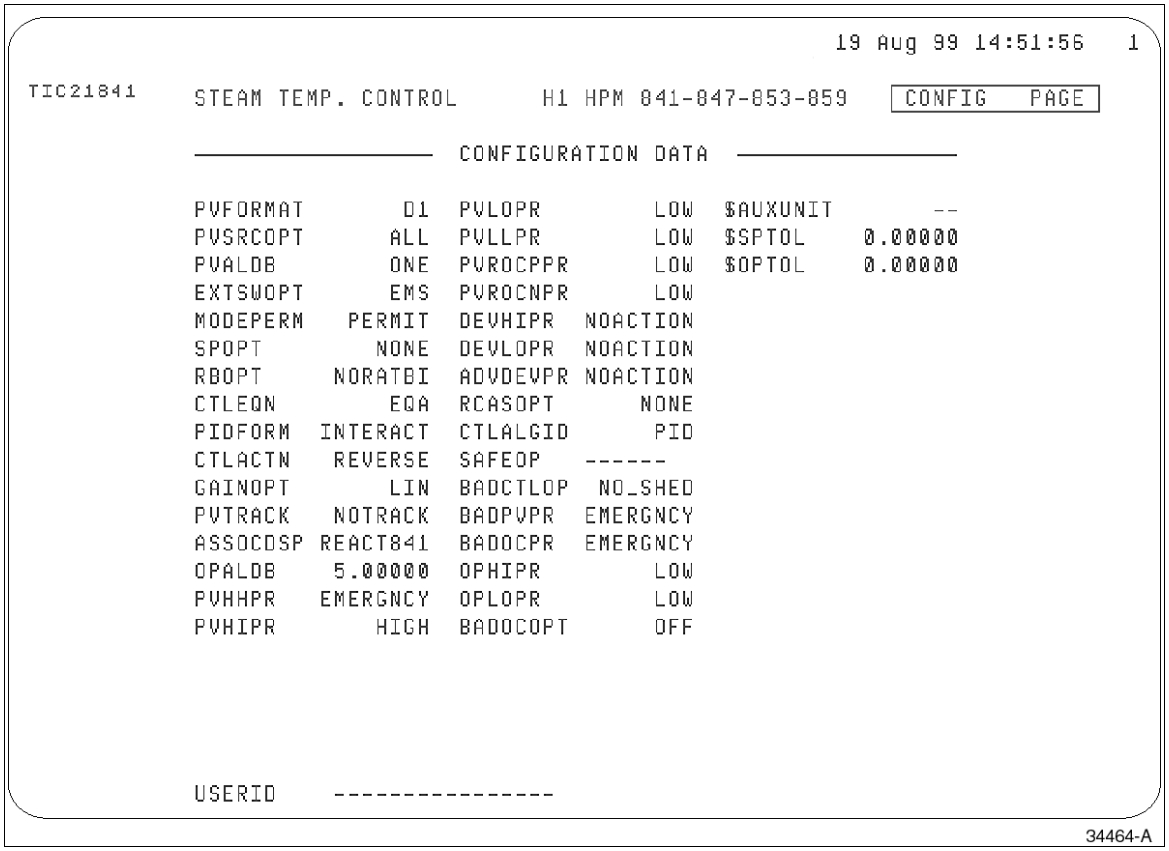

Figure 8 - UCN Detail Display Example, RECCLNIM Point (Config Page)

