Honeywell

Manage UCN Database

L5691
UCN

TotalPlant®

Copyright, Notices, and Trademarks

Revision 03 – February 27, 1997

Honeywell IAC courseware is subject to change without notice.

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

TPS is the evolution of TDC 3000^{X}.

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

1-800-852-3211

Table of Contents

Figures and Tables

Acronyms

AM ..... Application Module
AO ..... Analog Output
APM ..... Advanced Process Manager
APMM ..... Advanced Process Manager Module
CG ..... Computer Gateway
DEB ..... Data Entity Builder
DI ..... Digital Input
DO ..... Digital Output
EB ..... Exception Build
EIP ..... Event Initiated Processing
HG ..... Hiway Gateway
HM ..... History Module
I/O ..... Input/Output
IDF ..... Intermediate Data File
IOM ..... Input/Output Module
IOP ..... Input/Output Processor
LCN ..... Local Control Network
LM ..... Logic Manager
NCF ..... Network Configuration File
NIM ..... Network Interface Module
PLCG ..... Programmable Logic Controller Gateway
PM ..... Process Manager
PMM ..... Process Manager Module
PSDP ..... Processor Status Data Point
TAC ..... Technical Assistance Center
UCN ..... Universal Control Network

Parameters

PV ..... Process Variable
PVFL ..... PV flag

References

Publication Title	Publication Number	Binder Title	Binder Number
For R500 and later:			
Engineer's Reference Manual	SW09-505	Implementation/Startup & Reconfiguration	TPS 3030-2
For R400 and later:			
Engineer's Reference Manual	SW09-405	Implementation/Startup & Reconfiguration	TPS 2030-2

Introduction

Module Overview

About this module

This course module discusses considerations users encounter when saving or restoring database, or when changing existing configurations. As a result of completing this course module, you gain a better understanding of database concepts and procedures to follow when managing your UCN database.

Objectives

The objectives of this course module are to

- Change a node specific configuration.
- Describe how to clear IOP database.
- Use an Exception Build procedure for heavily loaded systems.
- Interpret what point inactivation and activation means.
- List checkpointing considerations, in particular
  - identify effects of disabling and inhibiting checkpoints
  - Automatic and demand checkpointing considerations
  - Discuss checkpoint file structures
- Define what is meant by valid/invalid IOP data
- List causes of point fragmentation (ghost points).

Sample test items

This course module's Criterion Test includes the following items:

- Suppose that you are changing the control partition in a UCN node. So far, you have
  - Backed up all control partition points into IDFs,
  - Reconstituted the network box configuration point, $NMuuBbb,
  - Made desired changes to the point mix, and
  - Loaded the new point/slot configuration.

  At this point, you pull up a slot summary display for the node. Will you see any tagnames displayed? Why or Why not?
- Obtain a general description of the files created when you demand a checkpoint SAVE for a specific UCN node.
- Describe the meaning of the possible values for the NIM LOADSCOP parameter, and give examples of when to use each.
- Give a definition of the two types of point fragments that can exist.

Configuration Changes

Changing Node Specific Configurations

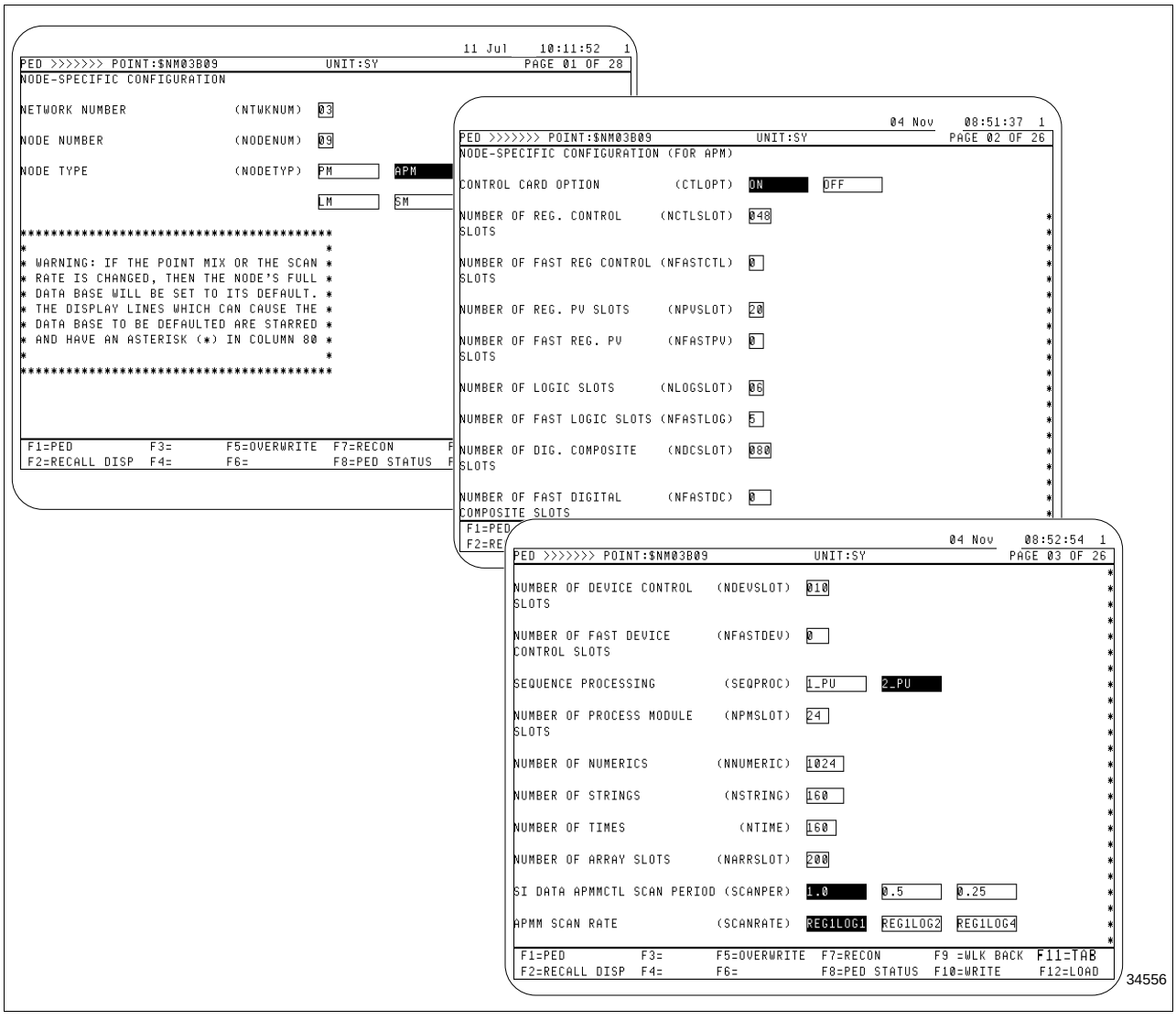
Description

This section describes what changes you can make to node-specific configuration, including point mix, and what changes require additional considerations.

Control partition

The control partition includes all PMM, APMM, or HPMM points assigned to do control. Figure 1 shows the first three pages of the PM and APM Node Specific Configuration PED, containing the control partition parameters (point mix, scan rate, and number of box variables). Figure 2 shows the first three pages of Node Specific Configuration for the HPM. In Figure 1, any item with an asterisk part of the control partition.

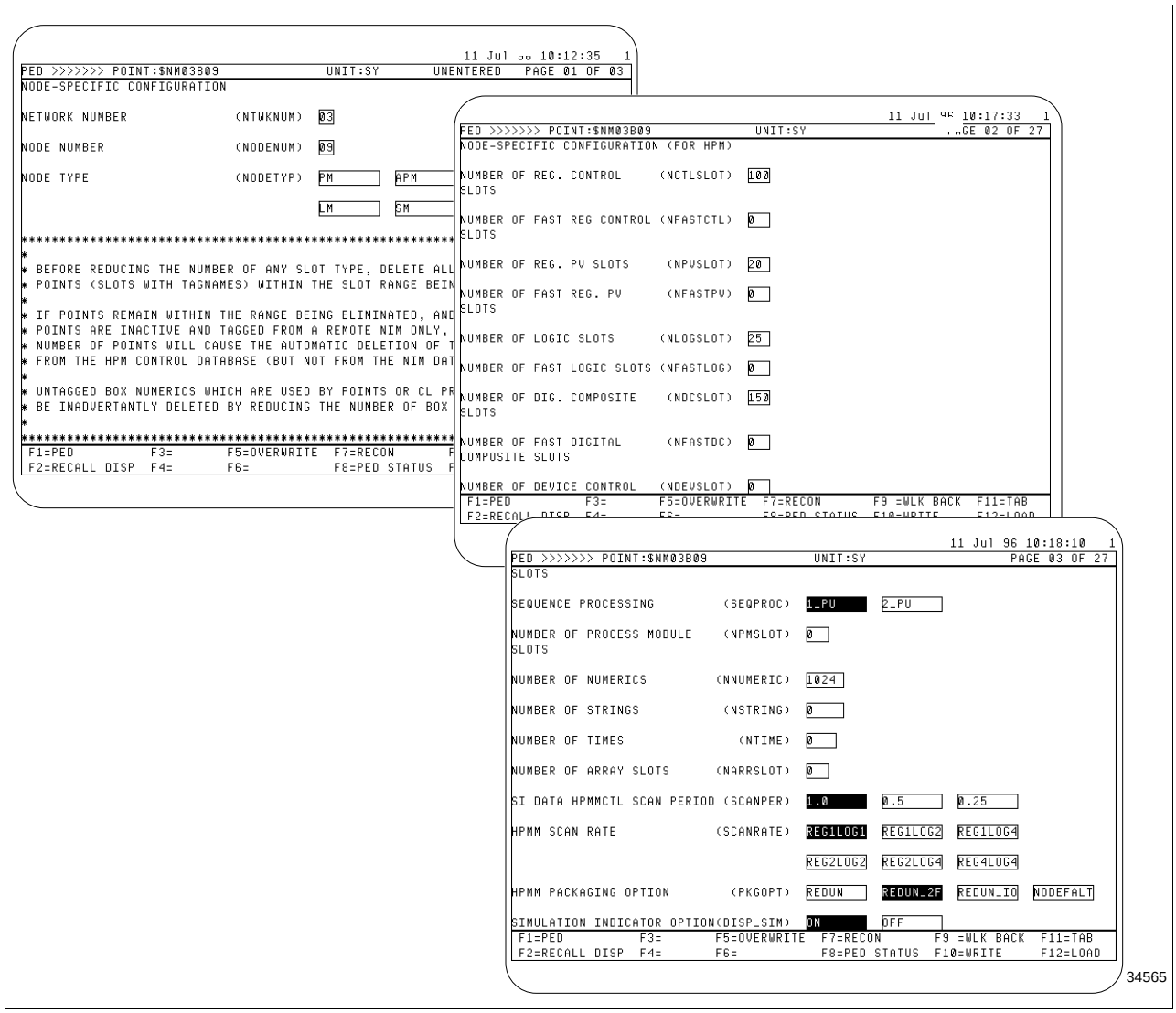
Figure 1 PMM/APMM Control Partition



Continued on next page

Changing Node Specific Configurations, Continued

Figure 2 HPMM Control Partition



Continued on next page

Effects of changing a PM/APM control partition

Changing a control partition in the PM or APM causes the PMM/APMM database to be initialized (cleared).

When the PMM/APMM database is cleared as the result of a control partition change, the following should be noted:

- The IOP database remains intact .
- The NIM database remains intact (unless you had previously deleted the points). Generally you will want to keep the NIM database intact to maintain the internal references that database items such as schematics, operating groups, and reports are using.

As shown at the top of Figure 1, a warning message appears on the first page of the PM/APM Node Specific Configuration PED. Essentially, this warning states that if the point mix or the scan rate is changed, the node's full database will be set to default values. If this occurs, the firmware must reallocate the PM/APM node's resources; consequently, all process data points will be lost and must be reloaded from Intermediate Data Files (IDFs) or from Exception Build Source Files (EBs).

Compare the warning on the first page of the PM/APM display in Figure 1 with the first page of the HPM display shown in Figure 2.

HPM rules

For the HPM, the control partition can be changed without having to reload the points. The HPM preserves existing control points, keeping the NIM database connections in sync with other configurations that you have already made.

The following rules will apply when making an HPM control partition change:

- The HPM must be in Idle to change any of the following:
  - point mix
  - scan rate, or
  - number of box variables (numeric, string, and time)
- An existing slot range number cannot be decreased below the number of points of that type that have already been built. If a slot range number is to be decreased below the number of points that have already been built, inactivate the points above the new slot range number that is to be entered, then delete those points. If the points to be deleted are process module points, set the sequence states for those points to off, then delete those points. The slot range number can now be decreased.
- The number of box variables (numeric, string, and time) cannot be decreased if it violates an array point mapping of the box variable.

Continued on next page

Database considerations

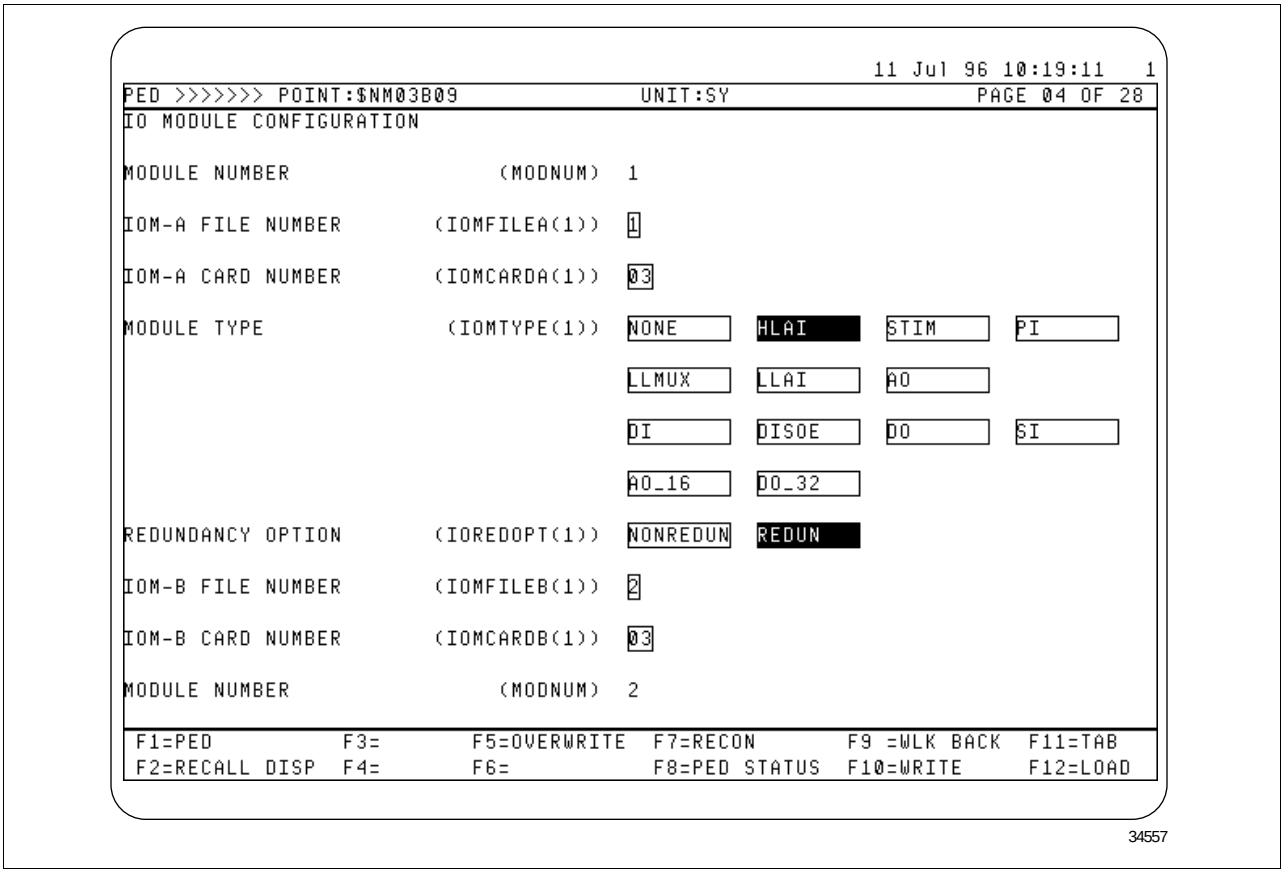
The point database is distributed between the HPM and the NIM and the point name resides in the NIM portion of the database. If the user fails to first delete a configured point that is to be removed *before reducing the point mix,* the point names will remain in the NIM. All configured points that are being removed because of a reduction in the point mix should first be deleted to ensure that the points (and point names) are removed from all NIMs on the UCN.

