Honeywell

Advanced AM Implementation Techniques

4410S

TotalPlant

Notices and Trademarks

Honeywell Inc.

Industrial Automation and Control

Automation College

2820 West Kelton Lane

Phoenix, AZ 85053-3028

1-800 852-3211

Honeywell

Lesson 1

4410S

TotalPlant

Lesson 1 - Table of Contents

Acronyms

CL ..... Control Language
AM ..... Application Module
CDS ..... Custom Data Segment

Introduction

LESSON OVERVIEW

Lesson Objectives

Upon completion of this lesson, you will understand CL/AM source file structures and related object files.

You will also understand:

- Implementation of default pathnames
- Implementation of Custom Data Segments, including attributes, defaults, and arrays
- Implementation of Enumerations (Types)
- Use External References
- Use Local Declarations (Type and Arrays)

Lesson Outline

AM HARDWARE AND POINT TYPES OVERVIEW

INTRODUCTION

LESSON OVERVIEW

AM HARDWARE REQUIREMENTS

CONCEPTUAL OVERVIEW OF AM MEMORY

AM POINTS - TYPES

AM POINTS - REGULATORY

AM POINTS - CUSTOM

AM POINTS - SWITCH

AM LOAD/SAVE DATA OPERATIONS

REVIEW of CL/AM STRUCTURES

RELATED FILE TYPES
IMPLEMENTATION
CL SOURCE FILES

CL "PACKAGE" SOURCE FILES\L 2

CUSTOM DATA SEGMENTS

DEFINITION
ATTRIBUTES

DEFAULTS
ARRAYS
IMPLEMENTATION

ENUMERATIONS

TYPES
IMPLEMENTATION

EXTERNAL REFERENCE

POINT IDENTIFIERS

LOCAL DECLARATIONS

TYPES
ARRAYS

Course Introduction

The basic AM course introduces the student to:

- Advanced control schemes that can be constructed using regulatory points with advanced control and OV algorithms.
- The AM Control Language (CL/AM), and its use in programming some custom calculations and algorithms.
- This advanced AM course expands on that initial exposure to AM features by focusing on the capability the AM has to function as a general data processing resource, whose domain is in the middle area between a host computer and process-connected devices.
- This course will emphasize AM custom data structures, such as CDS parameters, parameter lists, custom points, and more advanced CL/AM programs.
- You will learn how the AM can be used for medium-scale data processing, recipe management, and other functions.
- You will also learn how to predict, monitor and modify AM performance, as well as learn techniques to maximize the use of the AM memory and processing resources.