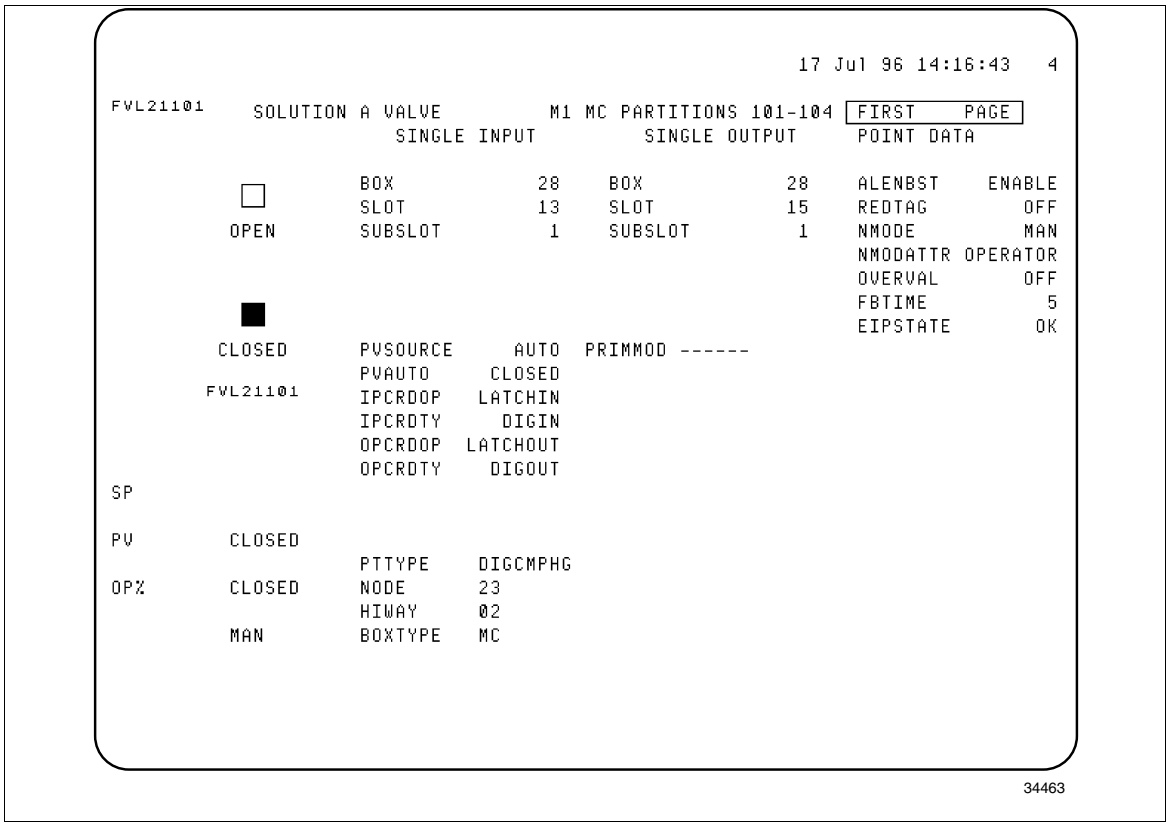

Figure 9 - Hiway Detail Display Example, DIGCMPHG Point (First Page)

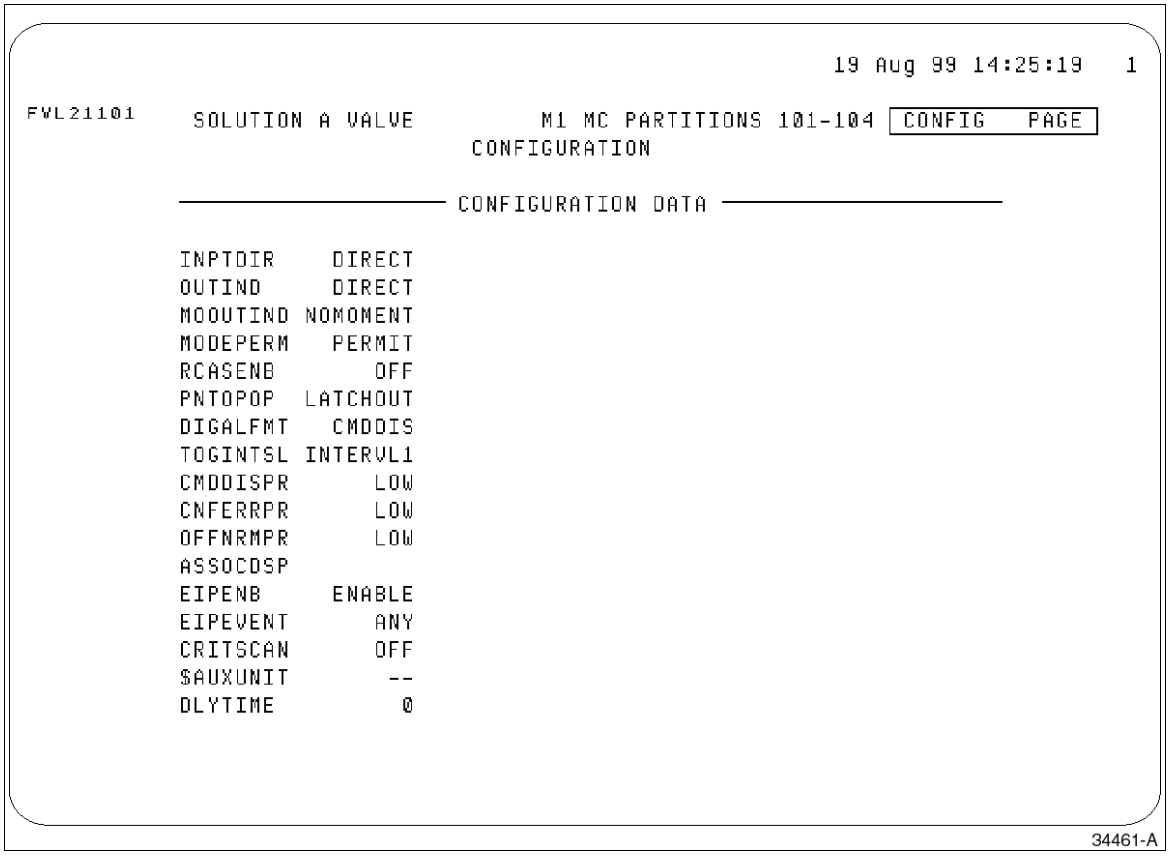

Figure 10 - Hiway Detail Display Example, DIGCMPHG Point (Config Page)

SOLUTIONS

The following solutions are possible answers to the preceding discussion questions. Note that other correct answers may exist.

If the following controller points are in your database, can you determine what parameters reside in the gateway versus what parameters reside in the controller? Refer to Figures 2 through 8 for this discussion.

Refer to the appropriate Parameter Reference Dictionary to determine parameter residency.

In a training setting or an off-process situation, you can easily determine parameter residencies of Data Hiway points by removing power from the Data Hiway box, then calling up a Detail display. Parameters that are resident in the Data Hiway box will appear with question marks. Figures 9 and 10 shown the Detail display for a Multifunction Controller (MC) Regulatory Control point on the Data Hiway. Table 3 lists the point's parameters and their residency.