REFERENCE—Refer to the "Reconfiguration" section of the *HPM, APM, or PM Implementation Guidelines* manual for information on "Changing the Control Point Mix or Scan Rate."

Effects of changing IOP configuration

Changing an IOP partition in a PM, APM, or HPM *does not* cause the PMM/APMM/HPMM database to be initialized (cleared). An IOP configuration can be added while the node is running. Figure 3 shows the IOP configuration PED.

Figure 3 IOP Configuration



Control Partition Change Procedure

Purpose

An orderly procedure should be followed if you wish to change control partitions. The following procedure shows one approach to changing a control partition. Note that you may decide to follow a different approach; however, the following procedure provides ideas on what to consider.

ATTENTION

ATTENTION—For the PM and APM, the following procedure should be conducted off-line because you will lose process view and automatic control while the procedure is in progress.

Procedure

Table 1 Control Partition Change Procedure

Step	Action	Comments about this step
1	Checkpoint the node you are modifying.	This ensures a current checkpoint is available in case you need to discontinue the procedure and recover.
2	For the PM and APM: Backup all control partition points into IDFs (and EBs if desired). For the HPM: If you are reducing the number of slots for a point type, you must backup the control points to IDFs.	PMM/APMM control points need to be in the IDFs. The IOP points do not need to be in IDFs, since they maintain their previous configuration. The new point mix will require changing slot assignments.
3	Reconstitute the $NMuuBpp node specific configuration.	This provides a template for changing the control partition.
4	If the point mix change is to take away from one or more entity types, delete these entities from the system before loading the new point mix.	Refer to the Find Names module for information on searching the system for references to points you plan to delete.
5	Enter your point mix changes.	
6	Idle the node and load the new control partition (node specific configuration).	This step clears the PMM/APMM database (Reg control, Reg PV, logic, digital composite, device control, process module, numerics, strings, times, and array slot configuration).
7	For PM and APM only: Load all PMM/APMM entities using the IDFs (or EBs) created in Step 2.	This re-establishes your existing control point connections to IOPs and allows new point types to be built.
8	When the load is complete, checkpoint the new changes to the HM and to the FAST_LOAD cartridge *two times.*	Checkpointing twice ensures that your checkpoint file and backup checkpoint have the “new” configuration.

Clearing IOP Database

Purpose

Clearing an IOP database means removing the database currently in the IOP, which causes the IOP to return to a default null database.

Procedure

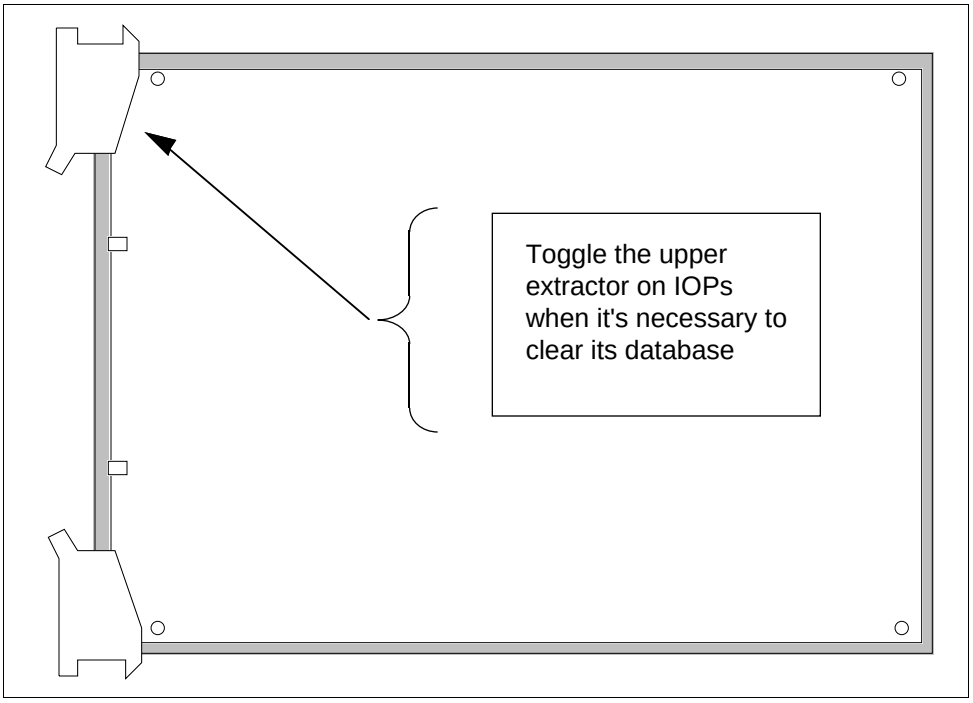
The procedure to clear an IOP database is

1. Toggle the extractor handle twice in quick succession. Figure 4 shows the extractor.

   RESULT: Toggling the extractor switch twice in quick succession makes the IOP "think" it has a checksum integrity problem. Because of that, the processor causes the IOP to return to a default null database.

2. Verify that the database shows as DBINVALID for the IOP (from the PM/APM/HPM Status display) to ensure that the database is cleared.

Figure 4 Clearing an IOP Database



Exception Build File Backups

Description

Exception Build (EB) file backups are recommended by Honeywell for backing up your database for several reasons:

- EB files are transportable between software releases (Note: Some editing may be required when going from release to release. This editing is for the purpose of adding any new parameters added to a point type.)
- EB files are transportable between sites and can be easily modified.
- EB files are ASCII files, which can be accessed on a PC (after going through a text file conversion.)

Preferred backup approach

One common way to create backup files is to do a Print System Entities to EB files. While this method is convenient, there is a scenario to consider on heavily loaded systems. On a heavily loaded system, the Data Entity Builder may timeout waiting for data and insert default values on your point types. The point types that are most susceptible to this problem are Application Module points with a large number of Custom Data Segments and HPM/APM/PM logic slot points.

To minimize the possibility that this will occur, you may want to consider reconstituting points (backbuilding) to Intermediate Data Files (IDFs), then building your EB files from IDF files.

Preferred procedure

Table 2 Preferred Procedure

Step	Action	Comments about this step
1	Determine how many points can exist in an IDF.	There is a limit to how many points can exist in an IDF, usually 255 entities.
2	Build a selection list for that IDF.	A selection list allows you to have an IDF for each point type. This also helps reduce NIM loading when backbuilding points.
3	Do a reconstitute multiple to IDF.	You would have to repeat this step for each point type IDF you had defined.
4	If an error occurs, locate the -UL file.	The unsuccessful entity list flags the points the system was not able to get. The usual reason for this is the system is too busy to get the data for you.
5	Rename the -UL file and use it again to try to get the points it had failed to get.	The -UL file is the “Unsuccessful List” of points that the system is not able to get.
6	After getting all your points, print entities from IDFs to an EB file.	In this step you have retrieved the current configurations for your points into an EB file.

Inactivating an IOP Slot

Purpose

The requirement to inactivate a point before modification or deletion prevents inadvertent changes to points that are in service, providing better system security.

Inactivating a point means that the point is deliberately taken out of service to make a change.

Some experienced users have found ways to make the inactivation requirement more convenient.

ATTENTION

ATTENTION—If an IOP point that had a tag built against it is deleted, the slot defaults to ACTIVE with the database initialized. This means that to load a new IOP point, the IOP must be in IDLE or the slot must first be set inactive.

With R321, R400 and later, all DI, DO, and AO points default to a point execution state of INACTIVE. Points can then be built while the IOP is in the RUN state.

Procedure

Table 3 Procedure to Inactivate an IOP slot

Step	Action	Comments about this step
1	Call up the PM/APM/HPM Detail Status display.	This step takes you to the displays to inactivate points
2	Select the IOP containing the point to be built or deleted.	
3	Select the SLOT SUMMARY target.	
4	Select the slot to be built.	If an untagged point is used, the point name for this slot is blank.
5	Select the PTEXECST target.	The point execution state is usually ACTIVE.
6	Select the INACTIVE target.	In this step you are taking the point out of service
7	Select ENTER target.	You can now return to the Data Entity builder displays and load the IOP point.

Checkpoints

Description

A checkpoint is a database save of a node's database that can be used later to restore the node. The term "checkpoint" comes from the idea that a snapshot of the database is taken at some particular point in time, or at some "check point." If it then becomes necessary to restart a node, the database save can be used to restore the node to the same state it was in at the "checkpoint" time. The files that are created when a database save is performed are referred to as "checkpoint" files.

The data for a process point is contained in a "distributed" database. This means that when a process point is loaded, some of the data defining the point is entered into the NIM's database because the NIM is the gateway to the UCN process boxes, while the remaining point data is entered into the database of the appropriate UCN process box. Furthermore, within the process box there exist databases in the PM/APMM/HPMM and the individual IOPs.

Because of the existence of a distributed database, whenever a checkpoint of a process box is requested, a checkpoint of the gateway for that box is performed as well to maintain consistency between the two node's checkpoint files. For the same reason, a checkpoint save operation can't be performed on an individual IOP; all modules for that node must be saved.

Initiating a checkpoint

There are two ways that a checkpoint save operation can be initiated:

1. To occur *automatically* on a scheduled basis,
2. On *demand* from one of the standard status displays.

Contents

Basically, there are checkpoint files for the NIM and checkpoint files for UCN process boxes. In general, the NIM checkpoint file contains description and configuration data, such as

- point names and their associated units
- sequence names
- logic descriptors
- EIP LCN IDs
- NIM-resident parameters
- UCN configuration information, such as the autocheckpoint state of the UCN nodes

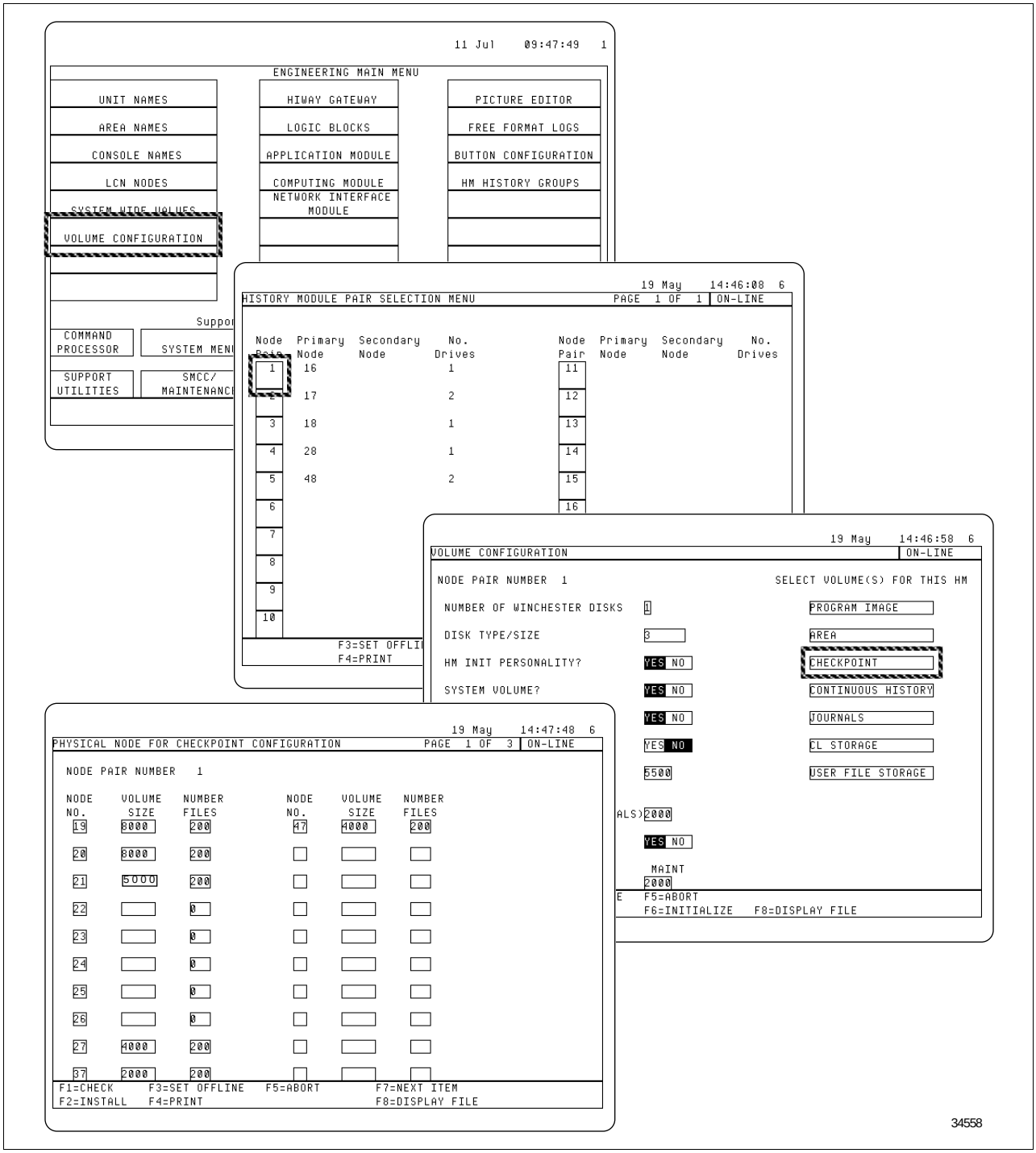
The process box checkpoint files contain all the information needed to restore the node's database if necessary. This includes configuration data as well as current process values for the APMM, and IOP databases.

Configuring Automatic Checkpointing

Description

Checkpoint Control is an HM function. An HM can be configured to do *automatic* checkpointing through NCF Volume Configuration.