AM Hardware Requirements

Memory Sizing

	R300	R400	R500	R600
Minimum Memory				
Maximum Memory				
Am Redundancy				

Dual Node Board Configurations

Slot	Front	Rear
3		
2	AMR	AMR/IO
1	K2LCN/K4LCN	KLCN

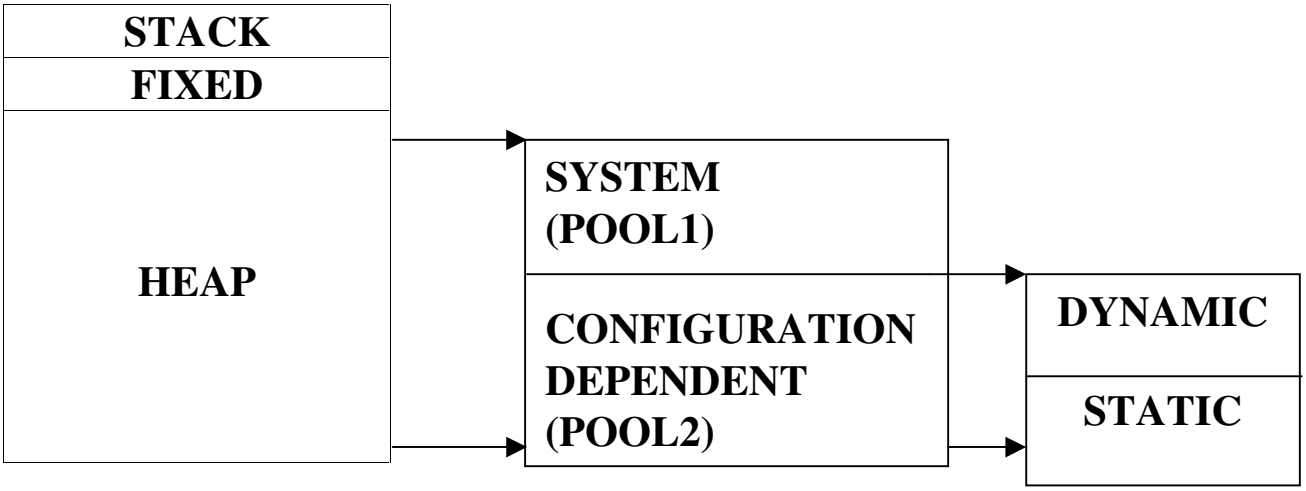
Five Slot Board Configurations

Slot	Front	Rear
5		
4	QMEM/EMEM	
3	AMR	AMR/IO
2	LCN	LCN/IO
1	HMPU/HPK2	Optional Clock/IO

Five Slot Node Configurations with K2LCN/K4LCN

Slot	Front	Rear
5		
4		
3		
2	AMR	AMR/IO
1	K2LCN/K4LCN	LCN/IO

Conceptual Overview of AM Memory



AM Unit Structure

AM User Memory

Current - Value Buffers (CVB's)	
Unit A	Unit Overhead
	Point Data
	CL Blocks and Reference Lists
	CDS Descriptors
Unit B	Unit Overhead
	Point Data
	CL Blocks and References Lists
	CDS Descriptors
	Checkpoint Snapshot Buffer
	Free Memory

AM Points

Types

- Regulatory - Controller
- Flag - On/Off State
- Numeric - Real Number
- Counter - Used for PIU/DHP Interface
- Timer - Countdown Integer Timer
- Custom - User Configurable Parameters and Functions
- Switch - Up to 5 States, User-Configurable Functions

AM Points

Regulatory

Very similar to NIM/HG Regulatory Points, with advanced features and options:

- Configurable execution rate, before/after/cycle options or Point Process-Special (PPS).
- 11 PV algorithms, 13 control algorithms.
- Up to 8 configurable inputs to algorithms.
- Up to 8 configurable outputs from control algorithms.
- Up to 8 general inputs and outputs.
- _____ CL insertion points.
- Up to _____ CL blocks linked per insertion point.
- Internal switch option.
- Enhanced alarming features.
- Enhanced initialization features.
- May attach up to _____ CDS packages on a point, containing up to _____ parameters per CDS class.
- Three CDS parameter classes on this point type:
  _____, _____ and _____.
- Useful for advanced control schemes and simulations.

AM Points

Custom

Totally user-configurable point type, with these standard parameters:

- Configurable execution rate, before/after/cycle options or Point Process-Special (PPS).
- Two CL insertion points; up to _____ CL blocks may be linked at each point.
- May attach up to _____ CDS packages, containing up to _____ parameters.
- One CDS class, _____.
- Useful for advanced control schemes, file management, simulations, and many other custom applications.
- Two CDS Parameter Classes on this Point Type:
  _____ and _____.

AM Points

Switch

Five-position switch with configurable descriptors:

- Configurable execution rate, before/after/cycle options or Point Process-Special (PPS).
- Configurable alarming parameters.
- Two CL insertion points; up to _____ CL blocks linked at each one.
- May attach up to _____ CDS packages, containing up to _____ parameters.
- One CDS class, _____.
- Switch functions are programmed through CL blocks.
- Useful for switching advanced control schemes, simulations, and many other custom applications.

AM Load/Save Data Operations

- First Load must use startup checkpoint for data (the *.MAS.CP file for each assigned unit is needed).
- First Save may be made to blank disks or HM, directories and files are automatically created; first Save takes longer because the checkpoint file structure has to be created.
- _____ directories have nothing in them, but are needed to locate the checkpoint data.
- Load modes: hot/warm/cold/no process.
- Data base items are logically grouped by _____ in AM memory; designed that way so that they can be easily moved from one AM to another.
- Data is Saved (checkpointed) by the unit _____ that is linked internally to the unit id.