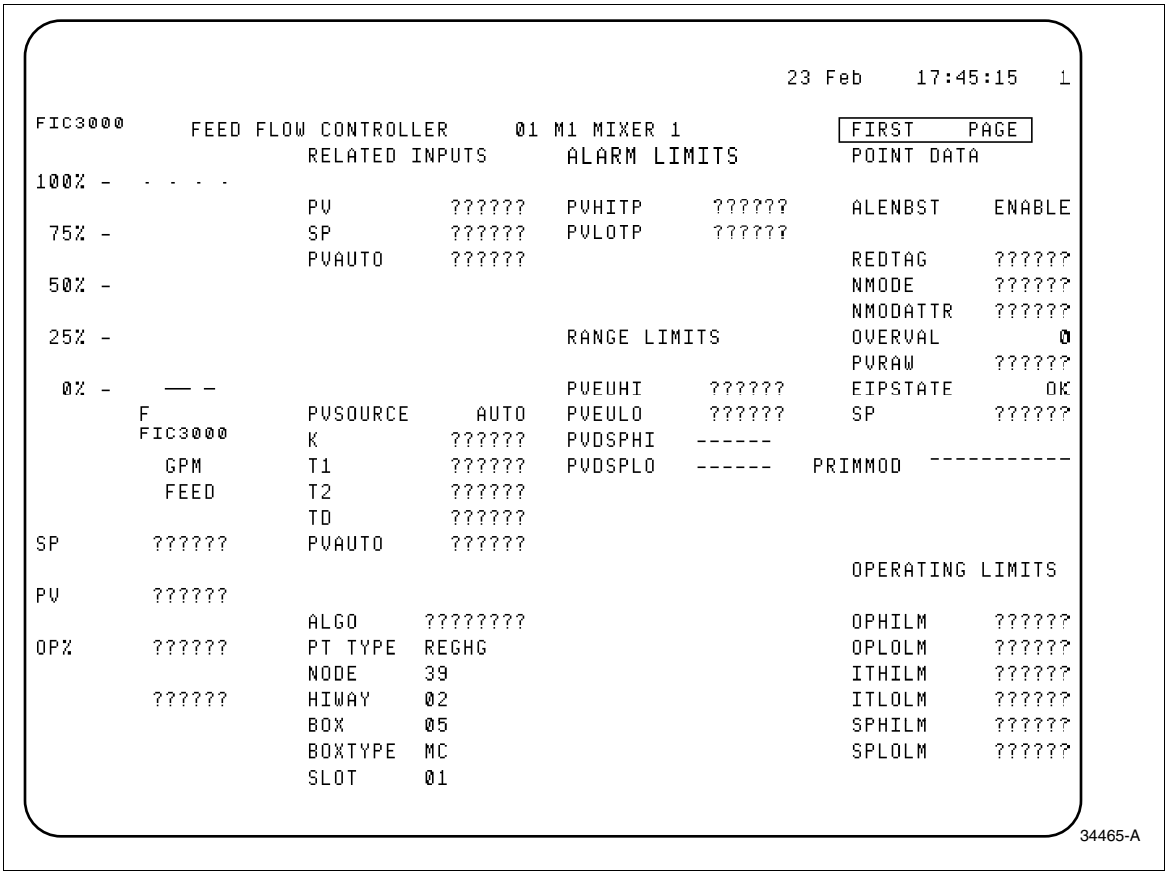

Figure 11 - Data Hiway Point Detail Display (First Page)