Figure 5 Volume Configuration for Automatic Checkpointing



Continued on next page

Configuring Automatic Checkpointing, Continued

Configuration Overview

You configure automatic checkpointing support by entering the following information on the PHYSICAL NODE FOR CHECKPOINT CONFIGURATION display:

- Node number
- Volume size
- Number files

Node number

The LCN address of the node to be checkpointed must be entered into the NODE NO. column. The LCN nodes that can be automatically checkpointed by an HM include HGs, NIMs, AMs, and CGs; however, for automatic checkpointing functions it is not actually the physical nodes that are checkpointed, but the logical nodes contained within them. This subject will be addressed later in this course module.

Volume size

The disk space (in kilobytes) required for checkpoint files for the physical node should be entered under the column VOLUME SIZE. This number can be estimated using a formula.

Refer to Figure 4 for this example. Suppose that node number 21 is a NIM for UCN 1 and that there are a PM and an APM on UCN 1. The disk space required to support the checkpoint files for this UCN can be estimated as shown in Table 4a (R4xx) and Table 4b (R500).

According to the calculation, approximately 4688 kb on R400 and 4884 kb on R500 are required to provide ample space to checkpoint UCN 1 on the HM. You can round this value up and enter it for the VOLUME SIZE column for node 21.

Continued on next page

Table 4a R4xx Checkpoint Volume Size Calculation

Step	Calculate	Formula	Example
1	Sectors	Sectors = 7890 + (#PMs * 2812) + (#APMs * 8012) + (#LMs * 1912)	Sectors = 7890 + (1 * 2812) + (1 * 8012) = 18,714
2	Tracks	• Tracks = Sectors / 32 • Tracks = Integer value in Tracks + 1	• Tracks = 18,714 / 32 = 584.8125 • Tracks = 584 + 1 = 585
3	Kilobytes	Kb = (Tracks + 1) * 8	Kb = (585 + 1) * 8 = 4688

Table 4b R500 Checkpoint Volume Size Calculation

Step	Calculate	Formula	Example
1	Sectors	Sectors = 9615+ (#PMs * 2212) + (#APMs * 7712) + (#LMs * 2072) + (#HPMs * 14012) + (#SMs * 4012) + (#FSMs * 4012) + (4 future UCN nodes * 4012) + (# of FBO IOP's * 1410) where # = maximum planned	Sectors = 9615 + (1 * 2212) + (1 * 7712) = 19,539
2	Kilobytes	Kb = Sectors * 0.25	Kb = 19,539 * 0.25 = 4884

Continued on next page

Number files

The NUMBER FILES column expects a value between 8 and 404. The formula to calculate the number of files is

 # Files = (4 * #UCN nodes) + 4

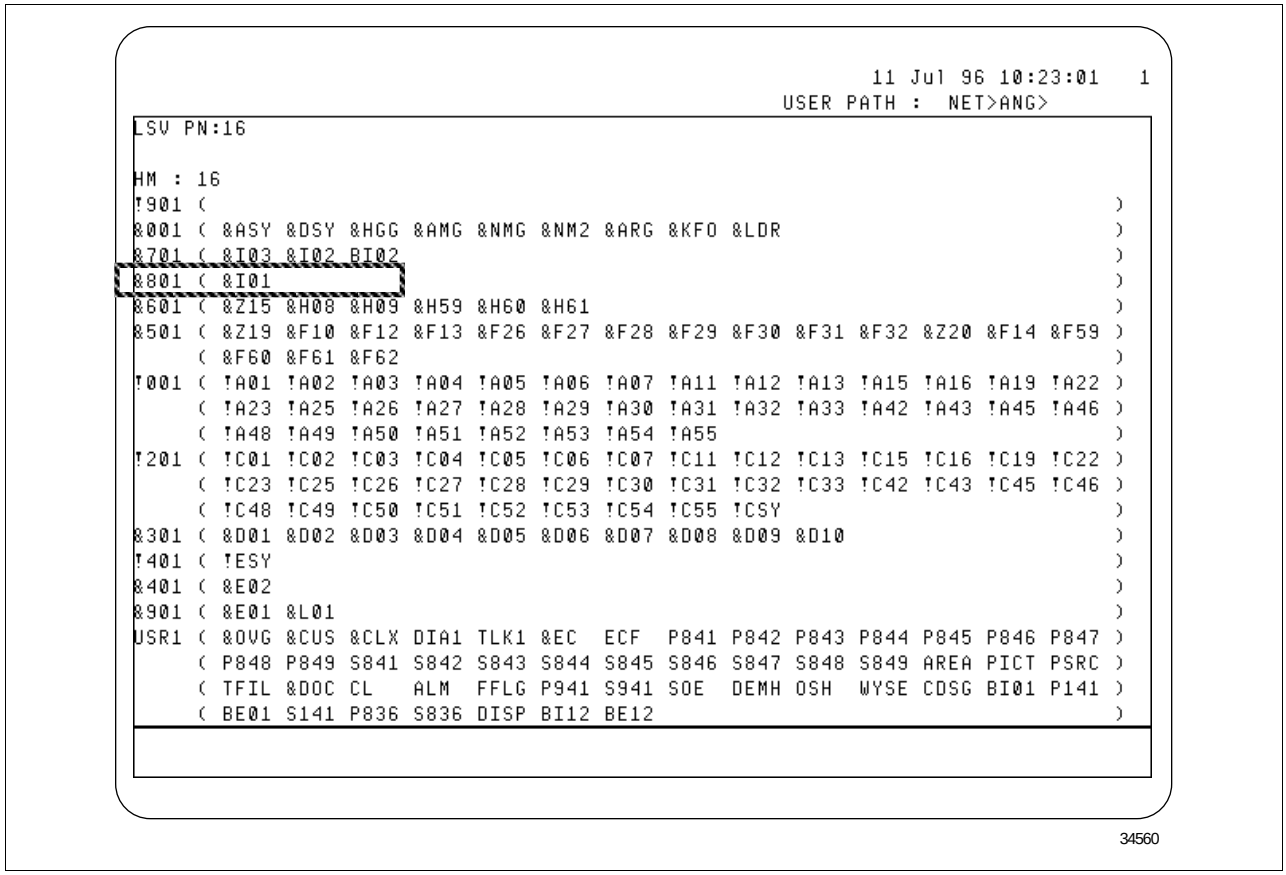
Four files are required for the NIM, and four files for each additional device on the UCN.

Listing of example configuration

If automatic checkpointing has been configured for a History Module under Volume Configuration, the volumes and directories needed to support automatic checkpointing are created when the HM is initialized.

Figure 6 shows the NIM checkpoint volume and directories that would be created during initialization of the HM to support the configuration described earlier.

Figure 6 HM Volumes and Directories



Continued on next page

HM checkpoint volumes and directories

Table 5 lists the HM checkpoint volumes and directories (Note: np = HM node pair).

The directories support the checkpointing of logical nodes within the physical nodes specified under Volume Configuration.

Table 5 Checkpoint Volumes and Directories

Volume	Description	Directories	Purpose
&5np	AM checkpoint volume	&Znn &Fuu	• One &Znn directory is created for each AM being checkpointed by this HM, where nn = LCN node address. This directory is called the Master Reference Directory, but currently contains no files. • One &Fuu directory is created for each unit assigned (through LCN NODES) to the AM being checkpointed, where uu = Unit number. The actual checkpoint files for each unit database are created here.
&6np	CG checkpoint volume	&Znn &Huu	• One &Znn directory is created for each CG being checkpointed by this HM, where nn = LCN node address. This directory is called the Master Reference Directory, but currently contains no files. • One &Huu directory is created for each unit assigned (through LCN NODES) to the CG being checkpointed, where uu = Unit number. The actual checkpoint files for each unit database are created here.
&7np	HG and PLCG checkpoint volume	&Inn	One &Inn directory is created for each HG/PLCG being checkpointed by this HM, where nn=Hiway number (1-20) as assigned to the HG/PLCG under LCN NODES of the NCF.
&8np	NIM checkpoint volume	&Inn	One &Inn directory is created for each NIM being checkpointed by this HM, where nn=UCN number (1-20) as assigned to the NIM under LCN NODES of the NCF.

Logical Nodes Used in Checkpointing

Description

It is important to realize that although checkpointing functions are assigned to an HM by physical node number through Volume Configuration, checkpointing is actually done on a logical node basis.

To clarify, consider a NIM that has been assigned to an HM for checkpointing purposes. When Checkpoint Control runs in that HM, a database save will be performed not only for the NIM but also for all the other nodes on the UCN associated with the NIM. This is because the logical node in the NIM is the UCN for which it serves as a gateway.

Logical nodes

For checkpointing functions, it is not actually the physical nodes that are checkpointed, but the logical nodes contained within them.

Logical nodes are databases that exist within a physical node. Table 6 summarizes the logical nodes that exist within the different physical nodes on the LCN.

Table 6 Logical Nodes in a Physical Node

Node	Logical Nodes Existing Within	Representation
HG	The Hiway that the HG connects to the LCN	Hiway number
NIM	The UCN that the NIM connects to the LCN	UCN number
AM	Each unit loaded in the AM is a logical node	Unit number
CG	Each unit loaded in the CG is a logical node	Unit number

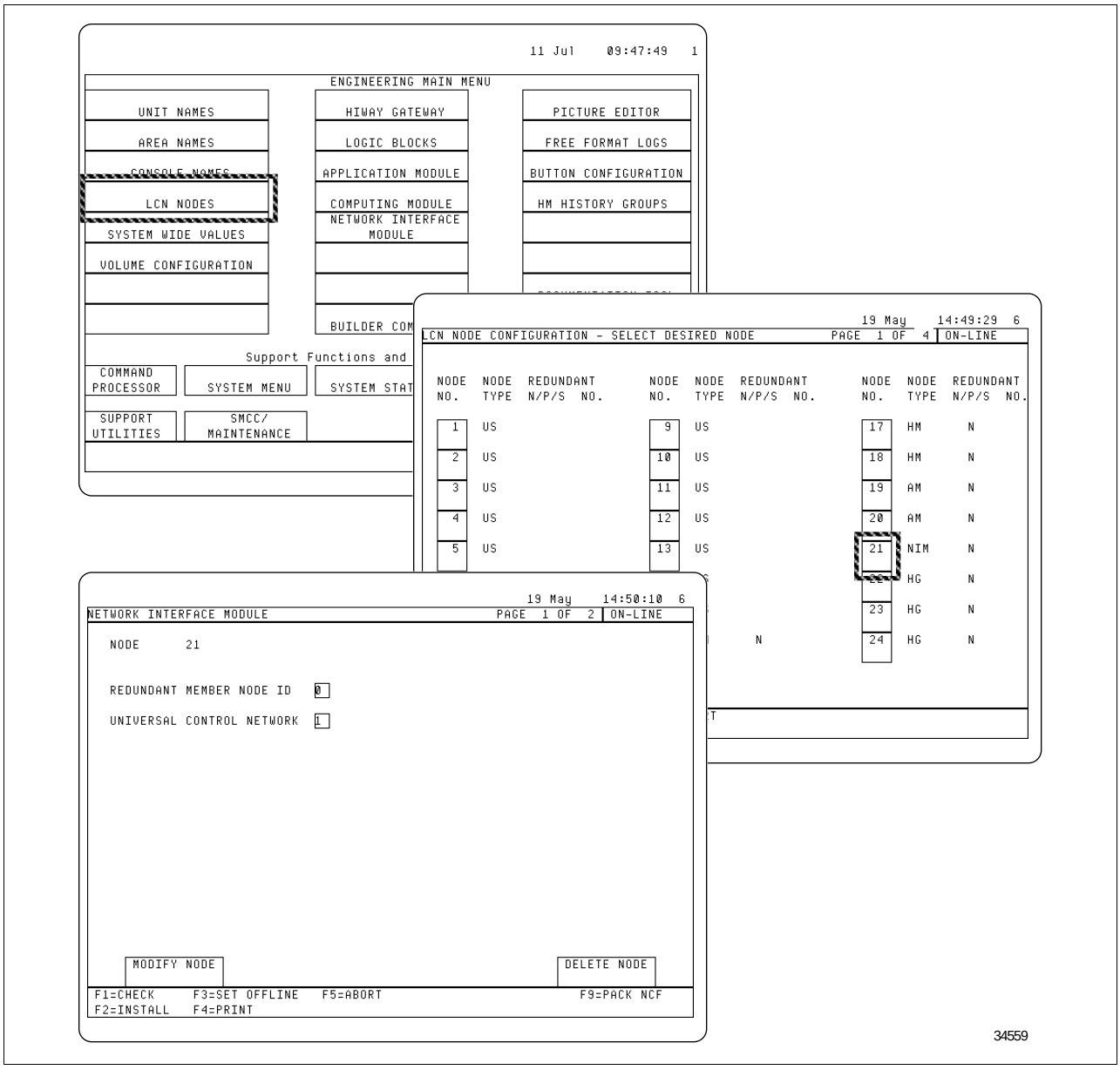
Configuring logical nodes

The specific logical nodes that exist within a particular physical node are determined by the information entered for each physical node under the LCN NODES portion of the NCF.

Figure 7 shows the LCN NODES definition for a NIM. Notice that the UNIVERSAL CONTROL NETWORK entered for this NIM is 1. This represents a logical node contained in this NIM. In the case of an AM or a CG definition, each unit assigned under LCN NODES represents a separate logical node.

Continued on next page

Figure 7 LCN NODES Definition and Logical Nodes



Enabling/Disabling AutoCheckpoint

Description

There are three levels of enabling and disabling automatic checkpointing. Automatic checkpointing can be disabled on a NIM basis, a process box bais, or a UCN basis. Targets on these displays allow you to control the current state of automatic checkpointing:

- Process Network Node Status,
- UCN Status, and
- UCN node Detail Status displays.

When Checkpoint Control runs in the HM

- it looks at the list of physical nodes that have been specified under Volume Configuration.
- For each physical node that has automatic checkpointing currently ENABLED, Checkpoint Control will then determine the “logical nodes” that exist within that physical node.
- Checkpoints are then performed for each logical node in the physical node. The checkpoint files are created under the appropriate volumes and directories on the HM.

ATTENTION

ATTENTION—It is recommended that automatic checkpointing be disabled during point loading operations. Because point data is loaded to both the NIM and process box databases, it is possible that a checkpoint operation could occur while the two are in an inconsistent state (data is loaded in one of the databases only). This situation, although unlikely, results in inconsistent checkpoint files.