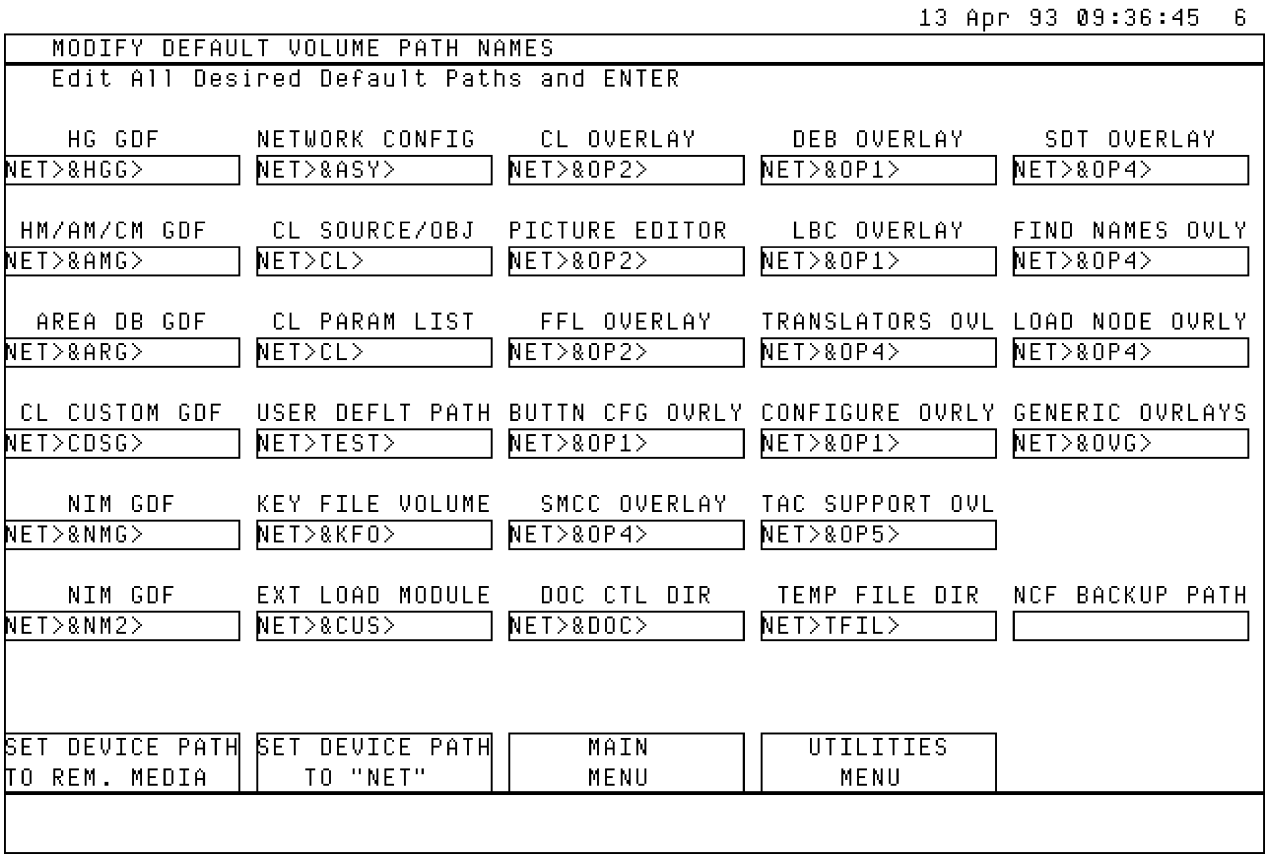
Review of CL/AM Structures

RELATED FILE TYPES

File Type	Residence	Description
		CL Source (created in Text Editor)
		CL/AM Object Block; linked to AM points
		CL Listing (no errors); for compile results
		CL Error Listing; for troubleshooting
		CDS Object; attached to AM points
		User Parameter List Object
		Enumeration Set Name/State Descr. Objects
		CDS/Parameter Name Objects