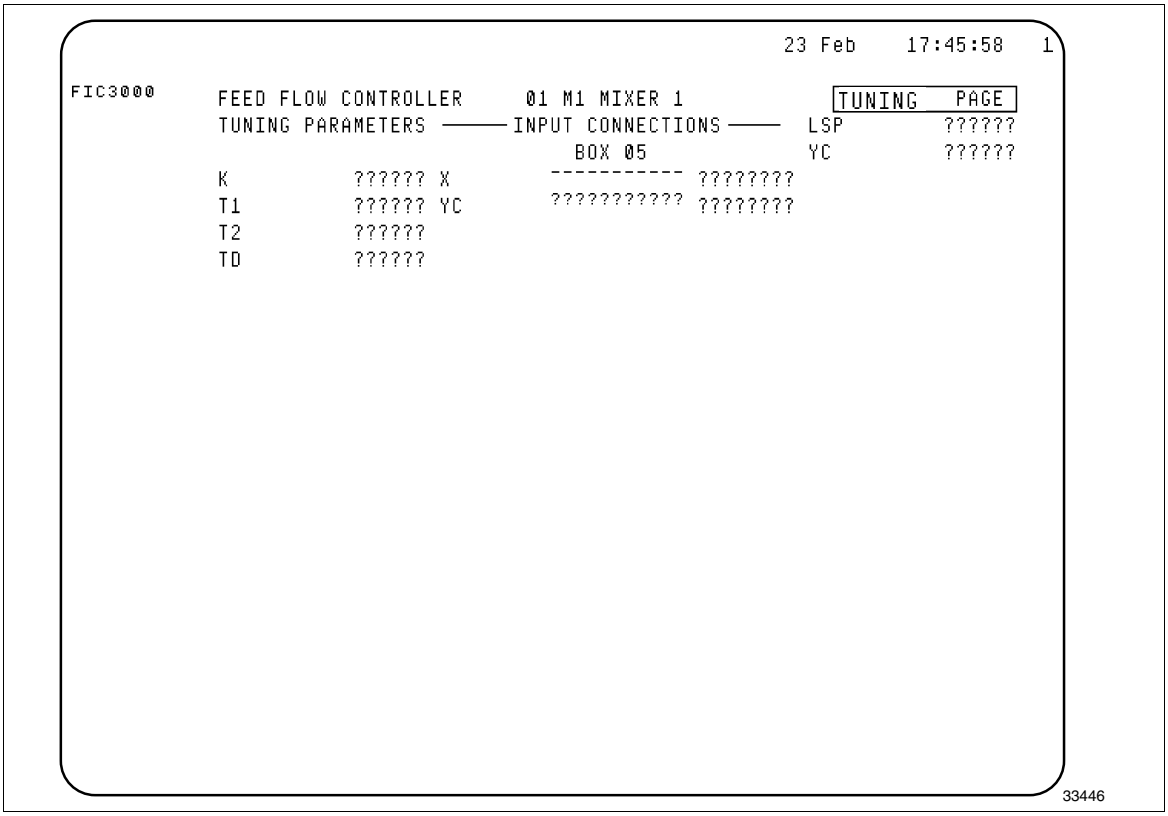

Figure 12 - Data Hiway Point Detail Display (Tuning Page)

Table 3 - MC Data Hiway REGCTL Point Database Residency

HG	BOX	HG and BOX
ALPRIOR (before R500)	ALMDIS	ALFMT
BADPVPR (R500)	CONFTRND	ALGIDDAC
BOXNUM	CTLACTN	
CALIBOFF	CTLEQN	
CCRANK	INITCONF	
CRITSCAN	INPTCOND	
EIPENB	ITHILM	
EIPEVENT	ITLOLM	
EIPPCODE	K	
EUDESC	MODE	
HWYNUM	NMODATTR	
KEYWORD	NMODE	
LOADDEST	OPLHILM	
MODEPERM	OPLOLM	
NAME	OUTIND	
OVERVAL	PSTMODE	
PNTBOXTY	PVCHAR	
PRIMMOD	PVEUHI	
PTDESC	PVEULO	
PTDISCL	PVFORMAT	
PVCLAMP	PVHITP	
PVDSPHI	PVLOTP	
PVDSPLO	PVSIGNAL	
PVLOPR (R500)	PVSLTSRC	
PVHIPR (R500)	PVTRACK	
PVRNGOP	SPHILM	
SLOTNUM	SPLOLM	
SUPPIO	SPSIGNAL	
UNIT	SPSLTSRC	
	T1	
	T2	
	TD	

FIND NAMES

Description

Find Names is a Command Processor utility that helps you find interconnections between database entities in order to determine database dependencies.

To access the utility from the Command Processor, enter `FN`.

Good Practices

Find Names is a file search (as opposed to a node query); as a result, the following are good practices:

1. Copy the file to be searched to removable media and direct your search there. This reduces the HM load and prevents the possibility of corrupting the master file.

2. If removable media is not available, copy the file to be searched to a user directory on the HM and perform the search there.

3. If a checkpoint is searched, you may want to demand a checkpoint before the search to ensure that the file has the most current data.

4. You can frequently reduce the duration and scope of your search if you are able to specify paths, units, and network numbers.

When to use Find Names

Suppose you are planning to modify a point and you need to know all the places that the point is being used.

Use Find Names to determine if the point is being used in schematics, programs, history groups, and area database entities.

Example:

After using Find Names to search Schematics, assume that you discover that the point is used in a schematic. You now know that after modifying the point, you may need to modify and recompile the schematic.

Use Find Names again to determine if the schematic is specified in any Area database entities (such as the pathname catalog).

If you discover that the schematic is specified in an Area database's pathname catalog, after the schematic is modified you will then have to perform an Area Change at any US/GUS loaded with that Area database.

Navigation Overview

Default pathnames

Figure 13 shows the Find Names display; on the display we have indicated the default pathnames that apply to each database selection.