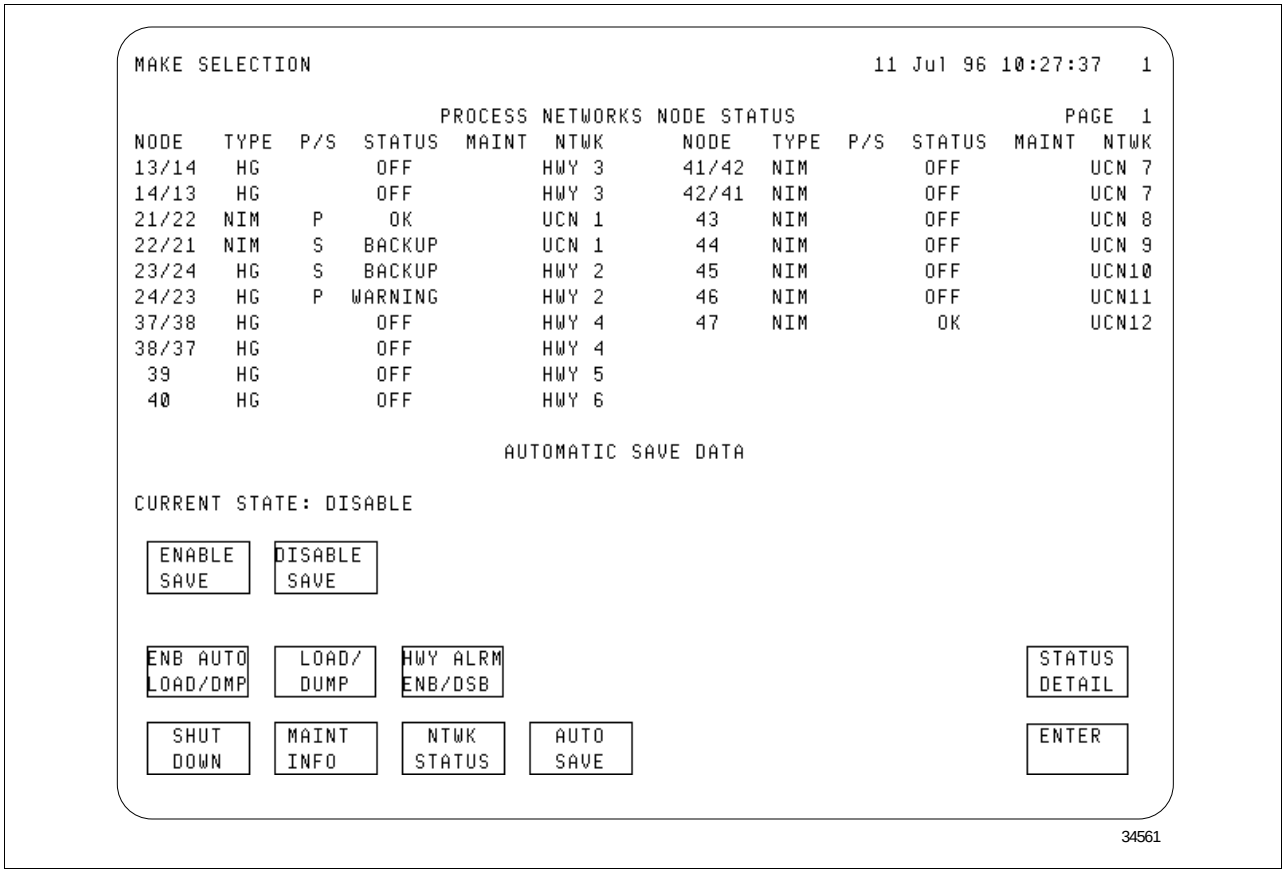
Disable NIM

The AUTO SAVE target on the Process Networks Node Status display is used to ENABLE or DISABLE automatic checkpointing from the NIM level.

If AUTO SAVE is DISABLED for a particular NIM, Checkpoint Control will not attempt to checkpoint the NIM or any of the nodes on the NIM’s UCN. AUTO SAVE defaults to the value stored in the checkpoint file used to load the NIM.

Continued on next page

Figure 8 Process Networks Node Status—AUTO SAVE



Continued on next page

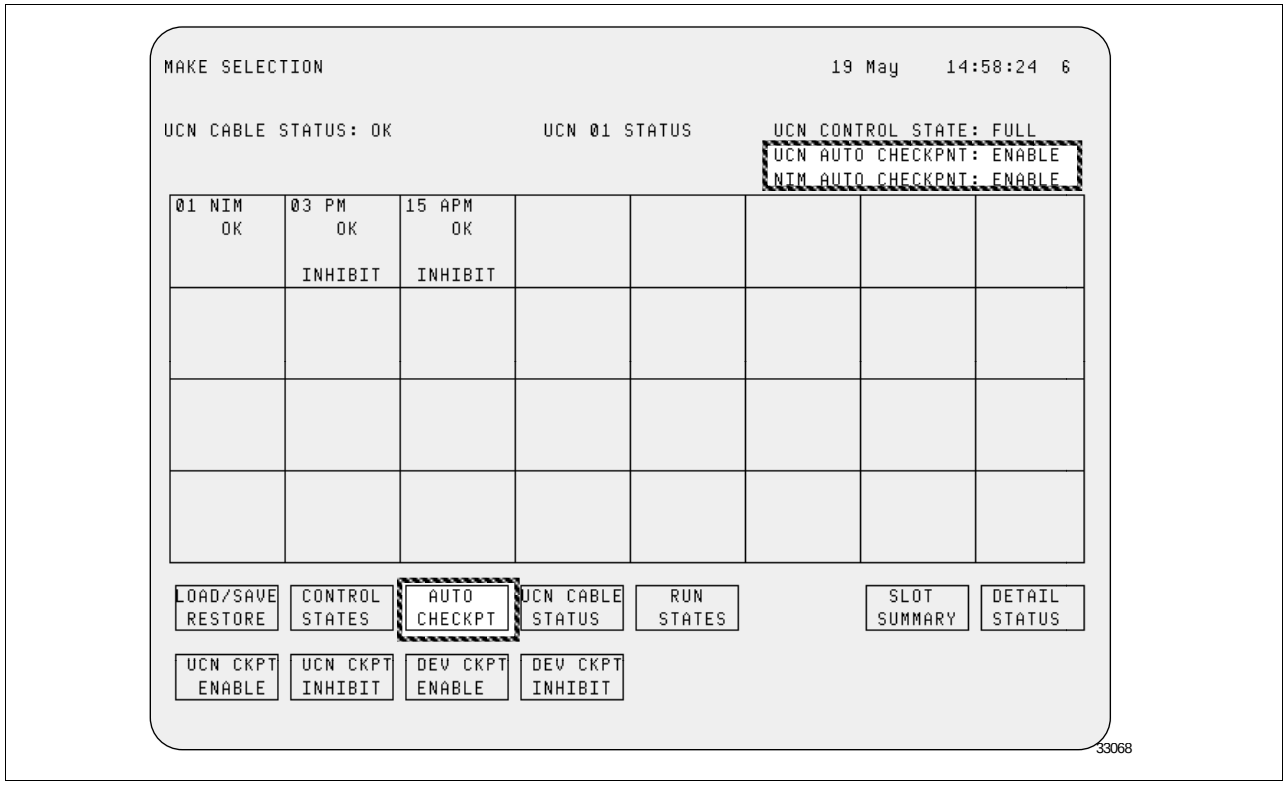
Disable UCN or UCN node

Targets on the UCN status display allow you to ENABLE or INHIBIT automatic checkpointing for the UCN, or for a specific node on the UCN. The UCN status display and the targets associated with automatic checkpointing are shown in Figure 8.

The UCN AUTO CHECKPNT state is displayed in the upper right-hand corner of the display. If the UCN autocheckpoint state is INHIBIT, Checkpoint Control will not attempt to checkpoint any of the nodes on the UCN. The UCN autocheckpoint state does not affect automatic checkpointing of the NIM. If the NIM's AUTO SAVE state is ENABLE, Checkpoint Control will checkpoint the NIM as previously explained. The UCN autocheckpoint state defaults to INHIBIT on a NIM load.

The autocheckpoint state for each node on the UCN is displayed on the UCN Status display after the AUTO CHECKPT target is selected. Checkpoint Control will not checkpoint a UCN node whose autocheckpoint state is INHIBIT. When an APM goes to the ALIVE state, it's autocheckpoint state defaults to INHIBIT, where it remains once loaded; however, on a NIM load, each node's autocheckpoint state will be restored to the state that was saved in the NIM's checkpoint file.

Figure 9 UCN Status Display—AUTO CHECKPT



Continued on next page

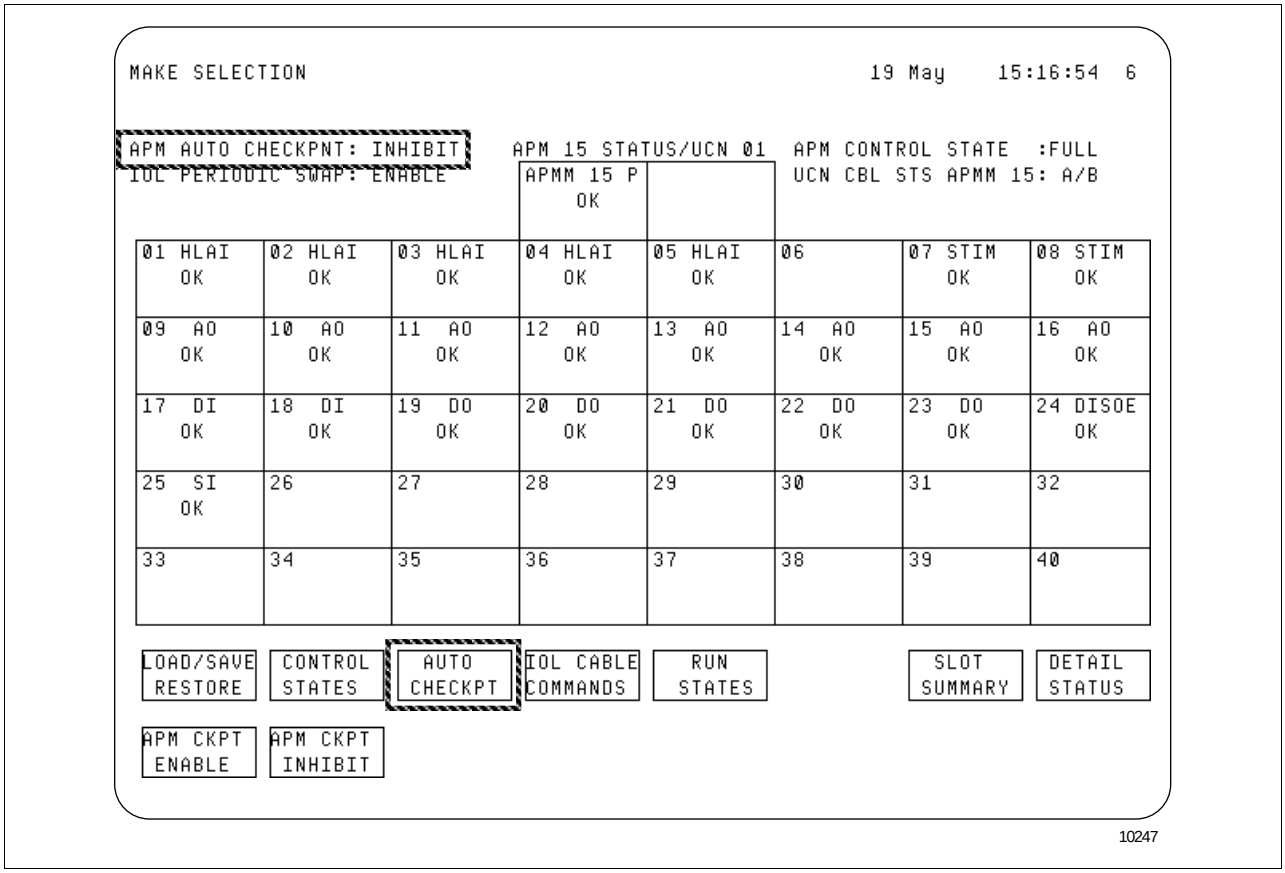
Enabling/Disabling AutoCheckpoint, Continued

Disable UCN node

From the Detail Status display for a UCN node, you may ENABLE or INHIBIT the autocheckpoint state for the node. The current node AUTO CHECKPNT state is shown in the upper-left corner of the display as seen in Figure 10.

As explained in the previous section, Checkpoint Control will not attempt to checkpoint a UCN node with an autocheckpoint state of INHIBIT.

Figure 10 APM Detail Status Display—AUTO CHECKPT



Continued on next page

Enabling/Disabling AutoCheckpoint, Continued

Autocheckpointing states

The three levels of automatic checkpointing, the status displays to use for viewing and modification, and checkpoint state descriptions are summarized in Table 7.

Table 7 Autocheckpoint States

Level	Status Display	State	Description
NIM	Process Network Node Status display (R400 = Gatways and Interfaces)	ENABLE	Checkpoint Control will attempt to checkpoint the NIM, and then proceed to attempt checkpointing of devices on the UCN based on • UCN checkpoint state, and • individual device checkpoint states.
		DISABLE	Checkpoint Control will not attempt to checkpoint the NIM or any devices on the UCN.
	UCN Status display	ENABLE	On R430 and later, the UCN Status display includes the NIM's checkpoint enable/disable state.
		DISABLE	
UCN	UCN Status display	ENABLE	Checkpoint Control will attempt to checkpoint any devices on the UCN whose checkpoint state is ENABLE.
		INHIBIT	Checkpoint Control will not attempt to checkpoint any devices on the UCN.
Device	UCN Status display or Device Status display (HPM, APM, PM, LM, SM)	ENABLE	Checkpoint Control will attempt to checkpoint this device.
		INHIBIT	Checkpoint Control will not attempt to checkpoint this device.

Continued on next page

Default autocheckpointing states on a node load

The default automatic checkpointing states on a node load (of a node with no active primary) for the different levels are shown in Table 8.

Table 8 Default Autocheckpoint States on Node Load

Type of Node Load	Checkpoint Level	Auto Checkpoint Default State
Primary NIM	NIM	Defaults to checkpoint state in the checkpoint file used to load.
	UCN	INHIBIT
	Device	Each device's checkpoint state will default to the state as it was saved in the checkpoint file used to load.
Primary Device	NIM	Unaffected by device load.
	UCN	Unaffected by device load.
	Device	INHIBIT

Autocheckpoint Schedule

Description

Checkpoint Control runs periodically in HMs that have been assigned checkpointing responsibilities. The default interval is every four hours. The offset value defines how long after midnight to begin the first checkpoint cycle. The default offset value is 30 minutes; therefore, if default offset and interval values are in effect, Checkpoint Control will run at the following times:

12:30 am	8:30 am	4:30 pm
4:30 am	12:30 pm	8:30 pm

Offset and cycle interval time for automatic checkpointing are configurable by HM. This is done by changing the values in the HM's Processor Status Data Point (PSDP) parameters TIMEBASE(1) and TIMEBASE(2).

Automatic checkpoint PSDP parameters

A Processor Status Data Point (PSDP) exists in each node on the LCN. The format of the PSDP is $PRSTSnn where nn is the LCN address of the node. This point has parameters that provide "processor status" information pertaining to the node. The HM has five automatic checkpointing PSDP parameters that are described in Table 9.