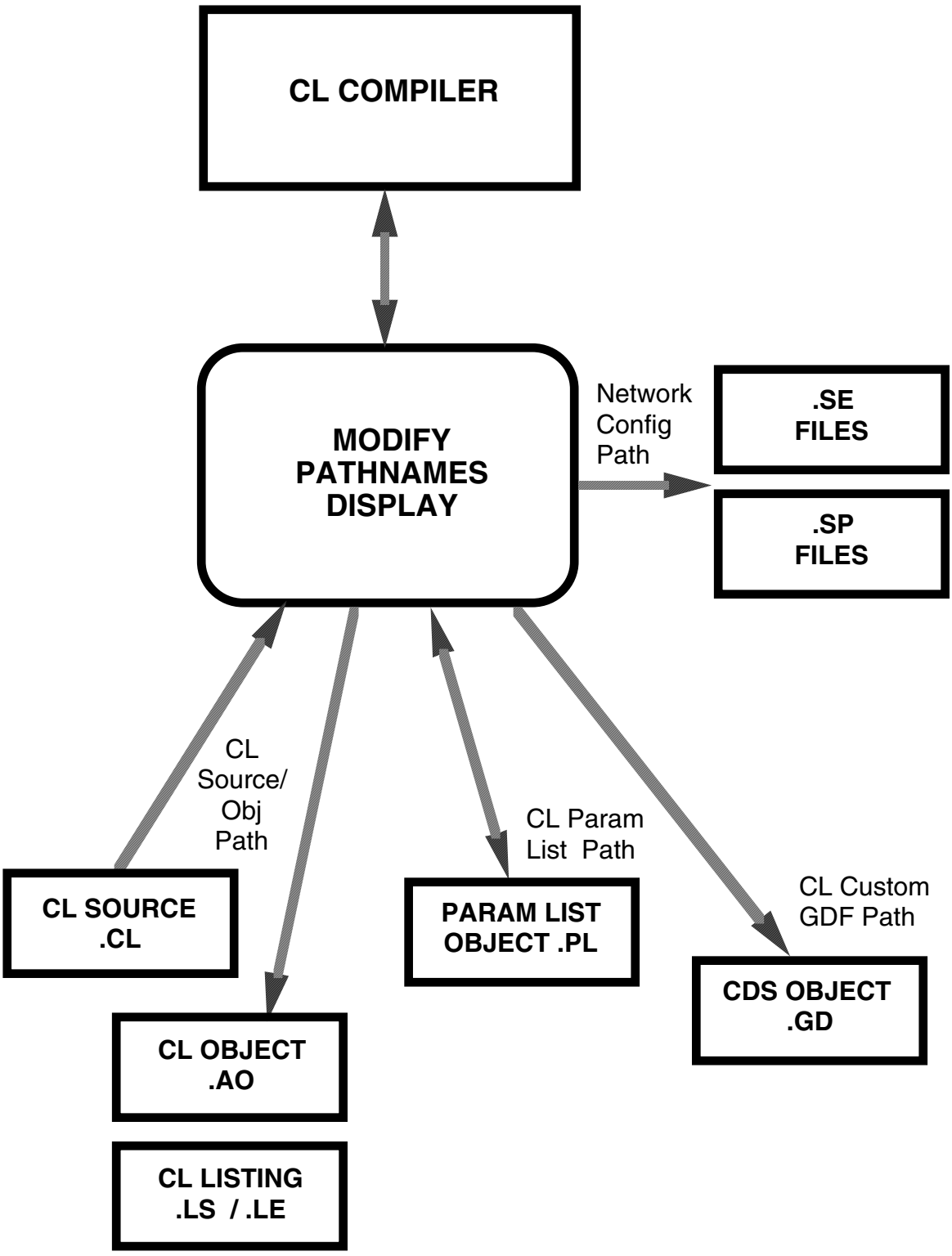
Review of CL/AM Structures

IMPLEMENTATION



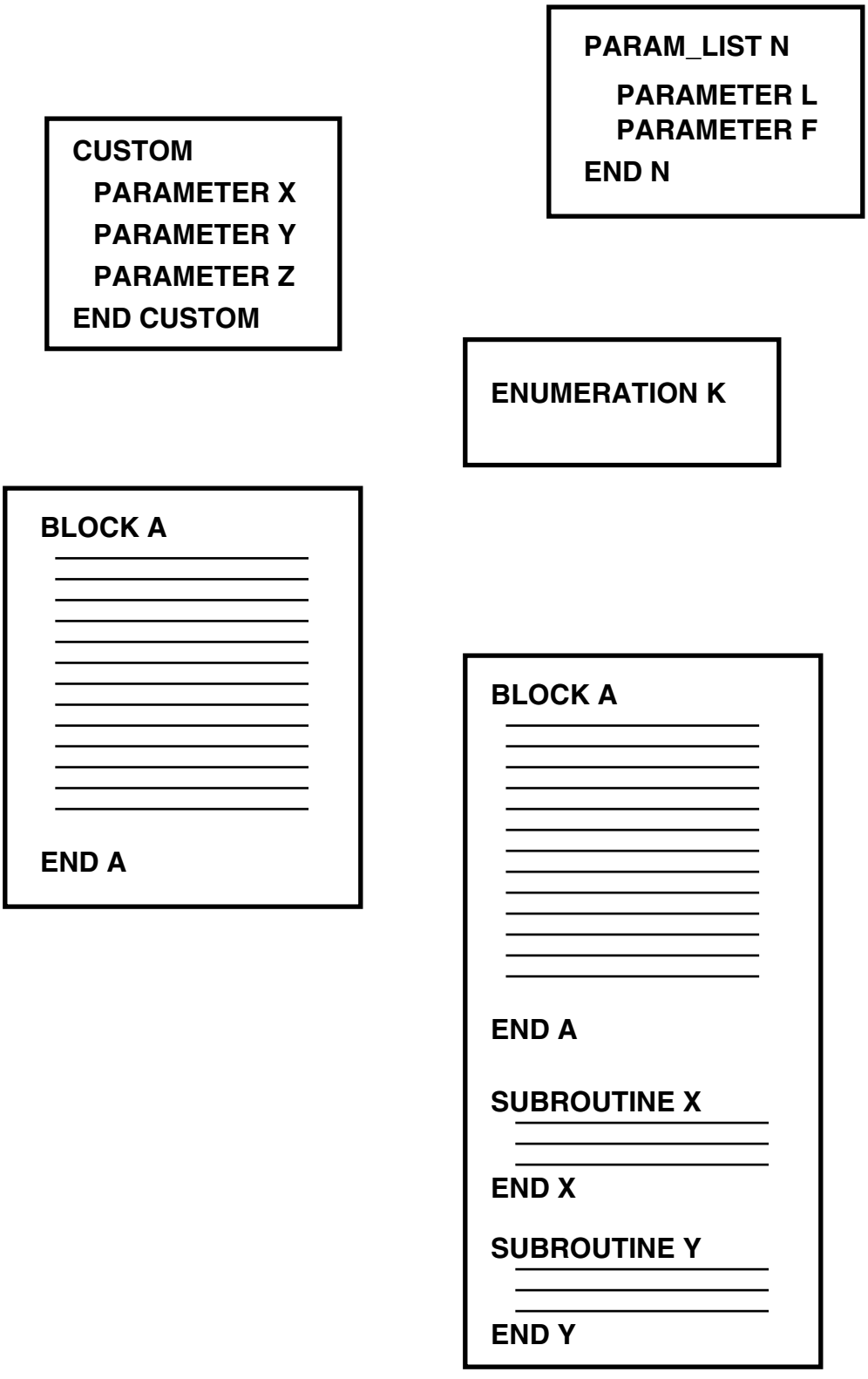
Review of CL/AM Structures

IMPLEMENTATION



Review of CL/AM Structures

CL SOURCE FILES



Review of CL/AM Structures

CL "PACKAGE" SOURCE FILES

```
PACKAGE
  CUSTOM
    PARAMETER X
    PARAMETER Y
    PARAMETER Z
  END CUSTOM
  PARAM_LIST N
    PARAMETER L
    PARAMETER F
  END N
  ENUMERATION K
  ENUMERATION H
  BLOCK A
    ____________________
    ____________________
    ____________________
    ____________________
  END A
  BLOCK B
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
  END B
END PACKAGE
```

```
PACKAGE
  PARAM_LIST N
    PARAMETER L
    PARAMETER F
  END N
  BLOCK A
    ____________________
    ____________________
    ____________________
    ____________________
  END A
  BLOCK B
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
  END B
END PACKAGE
```

```
PACKAGE
  PARAM_LIST N
    PARAMETER L
    PARAMETER F
  END N
  ENUMERATION K
  ENUMERATION H
END PACKAGE
```

```
PACKAGE
  CUSTOM
    PARAMETER X
    PARAMETER Y
    PARAMETER Z
  END CUSTOM
  BLOCK A
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
  END A
END PACKAGE
```