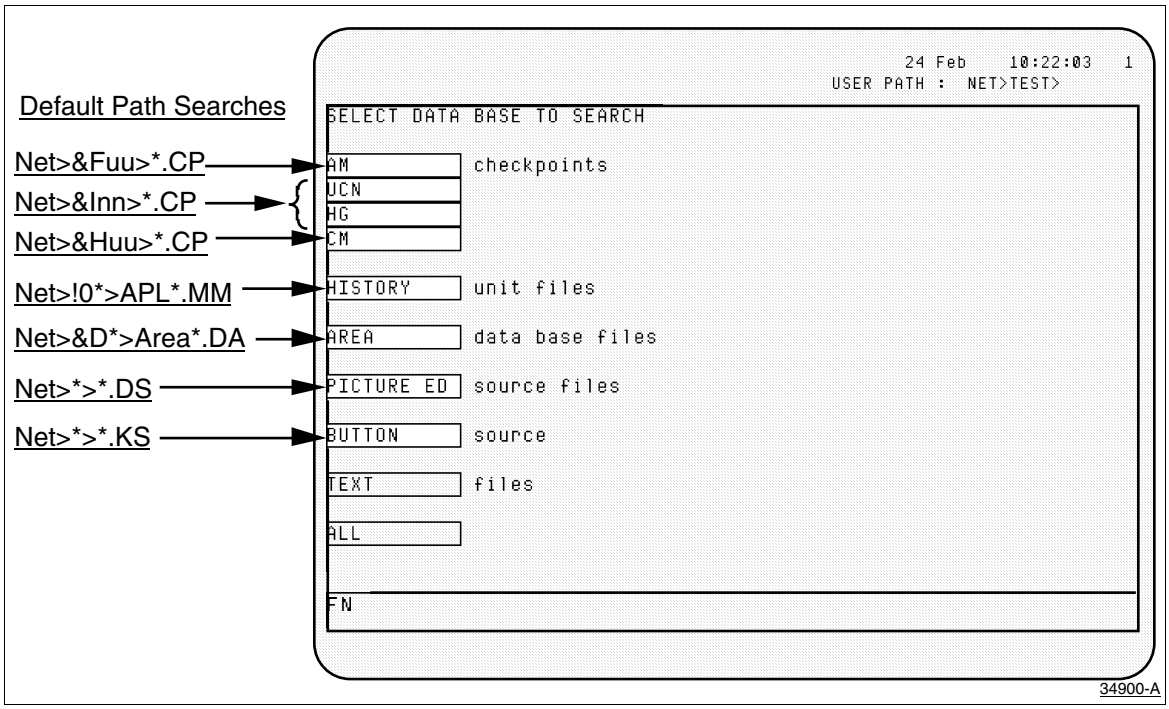

Figure 13 - Navigation to Find Names Menus

Find Names Display

Figures 14 and 15 show the navigation paths from the Find Names display.

