Honeywell

Control Language Application Module Data Entry

AM11-585

Implementation
Application Module - 3

Control Language Application Module Data Entry

AM11-585
Release 500
9/95

Copyright, Trademarks, and Notices

Revision 01 - September 1, 1995

About This Publication

This publication provides instructions for the following tasks:

- Creating CL source files, and entering and editing data in the source files
- Using the Engineering Personality's Utilities to manage and manipulate files related to CL
- Using the CL Compiler/Linker commands and options that are part of the `COMMAND PROCESSOR` activity on the Engineering Main Menu

This manual supports TDC 3000^{X} software Releases 500.

Table of Contents

Table of Contents

INTRODUCTION
Section 1

This section provides

- *A description of the intended use for this publication.*
- *References to other publications needed or useful for compiling, linking, and loading CL packages.*

1.1 WHO THIS PUBLICATION IS INTENDED FOR

This publication is prepared for use by people such as process engineers, control-system engineers, and application engineers who need to install custom functions in a TDC 3000X System, using Control Language for the Application Module (CL/AM).

1.2 HOW TO USE THIS PUBLICATION

Use this publication when

- You need to determine what work must be accomplished before you can install or modify a CL program. Section 2 should be especially useful for this purpose.
- You are ready to install a new CL program or to change an existing CL program.

The expected use of Sections 2 through 5 of this publication is as follows:

- **Section 2 – Prerequisites**—Use this section to determine what must be available to install or to change CL programs and what startup tasks must be completed.
- **Section 3 – Typical CL Data Entry Sessions**—Use this section to review the major steps to install or change a CL program and the recommended order of those steps.
- **Section 4 – Engineer's Keyboard**—Use this section when you don't know or are not sure of the effect of pressing a key on the Engineer's Keyboard.
- **Section 5 – Commands**—Use this section to determine the detailed steps in executing the commands used to install or to change CL programs, and for a description of the functions of the CL compiler and linker.
- **Section 6 – Error Indications and Recovery**—This section provides guidance in recovering from human errors.
- **Index**—An alphabetized list of CL data entry topics with references to paragraph numbers in Sections 1 through 6.

Before using the procedures in Sections 3 through 5 of this publication, you should review the publications mentioned under subsection 1.3.

1.3 RELATED PUBLICATIONS

The following are publications that contain reference information related to CL data entry, or provide procedures that are related to CL data entry.

- *Control Language/Application Module Overview, SW27-400*, in the *Implementation/Application Module - 2* binder. If you are just beginning to work with CL, start with this publication. It provides an overview of the complete CL implementation process, from determination of the need for CL-based functions, through the design of those functions, their installation, testing, and use.

- *Control Language/Application Module Reference Manual, AM27-410*, in the *Implementation/Application Module - 2* binder. This manual defines the Control Language, including its environment. The language's elements, syntax, and rules are defined. Also defined are the relationships of the parts of CL programs and their sizes and limitations.

- **Control Functions**—This is a set of two publications, one that covers data acquisition and control functions that are independent of the nodes that participate in the functions, and one for AM-specific items. The publications are

  System Control Functions, SW09-401, in the *Implementation/Startup & Reconfiguration - 2* binder.

  Application Module Control Functions, AM09-402, in the *Implementation/Application Module - 1* binder.

- *Application Module Algorithm Engineering Data, AM09-401* in the *Implementation/Application Module - 1* binder. Describes the functions of all algorithms available in Application Modules, including the CL PV Algorithm and the CL Control Algorithm.

- *Engineer's Reference Manual, SW09-405*, in the *Implementation/Startup & Reconfiguration - 2* binder. Provides guidance and tips that are useful in designing and implementing the system configuration, and in starting up the System.

- **Parameter References**—These publications provide information about standard parameters, including parameter names, data types, value ranges, and access levels. The publications are

  Application Module Parameter Reference Dictionary, AM09-440, in the *Implementation/Application Module - 1* binder.

  Computer Gateway Parameter Reference Dictionary, CG09-440, in the *Implementation/Computer Gateway, Implementation/Processor Gateway*, and *Implementation/CM50S* binders.