Custom Data Segments

DEFINITION

- Collections of custom-named parameters that can be attached to the following point types:

  AM _____
  AM _____
  AM _____

- Created by writing and compiling a CL source file containing the heading _____, followed by one or more _____ statements, followed by _____.

Example:

```
CUSTOM
     PARAMETER X
     PARAMETER Y
     PARAMETER Z
END CUSTOM
```

Custom Data Segments

ATTRIBUTES

- **Data types :**

  Number (default), String, Logical, Entity (param. list), Enumeration (enumeration set name), Date/Time (Time)

- **Value:**

  Initial Value of Parameter (n/a for type Date/Time)

- **Access Level:**

  Engineer (default), Supervisor, Operator, Entity_Bldr, Program

- **Ped Visibility:**

  Bld_Visible (default), Not Bld_Visible

- **EU Description:**

  " Eight Character String"

- **Class:** (n/a to custom or switch points)

  General (default), PV_ALG, CTL_ALG

Example:

```
CUSTOM
     PARAMETER X
          VALUE 100
     PARAMETER Y: $REG_CTL
          ACCESS SUPERVISOR
          EU " Temp In"
     PARAMETER Z: string
          VALUE "Dye Vat # 1"
          NOT BLD_VISIBLE
END CUSTOM
```

Custom Data Segments

DEFAULTS

- Custom Data Segment default attributes may be changed by denoting the new defaults on the CUSTOM heading.

  Only Access Level, Ped Visibility, and Class may be changed in that manner.

Example:

```
CUSTOM (CLASS PV_ALG; ACCESS OPERATOR; NOT BLD_VISIBLE)
     PARAMETER X : NUMBER
          VALUE 100
     PARAMETER Y
          EU " Temp In"
     PARAMETER Z : string
          VALUE "Dye Vat # 1"
          BLD_VISIBLE
END CUSTOM
```

- If the VALUE option is not used or values were not built in the DEB, parameters will be set to the following default values:

NUMBER	Bad Value
STRING	1 Blank character
LOGICAL	off
ENTITY	NULL_ID
ENUMERATION	First named state
DATE/TIME	0 00:00:00

Custom Data Segments

ARRAYS

- Custom Data Segment Parameters may be created as single dimension ARRAYs.
- An ARRAY may have up to 1000 elements.
- An ARRAY may be indexed by number or enumeration set name.

Example:

```
CUSTOM
     PARAMETER X : number array(1..5)
          VALUE (3,5,8,10,2)
     PARAMETER Y : string array(1..1000)
     PARAMETER Z : array (mode)
END CUSTOM
```

- In the body of a CL program the above parameters would be referenced as follows:

Example:

```
    if X(4) > 8 and Z(A100.mode) = 35 then set Y(4) ="Start
Motor"
```

Custom Data Segments

IMPLEMENTATION

- CDS name is the file name of the object file (`XXXXXXXX.GD`), which is the same as the source file name (`XXXXXXXX.CL`) that was used to create it, even if that file contains other items.

- CDS name is unique and kept in file _____; master is on HM system volume &ASY; copies exist in every running US on the LCN (used for display purposes).

- Parameter names are re-usable and kept in file _____; master on HM, copies in every US. Names can be system parameters (i.e., PV, OP) or previously established CDS parameters.

  Be aware of possible conflicts (see Section 2.2.7.4 in the CL/AM Reference Manual).

- `.GD` object files are stored in the user volume _____ (default).

- CDS and parameter names must be alphanumeric, may include an embedded (_); 8 characters max.

- Up to _____ CDS names can exist on an LCN.

- Up to _____ parameter names can exist on an LCN.