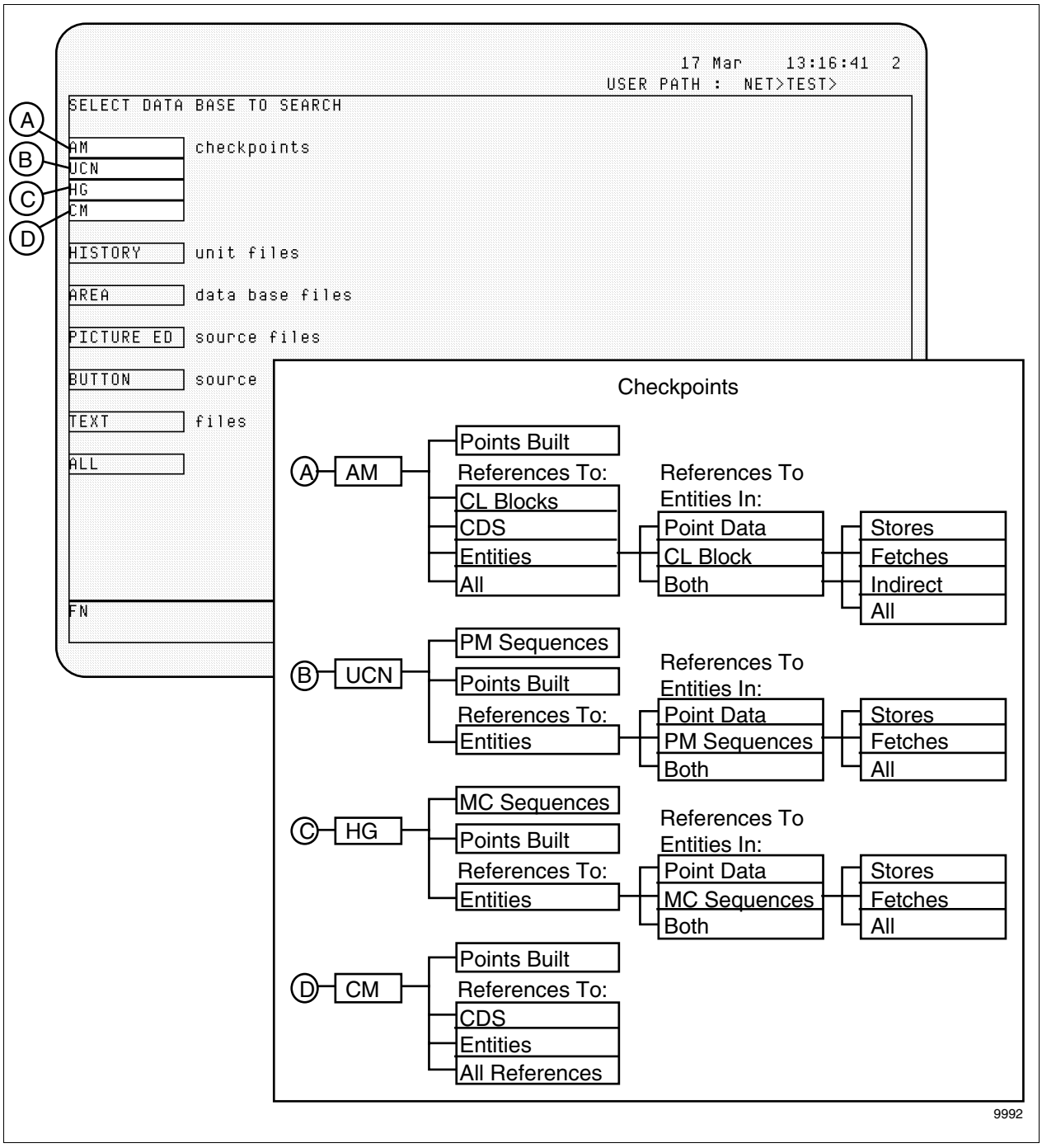

Figure 14 - Navigation to Find Names Menus

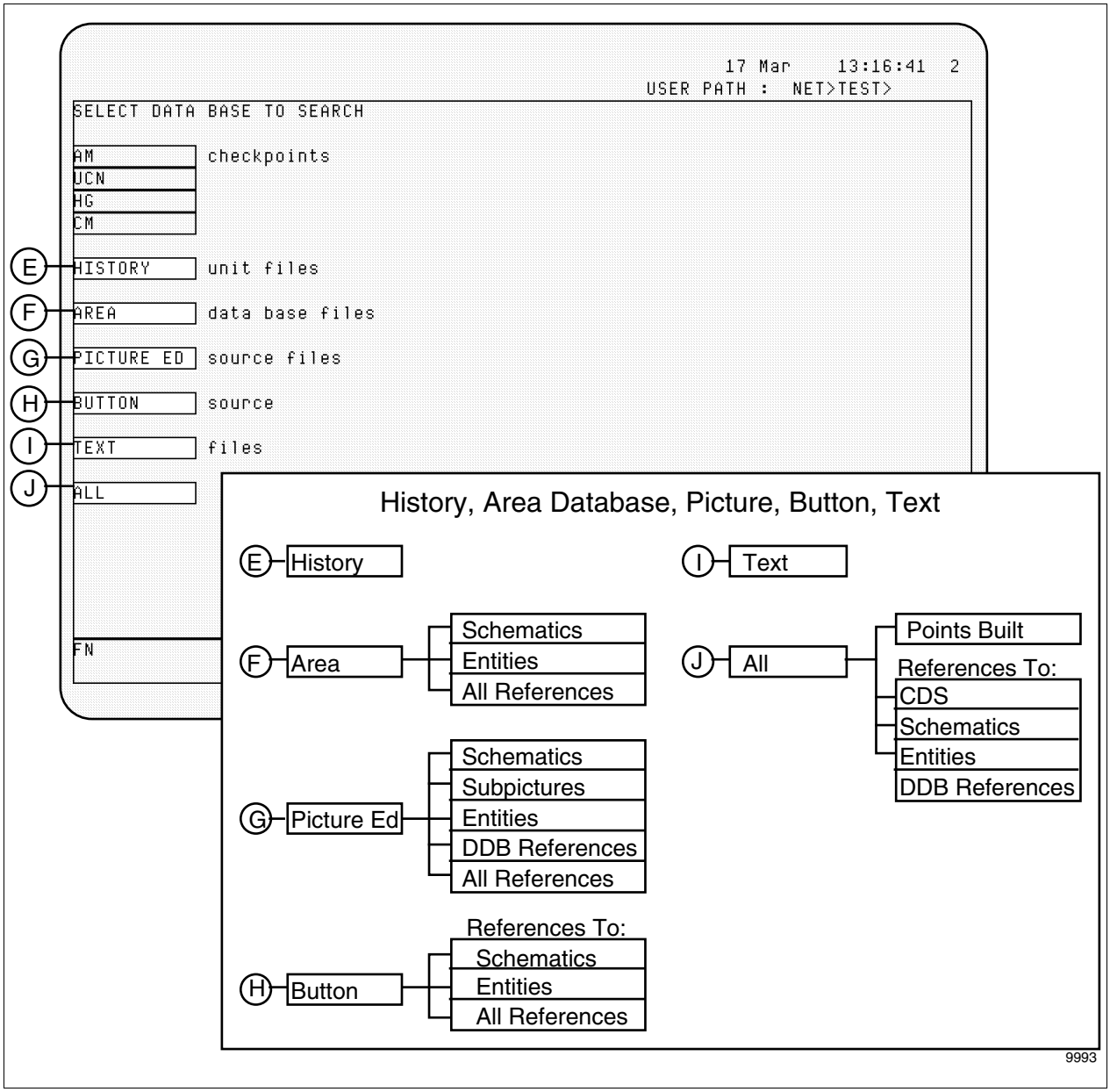

Figure 15 - Navigation to Find Names Menus

Doc Tool vs.Find Names

The Doc Tool and the Find Names Utility can both be used to document the system database. There are differences between the two in the data sources used as well as the format of the data displayed. Table 4 outlines some of these differences.

Table 4 - Doc Tool vs. Find Names

Function	Doc Tool	Find Names
Searches the following files for cross references to entities or patterns: • Checkpoint • History unit • Area Database • Picture Editor and Free Format Log source • Button source • Text files		X
Searches display databases in .DS files.		X
Lists the file name that contains the character string you are searching for and prints out the string.		X
Searches the online databases of LCN nodes and process-connected devices.	X	
Displays the entire contents of a file once the specified character string is found.	X	
Saves engineering effort and time by arranging information in user-defined manner.	X	

Area Database Files (Entities)

Find Names Selections for Area

Table 5 describes the Area Database selections on the Find Names menu. Table 6 lists the Area database entity names and their description.