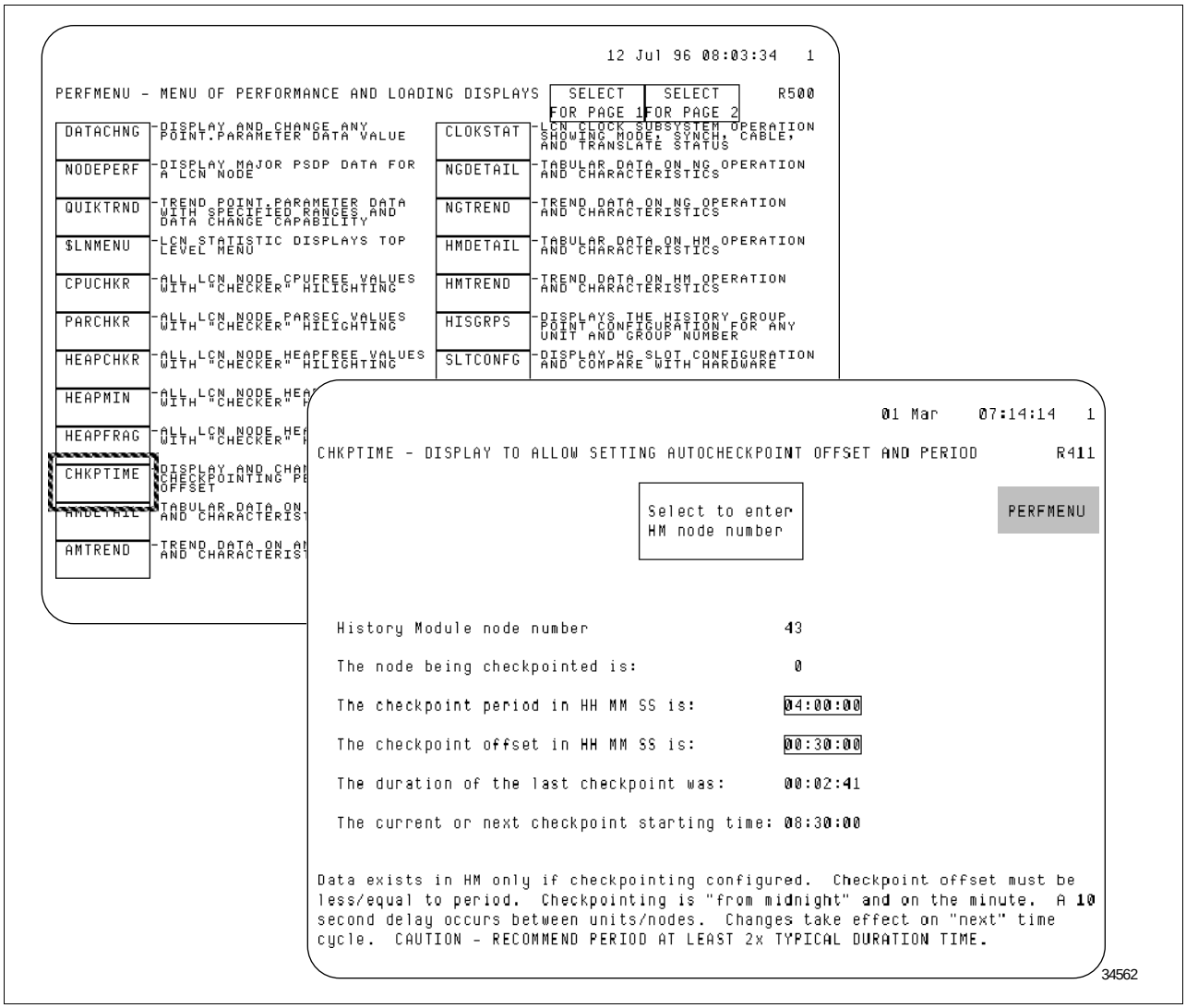
Table 9 Checkpoint PSDP Parameters

Parameter	Access	Description
TIMEBASE(1)	READ/WRITE	The HM's current interval. Default is four hours. The value of this parameter must be greater than or equal to OFFSET (TIMEBASE(2)), and must evenly divide into a 24-hour period.
TIMEBASE(2)	READ/WRITE	The HM's current offset value. Default is 30 minutes. The value of this parameter must be less than the interval (TIMEBASE(1)).
TIMEBASE(3)	READONLY	The time it took to complete the last checkpoint cycle.
TIMEBASE(4)	READONLY	The time of the last or current automatic checkpoint. This value is updated each time a new interval or offset is entered.
NODENUM	READONLY	The LCN address of the node currently being checkpointed. If Checkpoint Control is not running, this parameter contains a value of zero.

These PSDP parameters can be changed by using the Toolkit display CHKPTIME as shown in Figure 10. Also, any of these parameters can be included in a custom display by referencing the PSDP for the node and the desired parameter. For example, to display the time it took to complete the last checkpoint for HM 16, reference $PRSTS16.TIMEBASE(3).

Continued on next page

Figure 11 CHKPTIME Display



Automatic Checkpoint Errors

Description

If a UCN node other than the NIM cannot be successfully checkpointed, checkpointing of the other nodes proceeds; however, if an error occurs during a checkpoint of the NIM database, no other UCN nodes are checkpointed.

Corrupted file

The first time Checkpoint Control runs in an HM configured for automatic checkpointing, a file is created under the local volume (!9np) that stores the current interval and offset values. These values will contain default values until changed. The name of the file is Cnp_CPNT.MM, where np=node pair number of the HM.

If the checkpoint control file ever becomes corrupted, it is renamed to CP_RNzzz.MM where zzz is an integer between 0 and 30. A new Cnp_CPNT.MM file is then created. The values contained in the new file depend on when the corruption was detected:

- If detected while the interval and offsets were being changed, the new file will contain the newly input values.
- If detected during an HM startup, the new file will contain the default values.

Error messages

If an automatic checkpoint operation is not completely successful, a software error message is reported to the System Error Journal and the Real Time Journal. The software task that reports these errors is (the slow command handler) UG$_CH_SLOW, the NIM processing task responsible for checkpoint and restore operations. These messages contain a field in the following form:

28, 94, nn, xx

nn = UCN node number, 1 through 64
xx = type of error as listed in Table 10.

Continued on next page

Table 10 Auto Checkpointing Errors

xx Value	Meaning
05	Checkpoint volume too small to create the checkpoint file.
07	&Inn checkpoint volume not found on an HM.
41	UCN communication problem—NIM can't send checkpoint request to the UCN node.
55	The UCN node is in the IDLE state and cannot be autocheckpointed. The database may not be valid.
65	I/O link resource error. Not enough resources on I/O link to checkpoint IOM.
66	UCN node is in softfail state, so cannot respond to checkpoint request.
68	Checkpoint file on HM is wrong version or wrong revision. Proper upgrade of checkpoint files is needed.
69	Checkpoint request rejected by the UCN node. Contact Honeywell TAC.
73	Communication problem on the I/O link.

Demand Checkpointing

Description

A checkpoint save operation can be initiated on demand from the standard Process Network Node Status display. You can create the checkpoint files on removeable media (select Alternate Source), or the History Module (select Default Source) if the necessary directories exist there.

Recommendations

Recommendations for demand checkpointing to removeable media follow:

- Use initialized cartridges (Floppies cannot be used for checkpoints on R500 and later). The Command Processor Create Volume command (CR) can be used to format the media.
- Perform demand checkpoints to removeable media rather than copying HM checkpoint files to removeable media. Especially in the case of floppies where at least two are required, this will ensure that the data is split correctly between media and therefore useable for loading or restoring a node.
- If using cartridges, demand a checkpoint twice for each node. This gives you two checkpoint files for each node.
- Use demand checkpoints to removeable media as a backup to the checkpoints on your HM. Maintain several backups onsite and rotate them on a scheduled basis. Keep one backup offsite.

ATTENTION

ATTENTION—If the necessary directories are not present on the removeable medium used for a checkpoint save, they are created when the checkpoint is demanded.

ATTENTION

ATTENTION—To avoid problems, do not demand a checkpoint while a point build operation is in progress. Because point data is loaded to both the NIM and process box databases, it is possible that the demanded checkpoint operation could occur while the two are in an inconsistent state (data is loaded in one of the databases only). This situation, although unlikely, results in inconsistent checkpoint files.

Checkpoint files

Checkpoint files

We have already discussed the volumes and directories needed to support checkpointing operations. Now let's look at the different files created in those directories when a checkpoint is performed. Table 11 lists the checkpoint files related to the NIM and the UCN.

Table 11 NIM Checkpoint Files

Filename	Description
NM0uuMAS.CP	NIM master file
NM0uuNMr.CP	NIM data file
HPMuunnM.CP	HPM master file
HPMuunnr.CP	HPM data file
APMuunnM.CP	APM master file
APMuunnr.CP	APM data file
PM0uunnM.CP	PM master file
PM0uunnr.CP	PM data file
LM0uunnM.CP	LM master file
LM0uunnr.CP	LM data file
	where : • uu = UCN network number (1-20) • nn = UCN node number (1-32) • r = revision number (1-2)

File revisions

The NIM- and box-resident checkpoint files listed in Table 11 have a revision number (r) before the file extension of .CP. This revision number alternates between 1 and 2, changing each time a new checkpoint is created. When a checkpoint save is requested for a device, the new checkpoint file will overwrite the older revision checkpoint file (if it exists) and the master file for the device will be updated with the "new" revision of the most recent checkpoint file. When a restore operation is requested, the revision number in the master file is used to identify the most recent checkpoint file that will be used for the restore.

Number files calculation in Volume Configuration

Refer back to the Number Files calculation given in the section on configuring an HM for automatic checkpoint support. The three files for each node that are required for checkpoint support are

- one Master file, and
- two revisions of the node's data file.

Save/Restore Error Messages

Description

The UCN Status display shows current status information for each device on that UCN. Each node is represented by a square that contains the UCN device type (NIM, HPM, APM, PM, LM, SM), primary and secondary device addresses and their current states.

When a command is initiated from the status display, such as a SAVE or RESTORE operation, auxiliary information concerning the status of the operation is displayed at the bottom of the affected device's status box.

Error messages

The following table lists statuses displayed when a requested LOAD, SAVE, or RESTORE operation cannot be completed.

Table 12 Status on UCN Status Display

Text	Meaning
ACC_LVL	Invalid access level
BAD_CMD	Invalid command
BAD_RQST	Bad request
BAD_STAT	Node illogical (in a bad state)
CHECKSUM	Bad checksum
CNGF_MIS	Configuration mismatch
CTL_STAT	Invalid control state
DATA_ERR	Data inconsistency
DB_BUSY	Checkpoint blocked by Data Access
DB_INVLD	IOP database invalid
DV_IN_SF	Device in softfail error
FILE_ACC	File access error
FILE_FMT	Invalid file format
FILE_REV	Invalid file revision
FW_REV	Firmware software mismatch
INHIBIT	Checkpoint inhibit error
INV_NODE	Invalid node number
INV_STAT	Incompatible (invalid) state

Continued on next page

Error messages, continued

Table 12 Status on UCN Status Display, (continued)

IOL_COMM	I/O Link communication error
IOL_RSRC	I/O Link resource error
LOST_DAT	Missed packet
MEM_OVRF	Memory overflow
NO_CKPT	Checkpoint data not present
NO_ROOM	No room on device (media) for data
NO_RSPNC	No response
NODE_ASN	Invalid node assignment
NODE_BSY	Node busy
NONEXIST	Target does not exist
NOT_ALIV	Not alive
NOT_ATMP	Operation not attempted
NOT_CNFG	Target not configured
NOT_IDLE	Target not in IDLE
NOT_RUN	Target not in RUN
PARM_ERR	Parameter access error
PI_FILE	Personality image file
TGT_ACTV	Node active
UCN_COMM	UCN communication error
UCN_DA	Local database access error
UNK_VOL	Volume not found (unknown volume)

IOP Database Validity

Definition of Valid/Invalid

Definition

As mentioned earlier, the UCN process box database consists of data stored in the HPMM/APMM/PMM and the IOPs. An IOP database is considered VALID on a restore operation if all slots are restored successfully. If all slots can't be restored, the database is marked as INVALID. This will happen if the IOP database is cleared, when the IOP is loaded with the default database, and in some rare cases where the IOP detects a database problem.

Checkpointing and INVALID IOPs

If a checkpoint operation is performed on an HPM, APM, or PM that has an IOP marked as INVALID, the INVALID IOP database is not saved in the checkpoint file.

If a previous checkpoint file exists with a VALID database for that IOP, the IOPs database is copied from the previous file into the new checkpoint file being created. If this checkpoint is later used in a restore operation, the IOP should come up with a VALID database.

If there is no previous checkpoint file (as in the case of checkpointing to a cartridge for the first time), there will be no data stored for the INVALID IOP in the new checkpoint file. If this checkpoint is later used in a restore operation, the IOP will come up with a (default) database marked INVALID. From the Node Status display, this IOP will show a message of NO CKPT rather that RESTORED.

Purpose of INVALID designation

The purpose of marking an IOP database as INVALID is to keep you from inadvertently saving an IOP database that can't be used to restore the IOP. For example, if the IOP database was cleared for some reason and an autocheckpoint was then performed, the resulting checkpoint file would contain default data for the IOP and the previous "good" data for that IOP would be lost. Marking the IOP as INVALID and not including it in a checkpoint file, forces you to confirm (validate) that the default database in the IOP is the one you want. This way, a default IOP database will never be written to a checkpoint file unless you perform the steps to VALIDATE it.

Continued on next page

Steps to validate an INVALID IOP

Steps to VALIDATE an INVALID IOP are listed in the following table.

Table 13 Validating an IOP Database

Step	Action	Result
1	From the HPM/APM/PM Status display, select the RUN STATES target.	New targets appear at the bottom of the display. Note: Beginning with R410, the target VALIDATE IOP DB will appear only in ENG key access.
2	Select the VALIDATE IOP DB target.	The current status of each IOP's database is displayed.
3	Select the IOP module with the INVALID status.	
4	Select the ENTER target.	The IOP's database changes from INVALID to DB_VALID.

Restoring from a Checkpoint

Overview

Use of checkpoint files

Checkpoint files can be used to restore a node in the case where the node does not have a database. When a checkpoint save operation is requested, files are updated for the NIM if you requested a save of the NIM, or for the NIM and a process box if you requested a process box to be saved. When a restore operation is requested, only the files necessary to restore that node's database are loaded.

Two revisions of the checkpoint file may exist on the media you are using to restore from. The restore operation will use the node's master file in order to determine the revision number of the most current checkpoint file to use.

Restore operations

Table 14 lists required states for a restore operation.

Table 14 Required State for Restore Operations

Type	Note	Required State
NIM	The NIM can't be "restored". The checkpoint is used as part of a LOAD operation.	FAIL, QUALIFIED, POWER ON (NOTE: a NIM in the ISOLATED state may also be loaded)
PMM APMM HPMM	The primary HPMM, APMM, or PMM can be restored, or all modules of the node.	Must be IDLE.
IOP	Individual IOPs may be restored.	Must be IDLE.

Redundant NIMs

When loading redundant NIMs, make sure to complete loading of one of the NIMs as primary before initiating a load on the secondary. If both nodes are loaded concurrently, both may attempt to be primary and one of them may fail.

Using an "old" checkpoint file

Suppose that the most current checkpoint file on the HM is corrupted for some reason. In order to attempt a restore using the "older" revision checkpoint file, it is necessary to rename (or copy) the older revision to have the same name as the current corrupted revision (the corrupted file would need to be deleted or renamed first). This is necessary because the restore operation will use the file revision listed in the master file as the most current.

If you want to use a previously checkpointed file, you can use the method described above. Please note, however, that this is *not* a Honeywell-recommended procedure. If you routinely checkpoint to removeable media as part of your backup procedures (as suggested under "Demanding a Checkpoint"), the current checkpoint file will be available.

Continued on next page

Restoring from removable media

When restoring from removable media, the cartridge or floppy being used must not be write-protected. This is because the restore operation opens the checkpoint files for exclusive (write) access in order to exclude any other tasks from writing to the files during the restore operation. If the removable media is write-protected, the restore operation will fail.

Point Fragments

Overview

Distributed database

Remember from the discussion on checkpointing that the UCN database is distributed between the NIM and the process boxes on the UCN. This means that part of the data defining a process point is stored in the NIM's database and part of the data is kept in the database of the process box.