- Once CDS names are established, they cannot be modified or deleted. Therefore, they should be managed carefully.

Custom Data Segments

IMPLEMENTATION

- Create source file in the Text Editor, using "Custom" format. This structure may be in its own file, or may be combined with CL Blocks, parameter lists and enumerations in a Package file.

- Set up MODIFY PATHNAMES to set CL and CDS paths.

- Compile source with CL compiler...

  w/ no options to test
  w/ _____option if first time "real" compilation
  w/ _____ & _____options if editing & adding new names
  w/ _____ option if editing and not adding new names

- Attach CDS to one or more points in the Data Entity Builder; up to _____ CDS packages per point:

  1. Begin building the point in the standard way.
  2. Set the `NOPKG` parameter to the exact number of CDSs to attach.
  3. Enter the CDS file name(s) at the `PKGNAME(n)` parameters. These entries are required.
  4. Page forward to fill in the parameter ports with a default value and press ENTER, or just press ENTER.
  5. Load the point in the usual way.

Custom Data Segments

IMPLEMENTATION

```
                                                        13 Apr 93 15:20:33  6
                                                USER PATH :  NET>S309>
ED TEMP309.CL
S309>TEMP309.CL      File Was Updated -    17 Lines Written
ED TEMP309.CL
S309>TEMP309.CL      File Was Updated -    17 Lines Written
CL TEMP309.CL
Loading Overlay
Elapsed time for  Pass 1      =      2.5 seconds
Elapsed time for  Pass 2      =     14.9 seconds
Elapsed time for  Pass 3/AM   =      0.0 seconds
Elapsed time for  Pass 3/CDS  =      4.6 seconds
Elapsed time for  Listing     =      9.4 seconds
Elapsed time for  Compilation =     35.2 seconds
Errors detected : 1

ED TEMP309.CL
S309>TEMP309.CL      File Was Updated -    17 Lines Written
```

```
                                                        13 Apr 93 15:17:46  6
                          Source File :  NET>S309>TEMP309.CL
                                                TEMP309.LE  LINE 1
?
  CL V40.31     TEMP309                          04/13/93 15:16:04:8461     Page


  Line     Loc  Text

     1          -- CDS FOR CLWRT309.CL (ENTITY NAME)
     2
     3          CUSTOM
     4
     5          PARAMETER TEMP1: $REG_CTL
     6          VALUE RAMP309
     7
     8          PARAMETER TEMP2: $REG_CTL
     9          VALUE FRC309
    10
    11          PARAMETER TEMP3: $REG_CTL
    12          VALUE TC71309
    13
    14          parameter TINI: $reg_ctl
                          ^
  **ERROR   **  "TINI    " was not found in the system and -UL was not selected

  F1    F2     F3    F4     F5    F6     F7      F8
 QUIT  BLOCK  JUMP  MACRO  EDIT   SET   FILE   SBUFFER
```

Custom Data Segments

IMPLEMENTATION

```
                                                    13 Apr 93 15:24:34  6
                                                 USER PATH :  NET>S309>
ED TEMP309.CL
S309>TEMP309.CL      File Was Updated -    17 Lines Written
CL TEMP309.CL -UL
Elapsed time for  Pass 1      =      2.5 seconds
Elapsed time for  Pass 2      =     14.8 seconds
Elapsed time for  Pass 3/AM   =      0.0 seconds
Elapsed time for  Pass 3/CDS  =      4.6 seconds
Elapsed time for  Listing     =      9.8 seconds
Elapsed time for  Compilation =     34.1 seconds
     Existing Custom Data file:
          File TEMP309.GD  not overwritten, per command
Errors detected : None

CL TEMP309.CL -OCD
Elapsed time for  Pass 1      =      3.4 seconds
Elapsed time for  Pass 2      =     14.7 seconds
Elapsed time for  Pass 3/AM   =      0.0 seconds
Elapsed time for  Pass 3/CDS  =      4.6 seconds
Elapsed time for  Listing     =     18.0 seconds
Elapsed time for  Compilation =     42.8 seconds
     Custom Data file:
          File TEMP309.GD  overwritten
Errors detected : None
```

Enumerations

TYPES

- An Enumeration-type definition lists all the states that a discrete variable can take on; e.g., ON/OFF/IN_BETWEEN, or RED/BLUE/GREEN/AMBER.