- *System Startup Guide, SW11-403* (floppies), or *SW11-404* (cartridges), in the *Implementation/Startup & Reconfiguration - 1* binder. We recommend that you use one of these guides the first time you use any of the activities on the Engineering Main Menu. They provide step-by-step procedures for all available configuration activities and define all system startup tasks. Task 28 is Build, Compile, and Load CL Programs.

- *Data Entity Builder Manual, SW11-411*, in the *Implementation/Engineering Operations - 1* binder. This publication describes the use of the Data Entity Builder (DEB). The DEB is used for eight of the activities on the Engineering Main Menu. Of those eight, data-point building is the most important to CL data entry, because it is used to build the data points that CL programs work with, and to attach Custom Data Segments to data points.

- *Command Processor Operation, SW11-407*, in the *Implementation/Engineering Operations - 1* binder. This publication provides instructions for the use of all of the functions available through the Utilities pick on the Engineering Main Menu. Utilities are used to set up default volume pathnames, to manage volumes and files, including those used by CL, to catalog and list the content of volumes and files, and to display and print the content of CL files.

- *Text Editor Operation, SW11-406*, in the *Implementation/Engineering Operations - 1* binder. This publication defines how to use the Text Editor. It is used to create CL source files on user volumes, and to enter and edit information in those files.

- *Process Operations Manual, SW11-401*, in the *Operation/Process Operations* binder. This manual provides instructions for use of the Universal Station's Operator personality. CL/MC (sequence) programs are loaded in the MC through this personality.

PREREQUISITES FOR A CL DATA ENTRY SESSION
Section 2

Use this section to determine what must be available to install or to change CL programs, and what startup tasks must be completed.

2.1 PRECEDING SYSTEM STARTUP TASKS COMPLETED

You can create CL source files, enter data in them, and edit them at any time. Generic CL/AM programs can be compiled any time after Task 16 is completed, and specific programs can be compiled any time after System Startup Task 21 is completed. CL/AM programs can be linked any time after Task 21 is completed. See the Tasks shown in the *System Startup Guide* for further information.

CL data entry consists of creating a source file, compiling source files, linking the resulting programs, attaching Custom Data Segments to data points, and loading CL programs into AMs, where they will be executed.

For CL/AM programs, linking includes binding to designated data points and loading into the Application Module (node) that contains that point. Before a Link command can be executed, the point must be loaded in its node; the node must be running; its status must be OK; and the point must be inactive. Also, all points referred to by the program, either directly or indirectly, must be built and loaded. The process-connected boxes don't need to be attached.

2.2 USER VOLUMES ESTABLISHED WITH SUFFICIENT CAPACITY

You will store your CL source files on user volume removable media (floppies or cartridge disks) or in user volumes on an HM. The CL Compiler/Linker creates listing files, error-listing files, and object files on the volume that contains the source file (Table 2-1 lists all of the volumes used in CL data entry). You will find it easiest to use the HM.

Whether you use removable media or the HM, you must configure one or more user volumes with sufficient capacity to store all of your source files and the Compiler/Linker-generated files. If you will be building Custom Data Segments for AM points, you will find it more convenient if you create separate volumes for your CL source/object files (volume CL) and for your CL custom GDF files (volume CDSG).

Unless you create an unusually large number of CL programs, the CL user volumes will be much smaller than those that contain IDFs with DEB-built entities. Guidelines for estimating the size of CL-user volumes are provided under subsection 7.2 in the *Engineer's Reference Manual*.

You can use Task 4 (removable media) or Task 10 (HM) to configure your user volumes (see subsection 3.9.2). User volumes should be configured in Task 10 during the initial startup, but the HM volume configuration can be changed to include user volumes by using the on-line reconfiguration procedures (see *Network Data Entry* in the *Implementation/Startup &Reconfiguration - 1* binder). For a more detailed explanation of floppy formatting, refer to *Command Processor Operation* in the *Implementation/Engineering Operations - 1* binder.

We recommend that you periodically copy any CL-user files you have on an HM to removable media, so that you can recover them should something cause the data on the HM to be lost. Use the Utilities' Copy command to do this.