Table 5 - Description of Find Names Selections for Area Database

Selection	Description
Entity References	Refers to the tagname.parameter to search for.
Area Database to search?	Refers to the &Dnn (where nn is a number from 1 to 10) Area database files. A “wild card” default value of “*” searches all areas.
Entities to search?	Refers to the Area database entities listed in Table 6. In other words, what do I want to search in the Area database? For example, “$OGROUP(nnn)” is the Area database operating group *entity.*

Table 6 - Area Entities to Search

Entity Name	Description
$ANNDATA	Annunciator policy
$ANPOLC	LCN node annunciation policy
$DMONBLK(n)	Hiway box annunciation policy
$OABSTRT	Pathname catalog
$OBTCHSM	batch Summary display
$OFFLOG(n)	Free format log
$OGROUP(n)	Group display
$OLOG(n)	Standard log
$OMODGRP(n)	Process module group
$OPRCJOR(n)	Process journal
$OPRTRND(n)	Printed trend
$OREPORT(n)	Report
$SOEJOR(n)	SOE Journal
$OSYSJOR(n)	System Journal
$OSYSTAT	System Status
$OTREND	Area Trend display
$OUNITSM(N)	Unit Summary display
$OUNTRND(n)	Unit Trend display
$RTJNL	Real Time journal assignment
$RUNTSTS	Unit Assignment display

LAB TIME

•30 Minutes

Use your US.

Take with you:

- This module

LAB EXERCISES

Lab Exercise 1—Find Names

Lab Introduction

This lab exercise is intended to reinforce your understanding of database concepts. In this lab, you will use the "Find Names" utility to search the database for the occurrences of an entity.

Lab Instructions

1. Ask your course manager to assign the entity to be searched for.

2. Use the Find Names utility to search the database for occurrences of the assigned entity in the following:

   a. Checkpoint file

   b. Area database

   c. Schematic

3. Search the Area Database for references to the schematic found in step 2c.

4. Try any other Find Names search that you are interested in.

End of Lab Exercise 1

Lab Exercise 2—More Find Names Practice

Lab Instructions

1. Assume that you will move an assigned UCN or Hiway entity to another unit. This move must be transparent for the operator. Determine the impact of this move to the database.

   The solution follows the lab exercises.

2. Assume that you will move the secondary of a cascade loop in a UCN device to another slot in the UCN. What will be the impact of this move to the database?

   The solution follows the lab exercises.

3. Assume a flow transmitter used in a control loop failed and was replaced with a new transmitter from a different vendor.

   process range: 0-50 GPM
   old transmitter range: 0-60 GPM
   new transmitter range: 0-120 GPM

   What database changes might need to be made as a result?

   The solution follows the lab exercises.

End of Lab Exercise 2

Optional Lab Exercise 3—Find Untagged

Lab Introduction

In this lab exercise, you will locate untagged references. Suppose you have been assigned to a task that includes finding out and possibly changing what an engineer has configured; or you need to replace an HPM, APM, or PM Input/Output Processor module. You have heard about "untagged references" and want to make sure you account for them before you make any changes.

Untagged hardware references can be used to form connections between regulatory PV and control points, and I/O points within a PM/APM/HPM. They are referenced by module type, module number, and slot number as shown in the following examples:

- !DI06S01
- !DO05S04
- !AO12S05

The exclamation point (!) indicates an untagged hardware reference. The next two characters identify the type of point, the next two identify the module number, and the last three identify the slot number within the module.

Before continuing with the lab exercise, read the following Honeywell recommendation about the use of untagged references.

HONEYWELL RECOMMENDATION FOR UNTAGGED REFERENCES

Honeywell recommends not using untagged references, but also recognizes that there may be exceptions. Before deciding to use untagged references, the user should carefully weigh the following disadvantages and advantages. If the user decides to use an untagged reference for a particular application, Honeywell strongly recommends that the user cautiously plan for the long term management of such a configuration.

Disadvantages

The following list describes the disadvantages to using untagged references:

- The user must use the Command Processor Find Names function to see what connects to the hardware reference address.
- A node's Slot Summary display does not indicate untagged references; consequently, personnel may misinterpret the display. For example, a maintenance technician may infer that no tags would be affected by an IOP board removal, or a project engineer may incorrectly infer that any untagged slots are available for use.
- The user must accept the default database provided by an untagged reference (for example, untagged analog outputs cannot have an output characterizer).
- Operating displays cannot include untagged points.
- The APM and HPM I/O simulator program does not support untagged references.

Advantages

There are limited advantages to using untagged references; they are:

- Point building time is slightly reduced due to the reduced number of points.
- Provides a means for "invisible" references when it is desirable for items to be fairly inaccessible to the operator; however, untagged references can still be detailed from the Slot Summary display.
- Untagged references do not deduct from the NIM point count, which is 8000 points per NIM.

Lab Instructions

One way to check for untagged references is to use Find Names. You have a number of options in using Find Names, and effectively using wild card characters (such as ? or *) broadens your search for untagged references.

The example in this exercise will return a HPM, APM, or PM Input/Output Processor module's untagged references.

Figure 17 shows the display results you should expect to see.

Step	Action
1	From the Command Processor, type in FN, press [ENTER]. This will invoke the Find Names function.
2	Select the UCN target from the checkpoint field.
3	Select the ENTITIES target (do not select the POINTS BUILT target.)
4	After the screen updates, select the POINT DATA target.
5	After the screen updates, enter the following: a. In the ENTITY REFERENCES port, enter an untagged identifier with wild card characters. Example: If you are searching digital output module number 4, enter !DO04S??.* in the port. This will capture all untagged references for that module. b. Enter the UCN checkpoint number. Example: Enter 03 for UCN network number 3. c. Enter the UCN node number to search. Example: Enter 05 if the PM/APM/HPM address is number 5. d. The remaining entries can be default entries. Press [ENTER]. RESULT: Any tagged and untagged reference defined in that node's checkpoint is returned in hardware format; then all other references appear, as shown at the bottom of Figure 16.
6	If using a system running R400 or later software, press the [CTL] [R] keys to shift the display to the right to see any additional information.
7	You will need to look at the slot summary of the IOP from the UCN status display to determine which references are tagged and which are untagged.