This makes it very important that the NIM and UCN node databases remain consistent. A good understanding of the following concepts is necessary in order to avoid introducing inconsistencies between the NIM and process box databases:

- Point building
- Saving and restoring node databases
- Point deletion.

Definition of point fragments

Point fragments occur when part of a point exists in one node, but the other part does not exist in the other node. This means that a complete point definition no longer exists for that point, only a "fragment" of the point remains. Point fragments are sometimes referred to as ghost points because they may exist in one node and affect the system, even though you don't see them in the other node.

Types of point fragments

There are two types of point fragments that can exist on the UCN:

- A NIM point fragment exists when the NIM database contains data for a point, but the process box database does not.
- A UCN node point fragment exists when the process box database contains data for a point, but the NIM database does not. In this case there is no tagname associated with the point because the NIM is not aware that the point exists.

Continued on next page

Effects of point fragments

Point fragments can affect the UCN in several ways as shown in Table 15.

Table 15 Effects of Point Fragments

Fragment type	May affect ...	Because ...
NIM	Loading new points	• When attempting to load a new point into a slot that is actually empty at the process box level, the NIM thinks a point already exists for the slot because it has a tagname and data for the slot. • The NIM shows the point form of an IOP point as FULL, while the IOP database shows it as a COMPONENT point.
Process box	Peer-to-peer load	The process box point fragment will add to the peer-to-peer communication load of the UCN (even if the point is INACTIVE), if the point was originally configured for peer-to-peer. This is because the point fragment will still be included in the input scan table and output store cycles.
	Operator view of control	• An operator may see output oscillation on a point if a process box point fragment is configured to output to the same point as another valid point. The operator can see the valid point (from a Group or Detail display), but not the point fragment because there is no tagname in the NIM database for that point. • An operator may see a point acting like a secondary in a cascade loop (setpoint changing and output reacting) even though he sees no primary point acting as input.

Causes of Point Fragments

Effect of NIM LOADSCOP on point loading and deleting

The parameter LOADSCOP is part of the NIM network data point definition ($NMuuNbb). LOADSCOP defines the "load scope" of a load or delete operation on the UCN. There are two choices for LOADSCOP:

Table 16 LOADSCOP Choices

Parameter Value	Description
NIMOnly	On a load or delete operation, only the NIM's database is updated.
NIMAndPM	On a load or delete operation, both the NIM and the UCN node's database are updated.

When to configure LOADSCOP as NIMOnly

A LOADSCOP value of NIMOnly can be used for initial point building when the NIM is available, but the process box is not yet on line. This allows the establishment of points in the NIM database so that other configuration tasks can be performed, such as building schematics, free format logs, etc. Once the process box is in place and operational, the NIM LOADSCOP can be changed to NIMAndPM, and the points reloaded. This establishes the box-resident point data into the process box database.

When to configure LOADSCOP as NIMAndPM

The default value for LOADSCOP is NIMOnly; however, you will probably want to set LOADSCOP to NIMAndPM during normal operations, and change it to NIMOnly under special circumstances. Incorrect use of the LOADSCOP parameter can result in discrepancies between the NIM and process box databases.

Review of save and restore concepts

Remember that when a save (checkpoint) operation is requested for a process box, the gateway to that box is checkpointed as well. This is done in order to maintain consistency between the checkpoint files because point data is distributed between the two databases.

A restore operation then uses a checkpoint file to restore a single node. For example, if it is necessary to restore a process box, but not the NIM, the process box database will be restored with the data found in the process box checkpoint file. The NIM, however, continues to run with it's current database because it is not necessary to restore the NIM. If the checkpoint file used to restore the process box is not as "current" as the NIM's database (changes have been made since the checkpoint was taken), this will result in discrepancies between the NIM and process box databases.

Understanding the differences between save and restore operations and the files used by each, helps to explain how inconsistencies between gateway and process box databases can occur.

Continued on next page

Review of autocheckpoint disable levels

If you are relying on autocheckpointing to provide current checkpoint files for your UCN, keep in mind the three levels that must be enabled for a UCN node to be checkpointed.

For example, suppose it became necessary for you to reload the NIM. As listed in Table 8, the autocheckpoint state for the UCN defaults to INHIBIT on a NIM load. If you forget to change the checkpoint state back to ENABLE, the next time autocheckpointing is initiated for the UCN, checkpointing will be performed for the NIM (if NIM state is ENABLE) but not for any other nodes on the UCN. If database changes are then made and the NIM is checkpointed, the NIM checkpoint file will contain the new information but the process box checkpoint file will not be updated.

Table 17 shows the results of the different combinations of autocheckpoint states for the different levels of autocheckpointing.

Table 17 Autocheckpoint State Combinations

NIM	UCN	Process boxes(all)	Nodes that will be checkpointed
ENABLE	ENABLE	ENABLE	NIM Process boxes
ENABLE	ENABLE	INHIBIT	NIM only
ENABLE	INHIBIT	ENABLE	NIM only
ENABLE	INHIBIT	INHIBIT	NIM only
DISABLE	ENABLE	ENABLE	None
DISABLE	ENABLE	INHIBIT	None
DISABLE	INHIBIT	ENABLE	None
DISABLE	ENABLE	INHIBIT	None
DISABLE	INHIBIT	INHIBIT	None

Understanding how these levels affect the checkpoint files created during an automatic checkpoint will help you avoid creating inconsistent checkpoint files that may cause point fragments if used to load a node.

Continued on next page

Causes of Point Fragments, Continued

Causes of NIM point fragments

Following are some example situations resulting in the creation of point fragments in the NIM:

- New points are loaded to the NIM and APM. Before a checkpoint is taken, the APM is shutdown and reloaded with a previous checkpoint that doesn't contain the new point. The NIM will retain its data for the new point, but the APM will not.
- A point is deleted from the NIM and APM databases. Before a checkpoint is taken, the NIM is shutdown and restored with a checkpoint that still contains the deleted point. The NIM database will have data for the "deleted" point, but the APM will not.
- A point is loaded while the LOADSCOP of the NIM is NIMOnly. The NIM's database will be updated with the new point information, but the APM database will not.
- If the point mix in the APMM is modified without deleting configured points that may be affected, the NIM will retain it's data for those points while the APMM will not.
- If an IOP database is cleared (by toggling the top extractor), its database no longer contains the original point data, but the NIM still has the tagnames for the points that were built in the slots of that IOP.

Causes of process box point fragments

Following are a few example scenarios that create process box point fragments:

- New points are loaded to the NIM and process box. Before a checkpoint is taken, the NIM is shutdown and then reloaded with a checkpoint that doesn't contain the new points. Because the process box was not restored, it retains its data for the new points. The NIM, however, lost all data associated with those points, including the tagnames.
- A point is deleted, and so removed from the online NIM and process box databases. Before a checkpoint is taken, the process box is shutdown and restored with a checkpoint that still contains the deleted point. The point data would then exist in the process box database, but not in the NIM database.
- A point is deleted while the LOADSCOP of the NIM is NIMOnly that deletes the point from the NIM database only. The point data remains in the process box database.
- A database for an process box is created at a separate location and taken to a different site. If a checkpoint file is used to load the process box and not the NIM, the process box database will contain the point data, but the NIM will not.

Identifying Point Fragments

Identify NIM point fragments

You may become aware of a NIM point fragment when you attempt to load a new point into a slot that is in fact empty at the process box level, but has a tagname in the NIM. In this case, you have found the fragment and can proceed with recovering or deleting the fragment.

You may also find a NIM fragment by noticing that a particular point doesn't "look right" to you. For example, a Regulatory Control point with a DAS algorithm, or a Digital Composite point with missing input/output connections (---- appears).

You may even design a method to routinely compare your current database with known good point definitions. For example you might generate a printout of an EB or IDF that you have saved and compare it with the point's current configuration. Or, you might write a program (to run on a PC) to compare two EB files, one previously saved and one containing a current definition, of the same point in order to identify any changes.

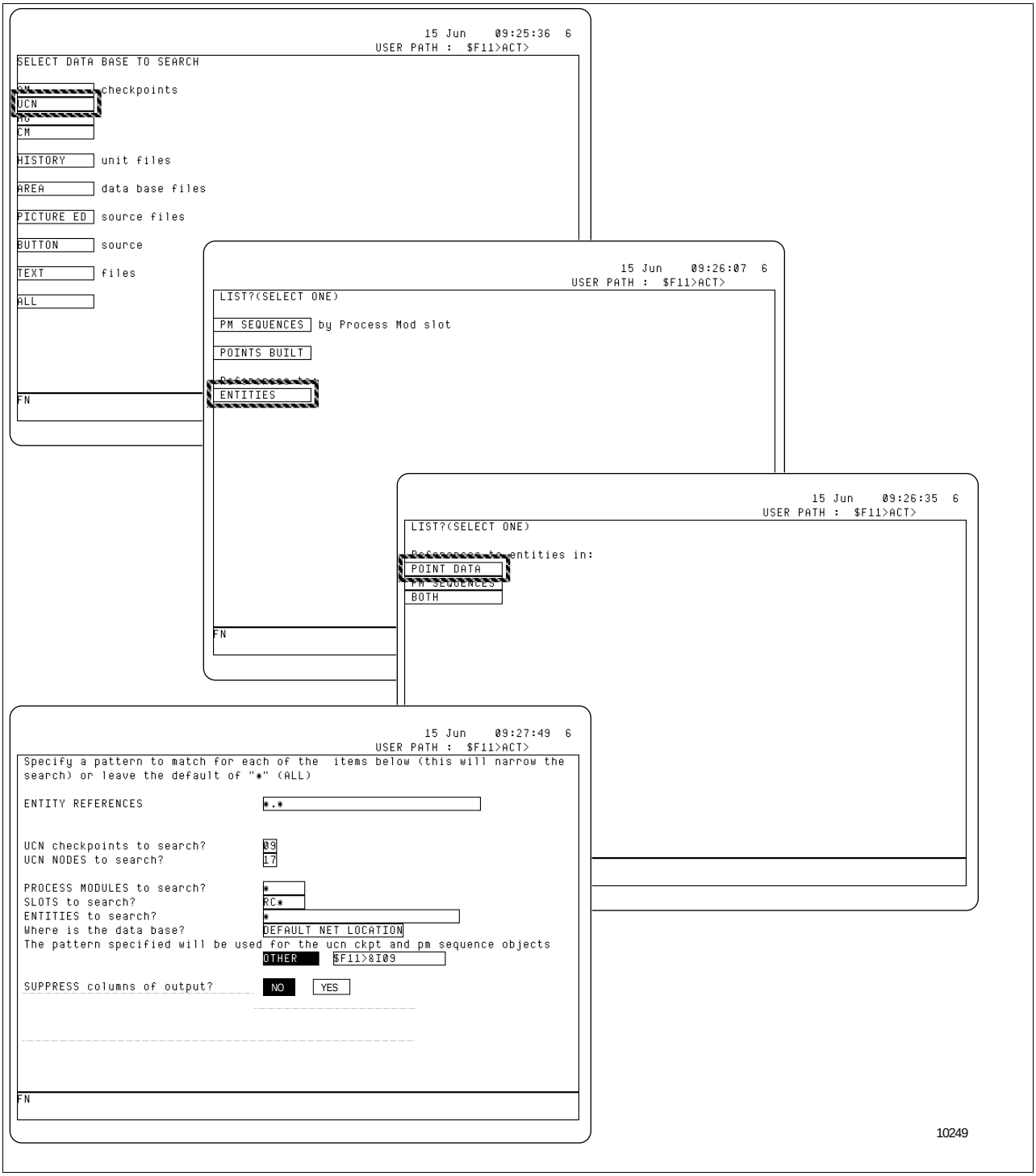
Use Find Names to identify process box point fragments

One way to identify process box point fragments is to use the Find Names utility. By selecting to search the checkpoint file for "References to ENTITIES" in both POINT DATA and PM SEQUENCES using the wildcard (*), you will get a list of the points in the checkpoint file, the PM/APM sequences, and the points they reference, even those without tagnames. You can then identify the point fragments by searching the Find Names output for "????????" under the column ENT_REF where there should be a tagname.

Figure 12 shows a Find Names search for all entity references in the checkpoint files "POINT DATA" for APM 17 on UCN 9

Continued on next page

Figure 12 Find Names and Point Fragments



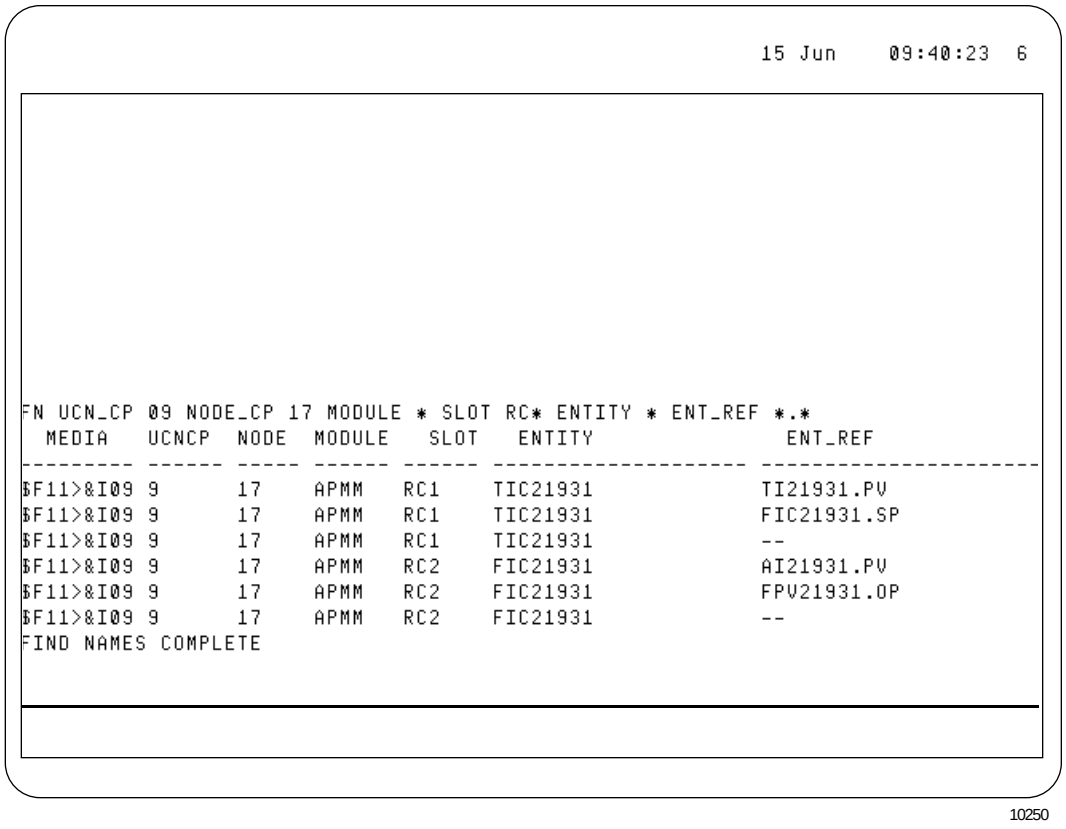
Continued on next page

Identifying Point Fragments, Continued

Typical Find Names results

Figure 13 shows a typical Find Names output when no point fragments are present.