2.3 ENGINEERING PERSONALITY RUNNING AND NEEDED VOLUMES AVAILABLE

All of the activities described in this publication take place at a Universal Station with the Engineering Personality running. To begin any of these activities, the Engineering Main Menu must be on the screen. To load the Engineering Personality and to get the Main Menu on the screen, you can use Startup Task 2 or refer to Section 5 of the *System Startup Guide*.

Table 2-1 lists all of the volumes and files involved in installing and changing CL programs and Custom Data Segments. These volumes should be available on an HM or on removable media, as needed. It is easier to have them on an HM, so mounting and dismounting disks isn't necessary.

2.3.1 Universal Station and Volume Handling Guidelines

CL data entry requires a Universal Station that has an Engineer's Keyboard. It's useful to have a printer connected to the US so that you can make printed records of your work. If you are using removable media, the US should have two drives connected to it, so that the transfers will be faster than if they have to go over the LCN.

Execution of any command that can change the system database or the process database requires the key switch on the Operator Keyboard to be in the ENGR position. It is easiest to check the switch when you start a session and leave it in ENGR until you are finished.

It is very helpful to have another Universal Station in the same console with the Operator Personality running. You can then look at the detail displays for the points associated with your CL programs, to verify that parameter values mentioned in this publication are as described, and to see the effects of your work.

It is easiest to have one or more user volumes for your CL files (see Table 2-1) on an HM (Startup Task 10) and to have the complete software complement on an HM (Startup Task 16). If you use removable media, it's easiest to use one user volume for all CL files, and you need to mount and dismount disks as needed. For example,

- When using the Text Editor or Utilities, &OV1 should be in one drive and your user volume in the other.
- When you enter CL from the COMMAND PROCESSOR to use the CL Compiler/ Linker, &OV2 should be in one drive and your user volume in the other. In some situations, you need to replace &OV2 with &ASY after the Control Language Compiler/Linker display appears. The procedures in Sections 3 and 5 tell you when.

Before you begin any activity on the Engineering Main Menu, you should select SUPPORT UTILITIES and check the Modify Default Volume Path Name display to see that the correct sources ($Fn or NET) and the correct volume names are entered. If not, change and then press ENTER.

Table 2-1 — Volumes and Files Used in Installing and Changing CL Programs

Volume	File Name and Suffix	Use
&OV1	(Several)	Engineering Overlay 1—contains the Text Editor, Utilities, and Data Entity Builder (DEB)
&OV2	(Several)	Engineering Overlay 2—contains the CL Compiler/Linker
&ASY	(Several .SE, .SP, and .PL)	Contains the Custom Name Library and the Standard including Parameter Lists (also contains the NCF)
	Duplicat.SP	Contains a list of the matches between the new standard parameter names and existing custom parameter names.
&AMG	(Several)	Contains the standard AM and CM GDFs
&CLX	(filenames).SF	Contains subroutine and/or function declarations of CL runtime extensions (created by Honeywell) that are callable by CL/AM programs.
&CUS	(filenames).LO	CL runtime extension external objects that are linked into the Application Module during node initialization.
CL	(filename).CL	CL source file (created by the user)
	(filename).AO	CL/AM object file (created by the Compiler)
	(filename).LS	CL listing file (created by the Compiler)
	(filename).LE	CL error listing file (created by the Compiler)
	(filename).PL	CL Custom Param List (created by the Compiler)
CDSG	(filename).GD	Custom GDF (created by the Compiler)

Note: CL and CDSG are user volumes that can have any user-defined names. They can also be the same volume with just one name. CL and CDSG are the default names that appear on the Modify Default Volume Path Names display. You can change the names on that display to the name(s) of your user volume(s).

CL Parameter Lists can also be on a separate volume.

2.4 PREREQUISITES FOR A CL/AM BLOCK COMPILE AND LINK SESSION

Use this check list before you begin to compile and link a CL/AM structure:

A. The following types of AM data points can have CL blocks linked and Custom Data Segments attached to them:

   Regulatory Switch Custom

B. If you are compiling a specific CL block that has one or more direct references to point data, the points referred to must be built and loaded with the DEB before the CL block is compiled. This is not necessary for a generic CL block and indirect references.