  An Enumeration description defines an Enumeration type,listing each state that can be assumed by a variable of that type.

- There are four types of Enumeration Data Types:

  _____ Enumerations

  _____ Enumerations

  _____ Enumerations

  _____ Enumerations

- System-Defined or Honeywell-defined standard Enumerations supplied with the system in support of standard point parameters. Examples include MODE, PTEXECST, and ALENBST.

- User-Defined Enumerations are defined in CL enumeration statements in support of Custom parameters.

- Self-Defined Enumerations where the user defines the state names at point build time for PVs of Digital points, Flag points and Switch points.

- Enumerations defined in the CL context only.

Example:

```
Local light: red/amber/green
```

Enumerations

IMPLEMENTATION

- User-Defined enumerations are used to define the state names of Custom Parameters. They are created in a CL statement beginning with the word ENUMERATION , the Name of the enumeration, followed by "=" and the enumeration states.

Example:

```
Enumeration Traflite = red/amber/green
```

- The enumeration set name (**Traflite**) may be used to define both Custom Parameters and Local variables.

Example:

```
CUSTOM
     PARAMETER X: Traflite
END CUSTOM

BLOCK XYZ (POINT XXXXX; AT GENERAL)
     ..............
     LOCAL Y: Traflite
     ..............
END XYZ
```

- The first time a user-defined enumeration is implemented in a CL program, it must be compiled with a **-UL** command option to add the enumeration to the .SE library.

- A maximum of 200 Custom Enumeration Sets are permitted on an LCN. A maximum of 1000 Enumeration State names are permitted. A State name appearing on more than one enumeration set counts as one of the 1000 allowed.

External Reference

POINT IDENTIFIERS

- CL statements can read/write/compare the parameters of other LCN points (those external to the bound data point) through both **Direct** and **Indirect** references to point and parameter names.

- **Direct** references are made via External declarations in the CL program.

Example:

```
BLOCK ABC (POINT F100; AT GENERAL)

EXTERNAL A100

     IF A100.PV > 50 THEN.............

END ABC
```

- **Indirect** references are made via Custom Data Parameters of type ENTITY. The value of this parameter is the name of the desired reference point.

Example:

```
PACKAGE
CUSTOM
     PARAMETER X: $REG_CTL
     VALUE A100
END CUSTOM

BLOCK ABC (POINT F100; AT GENERAL)
     IF X.PV > 50 THEN.............
END ABC
END PACKAGE
```

Local Declarations

TYPES

- There are two types of local declarations. They are Local Constants and Local Variables. Local values declared in a CL program are visible only inside the program they are declared in. A Local constant/variable name may be up to 40 characters long, however, a local declaration may not exceed one line.

Example:

```
Local a = 50                    -- a numeric constant
Local b = a                     -- a constant expression
Local c = 2 HOURS 5 MINS        -- a time constant

Local e : red/green/amber       -- variable enumeration
Local f : Traflite              -- variable enumeration
Local g : logical               -- variable logical
Local current_message_num       -- variable number
Local i : time                  -- variable time
```

- Local variables require a SET statement to initialize the value.

Example:

```
set e = green
set f = red
set g = on
set current_message_num = 50
set i = NOW         --returns current time
```

Local Declarations

ARRAYS

- Local Variables may be defined as one- or two-dimensional arrays. This differs from a Custom Parameter that may be only one-dimensional.

- An array index must be a discrete type (declared by enumeration set name) or a number declared by naming lower and upper bounds, in that order.

- An array indexed by number must be in integer form. The value of the lower bound must be less than the upper bound.

Example:

```
Local A: Logical Array (1..5)    -- an array
Local B: Array (1..16, 1..16)    -- a 2-d number array
Local C: Number Array (Mode)     -- indexed by enumeration
Local D: Traflite Array (1..5)     -- enumeration array
```

- In the body of a CL program the above Local variables would be referenced as follows:

Example:

```
if A(3) = off and B(2,5) > 15 then (set C(A100.mode) = 20 ;
&    set D(4) = red)
```