Figure 13 Find Names Output - Normal

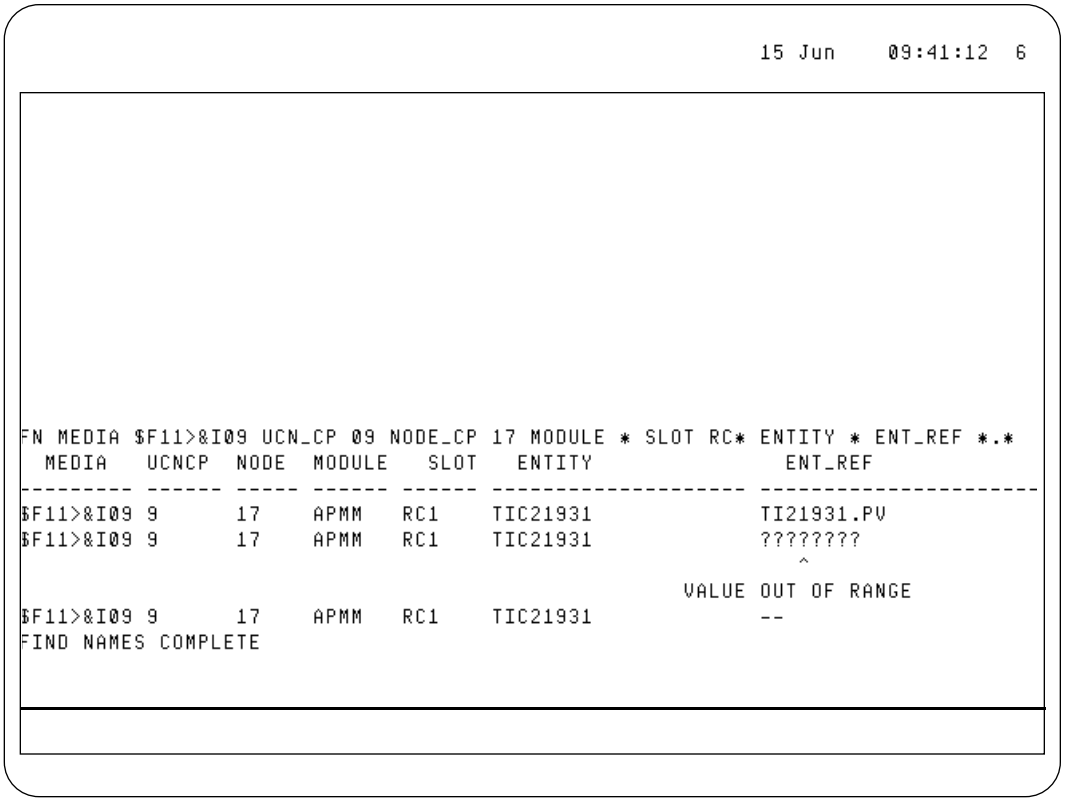


Continued on next page

Find Names results with fragments

Figure 14 shows the output after FIC21931 was deleted from the NIM only. Note that only point fragments that are referenced from other points will be listed as "????????".

Figure 14 Find Names Output - With Fragments



10251

ATTENTION

ATTENTION—As usual when using the Find Names utility, it is best to search the checkpoint files on removeable media rather than those on the HM. This avoids the possibility of interfering with automatic checkpointing operations and the remote possibility of corrupting the current checkpoint files.

Continued on next page

Using slot summary to identify process box point fragments

A second method uses the Slot Summary function to locate process box point fragments. Table 18 outlines the procedure.

Table 18 Slot Summary Procedure to Locate Fragments

Step	Action	Result
1	From the System Status display select the appropriate UCN	The UCN Status display appears
2	Select a UCN node	The node Status display appears
3	Select the APMM, then select the Slot Summary target	A list of point types appears
4	Select a point type	A list of points by slot appears
5	Select an empty slot (no tagname listed)	The slot is highlighted
6	Press the [DETAIL] key	Detail information for the slot will be displayed. If there is configuration data present, you have found a point fragment.
7	Repeat Steps 4-6 for each empty slot for each point type	

This procedure must be performed for each UCN node in order to identify all process box point fragments.

Correcting Point Fragments

Correcting NIM fragments

NIM point fragments occur when point data, including the tagname, exists in the NIM database, but not in its appropriate process box database. Once you have identified NIM fragments, you should correct this situation. You may desire to restore the missing portion of the point's definition in the process box database, or you may want to delete the point from the NIM database. The following table explains how to accomplish these tasks.

Table 19 Procedure to Restore/Delete NIM Fragments

Action	Method
Restore	• RECONSTITUTE the NIM network data point ($NMuuNbb), and verify that LOADSCOP is NIMAndPM, if not – Change the LOADSCOP to NIMAndPM – Reload the NIM network data point • From the point's Detail display, set the point to INACTIVE • From the Data Entity Builder, RECONSTITUTE the point, enter the desired box resident information, and load the point with OVERWRITE **OR** Load the point using an IDF or EB file • Activate the point upon successful restore **RESULT:** The point data has been established in the process box database.
Delete	• If the point is ACTIVE, set it to INACTIVE. • If there is a possibility that references to this point exist in the system, e.g., area database, history groups, etc., use Find Names to locate and resolve the references. • From the Data Entity Builder, select DELETE SYSTEM ENTITY and specify the tagname of the point you want to delete. **RESULT:** The NIM point fragment is deleted from the NIM database.

Checkpoint after correction

Be sure to checkpoint the database after resolving the NIM fragment.

Continued on next page

Correcting process box fragments

When process box point fragments exist, the NIM has no tagname or data associated with that point that still exists in the process box database. You will want to either restore the point in the NIM database, or delete the point data in the process box. The steps to accomplish this are listed in Table 20.

Table 20 Procedure to Restore/Delete Box Fragments

Action	Method
Restore	• RECONSTITUTE the NIM network data point ($NMuuNbb) – Change the LOADSCOP to NIMOnly – Reload the NIM network data point • Enter and load the NIM-resident point data (data will be loaded to the NIM only because LOADSCOP is set to NIMOnly) • RECONSTITUTE the NIM network data point ($NMuuNbb) – Change the LOADSCOP to NIMAndPM – Reload the NIM network data point **RESULT:** The point data in the process box remains as it was, and the NIM now has data, including a tagname, for the point in it's database.
Delete	• RECONSTITUTE the NIM network data point ($NMuuNbb) – Change the LOADSCOP to NIMOnly – Reload the NIM network data point • Enter and load the NIM-resident point data (data will be loaded to the NIM only because LOADSCOP is set to NIMOnly) • RECONSTITUTE the NIM network data point ($NMuuNbb) – Change the LOADSCOP to NIMAndPM – Reload the NIM network data point • If the point is ACTIVE, set it to INACTIVE • From the Data Entity Builder, select to DELETE SYSTEM ENTITY and specify the tagname of the point **RESULT:** After first establishing a tagname for the process box point fragment in the NIM database, the point can then be deleted from the NIM and process box databases.

Avoiding Point Fragments

Guidelines

These are some guidelines to help you avoid inadvertently creating point fragments:

- Make certain that when adding or deleting points from the Data Entity Builder, the UCN Node Configuration parameter LOADSCOP is set to NIMAndPM in the NIM. If it is ever necessary to change this parameter to NIMOnly, remember to return it to NIMAndPM as soon as possible.
- Update the UCN checkpoint files after adding or deleting points by using the SAVE DATA target on the UCN or HPM/APM/PM/LM/SM status displays. Perform the SAVE twice to ensure that both versions of the checkpoint files contain the new data. Also, update any cartridge backups.

Summary

Following the recommendations listed in this material will help you avoid point fragments; however, a solid understanding of the following concepts is even more important in helping you to develop safe practices in managing your database:

- Understanding the distributed database concept and the relationship between the point data stored in the NIM and the point data in the other UCN node databases.
- Understanding the point loading and point deletion processes. Knowing the significance of the NIM LOADSCOP parameter.
- Understanding the concepts of checkpointing and which files are updated when a checkpoint is requested.
- Understanding the affect that the disabling of automatic checkpointing at the different levels will have on the checkpoint files updated.
- Knowing what occurs when a node is restored from a checkpoint file.

A clear understanding of these concepts helps avoid the creation of point fragments in your UCN databases.

Untagged References

Definitions and Examples

Definition of untagged references

An untagged reference is a way to access PM/APM resources without using a tagname (entity name) in the NIM. There are two types of untagged references:

- Hardware references
- Box data point references

Definition of a hardware reference

When building points in the PM and APM, it is possible to reference IOP parameters (within the same node) by using a hardware address rather than a tagname. This type of referencing can be used with AO, DI, and DO IOPs An IOP resource can be accessed with either a tagname or a hardware reference, or with both tagname and hardware references.

The format for a hardware reference is

!MTmmSss.Parameter

Table 21 lists the components of the hardware address format and the possible values for each.

Table 21 Hardware Address Format

Characters	Meaning	Possible Values
!	Designates that a local hardware or box address follows	!
MT	IOP type	AO, DI, or DO
mm	Number of the IOP card in the card file	1 - 40
S	Slot	S
ss	Input or output slot number on the specified IOP card	Depends on IOP type: • AO= 1 - 8 • DI= 1 - 32 • DO= 1 - 16
Parameter	IOP resident parameter	Depends on IOP type

Continued on next page

Example of a hardware reference

As an example, **!DI07S04.PVFL** is an untagged hardware reference to

- a digital input (DI) IOP,
- assigned as module 7,
- slot 4 of module 7,
- PV flag (PVFL), which is current input state.

Definition of a box point reference

Global variables in the PM and APM can be referenced without a tagname. These variables can be referenced internally within the same box by using the format !BOX.variable(i), or by another node on the UCN using the network data point or the box data point for the node.

The format for the network data point and the box data point is:

$NMuuNbb.variable(i)
$NMuuBbb.variable(i)

where uu=UCN number,
bb=UCN node number,
variable= FL, NN, TM, TIME (APM), STR (APM), and
i =index number.

Table 22 summarizes PM references, and Table 22 summarizes APM references.

Table 22 Box Point References for the

Type	Reference Format	Index Value	Point Form
Box Flag	!BOX.FL(i) $NMuuNbb.FL(i)[1] $NMuuBbb.FL(i)[2]	1-1023	Full or component
Box Numeric	!BOX.NN(i) $NMuuNbb.NN(i)[1] $NMuuBbb.NN(i)[2]	1-2047 (configurable)[3]	Full
Box Timer	!BOX.TM(i) $NMuuNbb.TM(i)[1] $NMuuBbb.TM(i)[2]	1-64	Component

1 Format used in CL/PM

2 Format used in continuous points

3 Configurable amount during node specific configuration

Continued on next page

Table 23 Box Point References for the APM

Type	Reference Format	Index Value	Point Form	Notes
Box Flag	!BOX.FL(i) $NMuuNbb.FL(i)[1] $NMuuBbb.FL(i)[2]	1-16,384	1-2047 full or component	• 1-2047 can use tagnames • 1-4095 are LCN-accessible • 4096-16,384 are accessed through an array point only
Box Numeric	!BOX.NN(i) $NMuuNbb.NN(i)[1] $NMuuBbb.NN(i)[2]	1-16,384 (configurable)[3]	Full	• 1-2047 can use tagnames • 1-4095 are LCN-accessible • 4096-16,384 are accessed through an array point only
Box Timer	!BOX.TM(i) $NMuuNbb.TM(i)[1] $NMuuBbb.TM(i)[2]	1-64	Component	Can assign tagnames
Box Time[4]	!BOX.TIME(i) $NMuuNbb.TIME(i)[1] $NMuuBbb.TIME(i)[2]	1-4096 (configurable)[3]	Component	• Can't assign tagnames • 1-4095 are LCN-accessible • 4096 must be accessed through an array point
Box String[4]	!BOX.STRn(i) $NMuuNbb.STRn(i)[1] $NMuuBbb.STR(i)[2]	1-16384 (configurable)[3]	Component	• Can't assign tagnames • 1-4095 are LCN-accessible • 4096-16,384 are accessed through an array point only

1 Format used in CL/PM

2 Format used in continuous points

3 Configurable amount during node specific configuration

4 Cannot pull up from the Slot Summary display

Example of a box point reference

For example, **$NM01B05.FL(100)** is an untagged reference to

- UCN process network 1,
- node 5 on UCN 1,
- flag number 100.

Disadvantages of Untagged Hardware References

Recommendation

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).
- The user can inadvertently use an untagged reference more than once.
- 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.

Continued on next page

Disadvantages of Untagged Hardware References, Continued

ATTENTION

ATTENTION—If untagged referencing is used, it is important to clearly document which addresses are currently in use and their connections to other points. Accurate documentation benefits many people who work on your system. For example,

- Instrument technicians who may have to replace hardware need to be aware of the existence and location of untagged references.
- Engineers who are doing new configuration on the system need to know the (untagged) resources being used to avoid using the same resource.

This can be done manually by the user with some diligence and effort; however, the Find Names utility can also be used to document untagged references as saved in a checkpoint file. There is a lab in the “Find Names for UCN” module that provides practice in documenting untagged hardware references.

Lab Exercises

Lab 1—Use Exception Build

Introduction

One common way to create backup files is to do a Print System Entities to EB files. This method works well in most cases; however, on heavily loaded systems, you can avoid timeouts by reconstituting points (backbuilding) to Intermediate Data Files (IDFs), then building your EB files from IDF files. The lab uses that procedure.

Lab requirements

This lab requires an HPM, APM, or PM with a control partition defined. To minimize lab time, it is necessary only to backup the logic slot point types. At your plant, back up all point types.

Lab prerequisites

This lab assumes familiarity with Documentation Tool, and Data Entity Builder commands. Note: In the lab procedure, replace the characters ### with your assigned student partition number.

Backup procedure

Step	Action
1	Create a selection list for the logic points in your assigned UCN node. Use Documentation Tool (Doc Tool) or Data Entity Builder (DEB) commands to generate the lists. Both methods are documented, select one to use. • To use Doc Tool: – From the Engineering Main Menu, select Documentation Tool – From the top line of the Doc Tool display, select QUERY – Select “Build” – Select “UCN” – Enter data for the following: **Proc Net List?** (enter your UCN #) **Device List?** (enter your PM, APM, or HPM #) – For “Resource/Entity Types:” select LOGIC – Press [ENTER] – After data appears on the screen, select OUTPUT from the top line of the Doc Tool display – Select “To a File” – For “Pathname” enter: NET>S###>LOGIC.EL – Select “Overwrite Without Field Definitions” – Press [ENTER] – After “OPERATION COMPLETE” at the bottom of the screen, press the [CTL] and [HELP] keys to return to the Main Menu

Continued on next page

Backup procedure, continued

	• To use DEB commands: – From the Engineering Main Menu, select BUILDER COMMANDS – From the DEB COMMAND DISPLAY, select LIST ENTITIES IN MODULE – Enter the following data: **pathname for SELECTION LIST** NET>S###>LOGIC.EL **MODULE or BOX number** (your assigned PM/APM/HPM) **HIWAY/UCN number** (your assigned UCN) – For "BUILD TYPES of Entities to List:" select NIM – Select the LOGIC option – Press [ENTER] – The selection list created will be displayed on the screen. Press the [CTL] and [HELP] keys to return to the COMMAND DISPLAY Either method creates a selection list, named LOGIC.EL, of the logic slot points in your assigned UCN node. The file is created in your student directory.
2	From the DEB COMMAND DISPLAY, select RECONSTITUTE MULTIPLE. Enter the following data : **pathname for IDF** NET>S###>LOGIC.DB **pathname for SELECTION LIST** NET>S###>LOGIC.EL Press [ENTER]
3	If there are no errors, continue with the next step. If an error occurs, locate the -UL file. Rename the -UL file and perform Step 2 again, using it as the selection list to try to get the points it failed to get. If successful, continue to the next step. If not, consult your course manager.
4	After successfully reconstituting all your logic slot points into an IDF, select PRINT ENTITIES. Then select PRINT IDF Entities from the choices that appear. Enter the following data: **pathname for IDF** NET>S###>LOGIC.DB **pathname for SELECTION LIST** NET>S###>LOGIC.EL **PRINT device ID or DESTINATION pathname** NET>S###>LOGIC.EB Press [ENTER]
5	From the Command Processor, make a backup copy of your EB file.