C. The bound data point must be built, configured for CL (parameter CLSLOTS > 0), and inactive (PTEXECST = Inactive) before a block can be linked (bound) to the point.

D. There must be a slot available in the CL segment to accept a new block (CLSLOTS can specify from 1 to 255 slots).

E. Any Custom Data Segment used by a CL block, whether the CDS is on the bound data point or on other points, must be attached to that point(s) through the DEB before linking the block.

F. A block with the same name as one that already exists on the point cannot be linked.

G. A block with the same name as, but which is different from, the block being linked cannot be present in the same unit in the destination AM.

H. If an insertion order is specified in a block's header, a block with the same order indication must not already be linked to that insertion point.

I. A CL block cannot exceed 16k words and the point-parameter references made by the program cannot exceed 16k words. Information on the number of words used is included in the CL listing (.LS) file.

If you need to overcome condition F or H, you can use the Unlink command (see subsection 5.3.2.3) to remove the existing block and then link the new block.

2.5 PREREQUISITES FOR A CDS COMPILE AND LOAD SESSION

Check the following before proceeding to compile and load a Custom Data Segment:

A. CDSs can be attached to these types of AM and Computing Module data points:

Regulatory (AM)	Advanced Control Interface Data Point (CM50S/CM60)
Custom Data Point (AM)	Calculated Results Data Point (CM50S/CM60)
Switch Data Point (AM)	

B. For a custom-data compilation, the **Network Configuration File (NCF)** must be installed (System Startup Task 11). If an HM is present, the HM must be configured and operational (System Startup completed through Task 16).

NOTE

The same &ASY volume that contains the NCF must be used throughout all custom-data compilation, related DEB activities, and CL block compilation. This is because the custom names that are established by the custom-data compilations are held in the Custom Name Library files on volume &ASY.

If the Custom Name Library files are lost or corrupted, copies of the libraries can be obtained from any operational Universal Station loaded with the Operator's Personality. This is accomplished through the Custom Names function on the System Menu.

Also see 3.9—Custom Name Library Synchronization.

C. The DEB is used to attach a CDS to a data point. This requires that the destination **AM** or **CM's Computer Gateway** be fully configured and operational. CDSs can be attached as a point is initially built, or by reconstituting the point, changing selections and values, and reloading the point (this process is described in Section 3).

D. When a point with a CDS is loaded, there **must not** be a CDS on another point in that node with the same CDS-descriptor name (source-file name when compiled) and a different time stamp. (i.e., you can't have two different CDS descriptions with the same name in the same node at one time. The system assumes that two CDSs with the same name and differing time stamps are different.)

TYPICAL CL DATA ENTRY SESSIONS
Section 3

3.1 HOW TO USE THIS SECTION

Typical CL data entry sessions are described in this section for these two reasons:

- To provide an overview of typical sessions for new users.
- To define the order in which we recommend you accomplish the major steps.

References to detailed instructions for these major steps are provided, and Figure 3-1 provides an overview of the entire CL/AM program installation process.

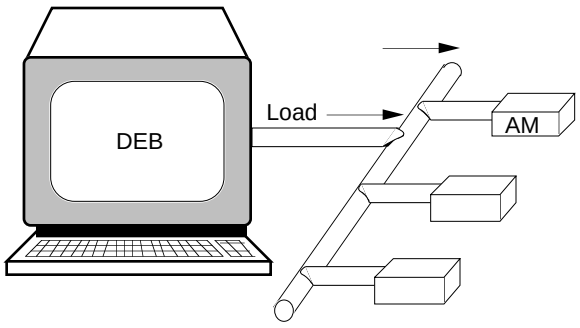
The chart on Figure 3-2 guides you to the appropriate paragraph in this section for the activity you are interested in, and indicates the volumes involved in those activities. It doesn't necessarily indicate the order of the activities.

3.2 INSTALL NEW CL/AM BLOCKS

3.2.1 Compile and Link to a Single Data Point

Use the following major steps to install a CL/AM block that contains only CL/AM blocks on a single point. All blocks name the same point in the block heading.