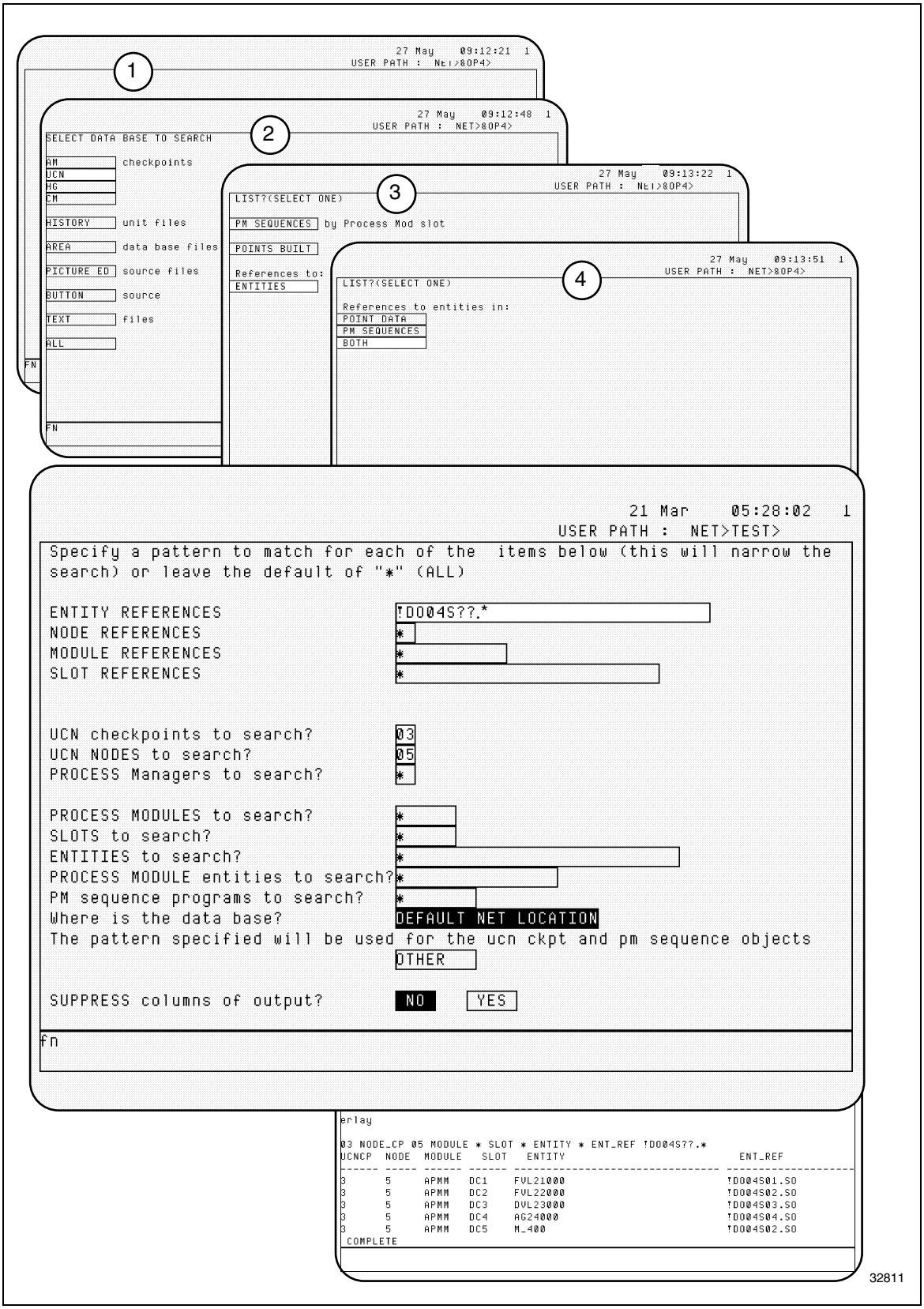

Figure 16 - Lab Exercise—How to Find Untagged Connections

End of Optional Lab Exercise 3

SOLUTIONS—LAB EXERCISE 2

1. *Solution:*

   Since the LCN internal ID of the entity will be changed when the entity is moved to another unit, run a Find Names search to determine what database entities/elements would need to be reloaded or recompiled, such as:

   - *other UCN or Hiway, AM, or CG entities referencing this point,*
   - *History Group entities*
   - *Area Database entities*
   - *schematic references*
   - *button file references*
   - *CL program references.*

2. *Solution:*

   A move on the UCN will not cause a change in the LCN internal ID. As a result, all LCN database items (such as history, area, schematics, buttons, AM/CL) will not be affected by the move; however, all UCN references to this entity will be affected.

   Other UCN points referencing it will lose their connections and will need to be reloaded. All UCN CL programs referencing this point will need to be recompiled. Find Names can be used to determine the UCN database items that will be affected.

3. *Solution:*

   a. *Anywhere PVP is used (such as in CL programs) database changes may need to be made. Find Names can be used to find these connections.*

   b. *The Gain (K) tuning constant of the controller should be doubled in order to have the same process response. The PID algorithm Gain is calculated using the difference between the process variable in percent (PVP) and the setpoint in percent (SPP).*

LAST PAGE