Lab 2—Change Control Partition

Introduction

The following procedure shows one approach to changing your control partition. Note that you may decide to follow a different approach, however, the following procedure illustrates various considerations.

Lab requirements

This lab requires a PM or APM with a control partition defined. Use the logic slot Exception Build file created in the earlier exercise. To minimize lab time, Exception Build files are already created for you with the other point types such as control points, digital composites that you will load to the PM or APM assigned to you.

Lab prerequisites

This lab assumes familiarity with use of command files and IDFs.

PM/APM Procedure

Step	Action
1	According to the Partition Change procedure (Table 1) of this module, your first step is to get all the control partition points for your assigned device into IDFs. One way to do this is by point type. You already have in your student directory the IDF LOGIC.DB that you created in Lab 1. Because you have performed the steps to create a selection list and IDFs in Lab 1, the IDFs for the remaining point types have been created for you to save time. Copy the following files from NET>3716 into your student directory, replacing ## with your UCN node number: For PM and APM **REGPV##.DB** **REGCT##.DB** **DIGCM##.DB** **PRMOD##.DB** **FLAGS##.DB** **TIMER##.DB** **NUMRC##.DB** For APM only **ARRAY##.DB** **DEVCT##.DB** **STRING##.DB** **TIMES##.DB** (Note: You should know how to create a selection list by point type, and corresponding IDFs from Lab 1; however, if you are not sure of the steps to do this, discuss it with your course manager now.)
2	Reconstitute the $NMuuBpp node-specific configuration for your assigned UCN node.

Continued on next page

Lab 2—Change Control Partition, Continued

PM/APM Procedure, continued

3	Change the point mix by adding an additional digital composite point. This means the control partition is changed, and will initialize the PMM or APMM database when loaded.
4	Load the new control partition configuration
5	Your next step is to load all the PMM or APMM entities you saved in IDFs in Step 1. You could do this by selecting LOAD MULTIPLE from the DEB Command display and specifying each IDF in sequence. Another way is to create an Executable Command file to do it for you. A sample EC file has been created for you in NET>3716. The name of the file is LOADPM##.EC, where ## is the UCN node number of your assigned PM/APM. Copy this file into your student directory now. Edit the EC file to use the IDFs in your student directory. (You may create your own instead if you prefer)
6	Load all control partition entities for your UCN node by using the .EC file you edited (or created) in the previous step. (Note that this EC file is written by using DEB commands, so it must be executed from the DEB Command display)
7	Call up a Detail display for one of the points in your assigned device's control partition and review its configuration.
8	When the load is complete, normally you would checkpoint the new changes to the HM. It is not necessary to do this in the lab to maintain our database integrity.

HPM procedure

OPTIONAL—Change an HPM control partition (refer to Table 1, if necessary):

1. Decrease the number of RegCtl slots by 2.
2. Increase the number of Logic slots by 2.

Lab 3—Clear IOP Database

ATTENTION

ATTENTION—The following lab exercise is optional.

Lab requirements

This lab requires an HPM, APM, or PM with an IOP point defined. This way you can observe the effects of clearing an IOP database.

Clear IOP database procedure

Step	Action
1	Call up the Status display for your assigned device.
	Call up the slot summary for your assigned IOP. Select one of the points listed, and press the [DETAIL] key.
2	Verify that its values are valid. Leave the detail display on your screen while you perform the following steps.
3	Locate the IOP card that the point is built against.
4	Remove the IOP card, then reinsert it.
5	Observe the detail display for the IOP point.
6	Observe whether its values have changed. If they have not, clear the IOP database in the next step.
7	Again locate your assigned IOP card. Toggle the upper extractor twice to clear the IOP database.
8	Again observe the detail display for the IOP point
9	Observe that the values have changed to default entries and the point is now inactive.
10	Return to the Status display by pressing the [PRIOR DISP] key.
11	Notice that the IOP is now in the IDLE state.
12	Select your assigned IOP. Next select the RUN STATES target at the bottom of the display. Notice that a message of DBINVALD appears in your IOP's status box.
13	Restore your assigned IOP using the Default Source (NET).
14	Once restored, start up the IOP. It should now display a status of OK.

Lab 4 - List Checkpoint Files

Introduction

The purpose of this lab is to become familiar with the checkpoint files for the UCN and the effect of disabling automatic checkpointing at different levels.

Lab requirements

For this lab you need a cartridge and access to a UCN node.

Lab procedure

Step	Action
1	Insert your cartridge into an accessible drive.
2	From the Status display of your assigned UCN node, request a SAVE to the cartridge.
3	Go to the Command Processor and list the files that were created on your cartridge. Notice that the checkpoint operation created the necessary directory on the cartridge if it did not already exist there. Directory created: _____ Files created: _____ _____ _____ _____
4	Request a second SAVE of your assigned UCN node to cartridge.
5	This time list the new files created on the cartridge by the checkpoint operation. New files created: _____ _____

Lab 5—Clear IOP and Restore from Checkpoint

Introduction

The purpose of this lab is to demonstrate how an older revision of a checkpoint file is used during a checkpoint SAVE operation to read and save the valid IOP data for an IOP with an INVALID database at the time of the checkpoint.

Lab requirements

For this lab you need a cartridge and access to a UCN node. Ask your course manager to assign to you an IOP that you can clear for this lab.

Lab procedure 1

In this procedure an IOP's database will be cleared, which will cause it to be marked as INVALID. A checkpoint will be performed and then used to try to restore the IOP.

Step	Action
1	If there are any checkpoint files on your cartridge, delete them at this time.
2	From the Status display of your assigned node, verify that the IOP currently has a database marked VALID.
3	Perform a checkpoint SAVE operation to your cartridge.
4	Locate the IOP that your course manager assigned to you. Toggle the upper extractor twice to clear the IOP database.
5	From the Status display, notice that the database for the IOP is now considered INVALID.
6	Perform a second checkpoint SAVE operation to your cartridge. Because the database for the cleared IOP is now INVALID (contains the default database), it's database will not be saved in the checkpoint file. Because there is an older checkpoint file on the cartridge with a VALID database for the IOP, this database will be saved in the new checkpoint file instead.
7	From the Command Processor, list the checkpoint files that were created on the cartridge from the two SAVE operations performed.
8	Perform a RESTORE operation on the IOP by using the checkpoint files on the cartridge.
9	From the Status display, notice that the database for the IOP is marked as VALID.

Conclusion

An INVALID IOP database will not be saved in a checkpoint file. If there is an older revision checkpoint file available with a VALID database for that IOP, that database will be written into the new checkpoint file. If the new checkpoint file is used to restore the IOP, it will come up with a database marked VALID.

Continued on next page

Lab 5—Clear IOP and Restore from Checkpoint, Continued

Lab procedure 2

In this procedure an IOP's database will be cleared, which will cause it to be marked as INVALID. A checkpoint will be performed and then used to try to restore the IOP.

Step	Action
1	If there are any checkpoint files on your cartridge, delete them at this time.
2	Locate the IOP that your course manager assigned to you. Toggle the upper extractor twice to clear the IOP database.
3	From the Status display of your assigned node, notice that the database for the IOP is now considered INVALID.
4	Perform a checkpoint SAVE operation to your cartridge. Because the database for the cleared IOP is now INVALID (contains the default database), it's database will not be saved in the checkpoint file.
5	From the Command Processor, list the checkpoint files that were created on the cartridge.
6	Perform a RESTORE operation on the IOP by using the checkpoint files on the cartridge.
7	From the Status display, notice that the database for the IOP is still considered INVALID.
8	RESTORE the IOP by using the checkpoint files on NET. The IOP database should be restored with a VALID status.

Conclusion

An INVALID IOP database will not be saved in a checkpoint file. If there is not an older revision checkpoint file available with a VALID database for that IOP that can be used in the new checkpoint file, the new file will not contain a VALID database for the IOP and the IOP will come up with an INVALID database status if restored with the new checkpoint.

Lab 6—Reload NIM and Check Auto Checkpoint States

Introduction

This lab will demonstrate the affect of a NIM load on the Automatic Checkpoint states for the NIM, the UCN, and the devices on the UCN.

Lab requirements

For this lab you will need access to a NIM and a UCN. Your course manager must coordinate this activity for you. Please notify your course manager that you are ready to begin.

Lab procedure

Step	Action
1	From the Gateway Status display, determine the AUTO SAVE state for your assigned NIM. If it is currently DISABLE, change it to ENABLE now.
2	From the UCN Status display, determine the AUTO CHECKPNT state of your assigned UCN. If it is currently INHIBIT, change it to ENABLE now.
3	From the UCN or HPM/APM/PM Status display, determine the AUTO CHECKPNT state of a node on the UCN. If it is currently ENABLE, change it to INHIBIT now.
4	At this point, the three levels of auto checkpointing should be in the following states: NIM = ENABLE UCN = ENABLE UCN NODE = INHIBIT
5	From the UCN Status display, initiate a checkpoint SAVE of your assigned UCN device to NET. This is done because the NIM and device checkpoint states are saved in the checkpoint file.
6	From the Gateway Status display, shutdown and reload your assigned NIM from NET.
7	After the NIM completes loading, note the auto checkpoint states for each of the three levels: NIM = _____ UCN = _____ UCN NODE = _____
8	The UCN auto checkpoint state should default to INHIBIT, while the NIM and UCN Node states default to the states as saved in the checkpoint file used to load (step 4).
9	From the UCN Status display, change the AUTO CHECKPNT state of your assigned UCN device to ENABLE.

Lab 7—Create and Resolve Point Fragment

Introduction

This lab demonstrates one way that a process box point fragment can be created. The point fragment is then restored to the NIM in the second part of the lab.

Lab requirements

For this lab you need your partition sheet and access to your assigned UCN device. Again, because of the nature of the exercise, your course manager must coordinate your activity with other users. Inform your course manager now that you are ready to begin. (Note: In the lab procedure, replace the notation ### with your partition number)

Lab procedure 1

In this part of the lab, a process box point fragment will be created.

Step	Action
1	Using the information on your partition sheet, call up your first operating group.
2	Notice that the first two points in your group are cascaded, with TIC21### being the primary and FIC21### the secondary.
3	Call up a Detail display for each point and verify the input and output connections.
4	Stop at this point and notify your course manager that you are on Step 4. Wait (and watch if you like) while your course manager changes the LOADSCOP parameter for your NIM to NIMOnly.
5	From the DEB, delete the point FIC21###. Because LOADSCOP = NIMOnly, the point is deleted from the NIM database, but remains intact in the HPM, APM, or PM.
6	Attempt to call up a Detail display of the point FIC21###. The point will not be found because it doesn't exist in the NIM.
7	Call up the UCN Status display for your UCN.
8	Select your assigned device and request a Slot Summary for Regulatory Control points.
9	Refer to your partition sheet for the slot number of FIC21###. Notice that there is no tagname for this slot. Select the slot and press the [DETAIL] key.
10	Verify that the configuration data remains in the node.

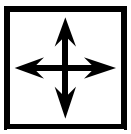
Continued on next page

Lab procedure 2

The following procedure will re-establish the tagname for FIC21### in the NIM.

Step	Action
1	Verify that the LOADSCOP of the NIM is still NIMOnly by reconstituting the NIM network data point ($NMuuNbb).
2	Proceed to load the point FIC21### by using the information on your partition sheet. You need only to enter the NIM part of the point definition because the LOADSCOP is NIMOnly and the process box portion of the point definition is intact.
3	Once loaded, call up a Detail display of FIC21### and verify that the configuration looks complete.
4	Inform your course manager that you are done.
5	When all others have also completed the lab, your course manager will reconstitute the NIM network data point and return LOADSCOP to NIMAndPM.

Directions



DIRECTIONS—This is the end of the study material for this module. Discuss questions concerning the study material or the lab activities with a colleague or a course manager

If you are satisfied that you have achieved the objectives of this module, continue with the next section, the Student Proficiency Evaluation.

Student Proficiency Evaluation

Criterion Test

Test item 1

Suppose that you are changing the control partition in a HPM, APM, or PM by using the procedure documented in this course module. So far, you have

- Backed up all control partition points into IDFs (of those, 30 are regulatory control points)
- Reconstituted the network box configuration point, $NMuuBbb
- Made desired changes to the point mix
- Loaded the new point/slot configuration

At this point call up a slot summary for the box and select Regulatory Control points from the list.

Will you see any tagnames displayed? _____

Why or why not? _____

_____

_____

_____

_____

Test item 2

List a general description of the files created when you demand a checkpoint SAVE for a specific HPM, APM, or PM.

1. _____

2. _____

3. _____

4. _____

Continued on next page

Criterion Test, Continued

Test item 3

Complete the following table of information about the NIM parameter LOADSCOP.

Parameter Value	Meaning	When to Use
NIMOnly		
NIMAndPM		

Test item 4

Give a definition of the two types of point fragments that can exist.

1. _____

_____

_____

2. _____

_____

_____

Self-Evaluation

Test item 1

Will you see any tagnames displayed? *Yes*

Why or why not? *The NIM database (including tagnames and descriptors) remains intact when a process box control partition change is performed. Only the PMM or APMM database is initialized. The HPM database remains intact.*

Test item 2

A demand checkpoint of a specific HPM, APM, or PM will create the following files:

1. *box data file*
2. *box master file*
3. *NIM data file*
4. *NIM master file*

Test item 3

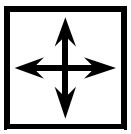
Parameter Value	Meaning	When to Use
NIMOnly	*Only the NIM database is updated when an entity is loaded or deleted.*	• *To establish points in the NIM before process box installation in order to continue with other configuration tasks such as picture building, area database and history group configuration.* • *To load a tagname in the NIM for a process box point fragment in order to restore or delete the fragment*
NIMAndPM	*Both the NIM and process box database are updated when an entity is loaded or deleted.*	*Under normal conditions*

Test item 4

A NIM point fragment *exists when the NIM database contains data for a point, but the process box database does not.*

A process box point fragment *exists when the process box database contains data for a point, but the NIM database does not. In this case there is no tagname associated with the point because the NIM is not aware that the point exists.*

Directions



DIRECTIONS—This is the end of this module.

Use your course map to

- Get your course manager to sign off this module.
- Choose your next eligible module.

If you have a question

- Ask your course manager.

LAST PAGE