1. Use the Data Entity Builder (DEB) to build the point that the block or package will be linked to. Parameter CLSLOTS on the AM General I/O and Custom Parameter Entry Display (PED) must specify the number of blocks to be linked to the point. (See the *Data Entity Builder Manual*.)



1345

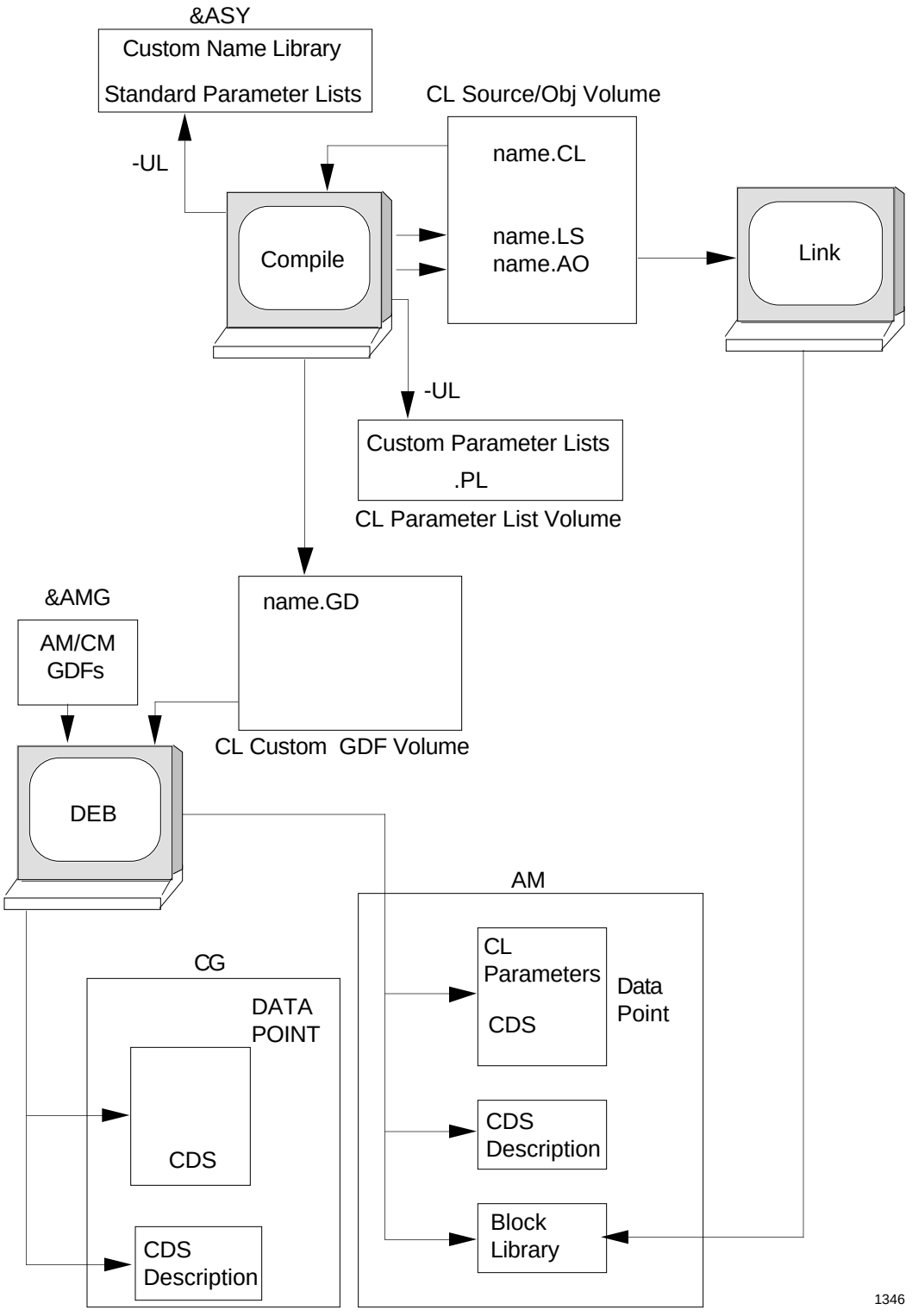


Figure 3-1 — Overview of CL/AM Program Installation

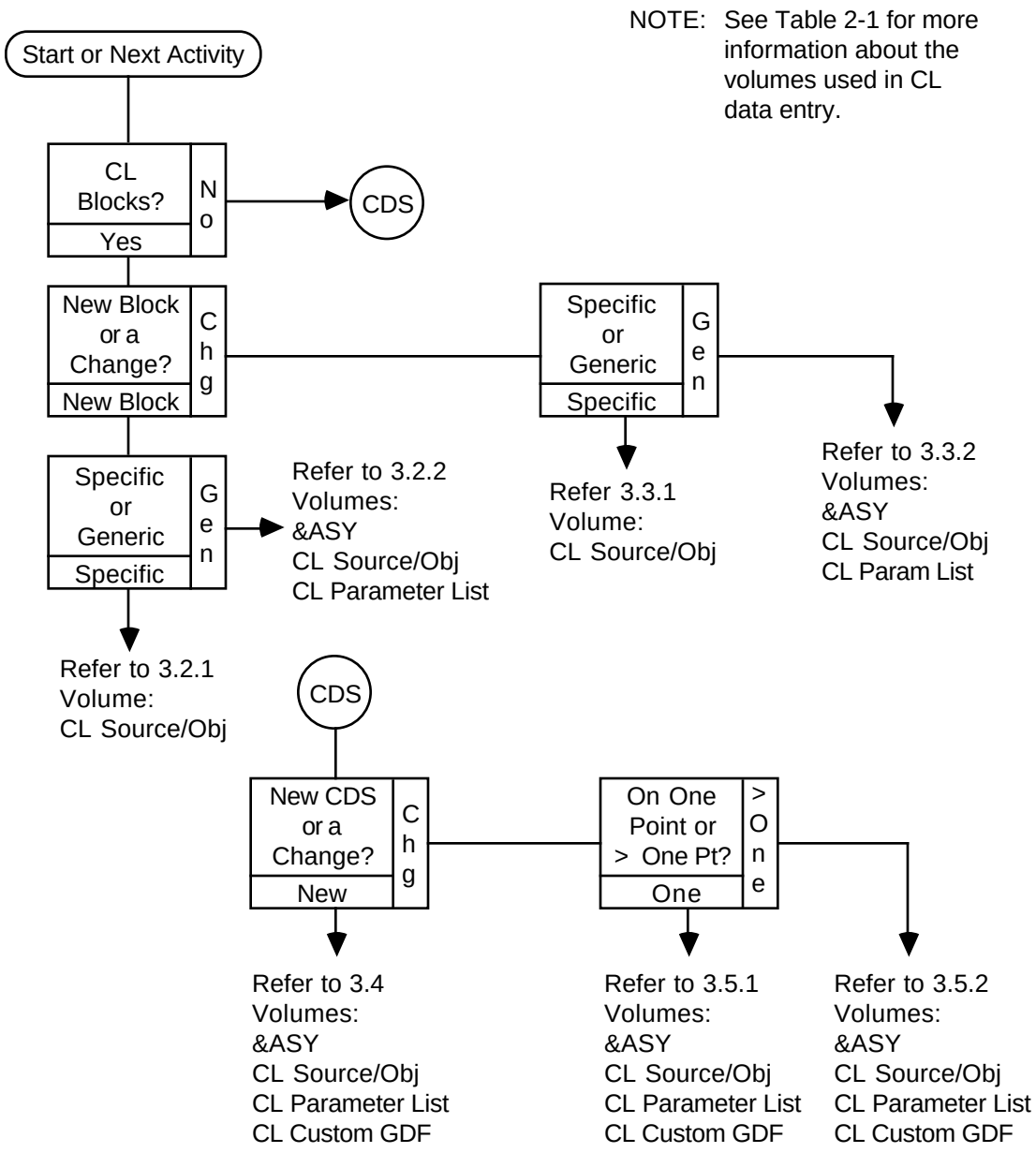
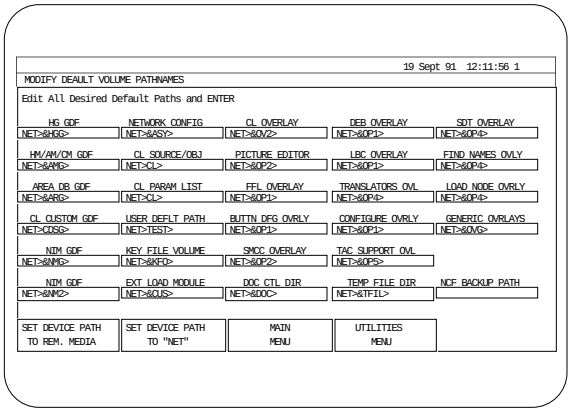


Figure 3-2 — CL Data Entry Activities and Volumes Involved 1347

2. From the Engineering Main Menu select SUPPORT UTILITIES and then select MODIFY VOLUME PATHS. On the Modify Default Volume Path Name Display check that the pathnames for the following volumes are correct (see Table 2-1):



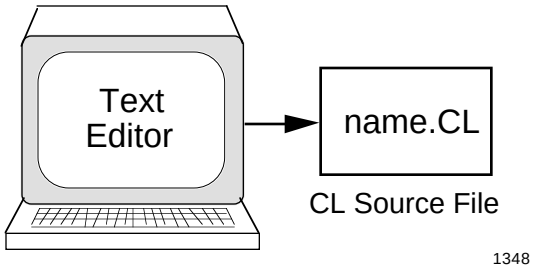
2981

CL SOURCE/OBJ
examples; $F2>CL> or NET>CL>

CL OVERLAY
examples; $F1>&OV2>
or NET>&OV2>

If necessary, correct the pathnames and press ENTER. The display reappears with all pathnames in blue.

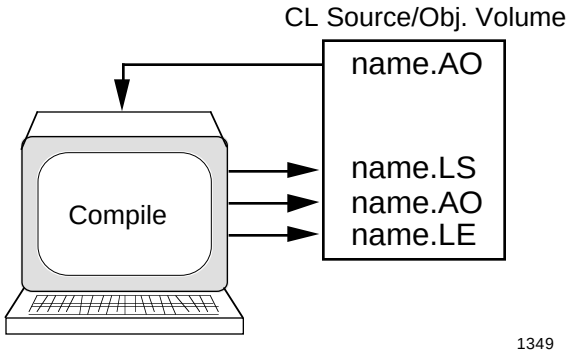
3. Use the Text Editor to create the CL source file on the CL Source/ Object volume named in step 2 (see subsection 5.2).



1348

4. On the Engineering Main Menu, select COMMAND PROCESSOR, type CL and press ENTER. The Control Language Compiler/Linker display appears.

5. See the checklist under 2.4, then Compile the CL source file (see subsection 5.3.2). If no errors are detected, these files are created on the (.CL) user volume:



1349

(filename).LS—the listing file

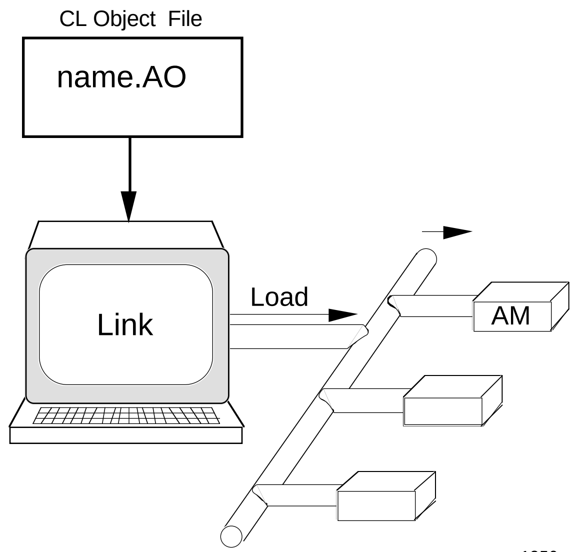
(filename).AO—the object file

If one or more errors is detected, this file is created on the user volume:

(filename).LE—error listing

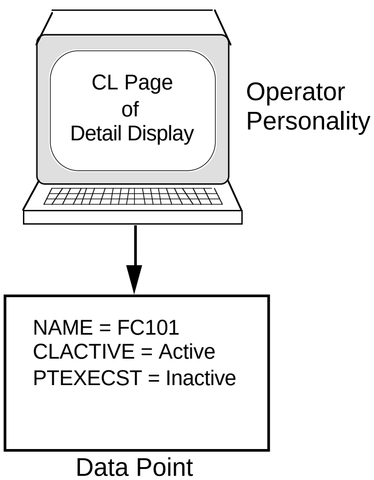
(filename) is the same name you gave your CL source file.

6. Link the CL object file (.AO, see subsection 5.3.2). The Linker checks that all parameters referenced are present on the network and that the data-types for each parameter are as defined to the Compiler.



1350

If the program is successfully linked and loaded, the block name(s) of the newly loaded block(s) can be seen on the CL page of the Detail Display for the point. The Linker leaves the CLACTIVE value for the block Active. The point-execution state remains inactive (PTEXECST = Inactive). To cause the point to be processed and the blocks to be executed, at the Detail Display, change PTEXECST to Active (see the *Process Operations Manual*) and process special the point.



1351

If an error is detected while linking only one block, linking terminates. If the source file contains more than one block and a linking error is detected, linking may be terminated after some blocks have been linked and before others are.

```
                                                     05 Dec 85  09:16:00  3
CDS          TEST POINT                       UNIT 03 UNIT 03                PAGE  2
                       CL BLOCK INFORMATION - ERRSUM NOERROR

             BLOCK BLOCK  ACTIVITY   TEST INSERTN  INSERTN  BLOCK    BLOCK TIME
             INDEX NAME                   POINT    ORDER    ERROR

                1  ONE      ACTIVE    OFF GENERAL        1  NOERROR  05 Dec 85  09:03
                2  TWO      ACTIVE    OFF GENERAL        2  NOERROR  05 Dec 85  09:03
                3  LAST     ACTIVE    OFF GENERAL        0  NOERROR  05 Dec 85  09:03
                          NOTCONFG
                          NOTCONFG
```

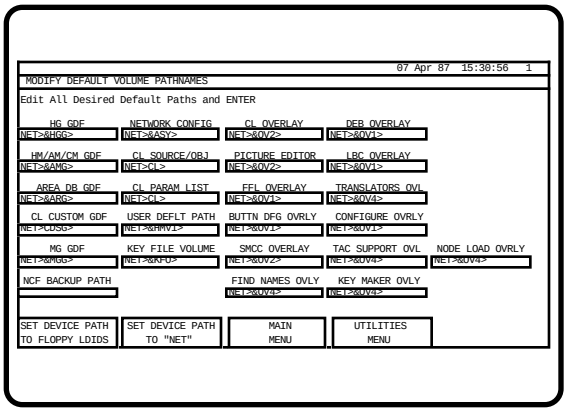
3114

3.2.2 Compile and Link Generic CL/AM Blocks to Multiple Points

Use the following major steps to install a single generic CL/AM block, or a package containing only generic CL/AM blocks, to multiple points. Generic blocks contain the reserved word `GENERIC` in the heading and can be linked to more than one point, one point at a time.

1. Perform step 1 under subsection 3.2.1 (build the data point). This step is normally done first for generic blocks that use indirect references, but it is not actually required until just before step 5.

2. From the Engineering Main Menu, select `SUPPORT UTILITIES` and then select `MODIFY VOLUME PATHS`. On the Modify Default Volume Path Name Display check that the pathnames for the following volumes are correct (see Table 2-1):



3113

`NETWORK CONFIG`
examples; `$F2>&ASY>`or `NET>&ASY>`

`CL SOURCE/OBJ`
examples; `$F1>CL> or NET>CL>`

`CL PARAM LIST`
examples; `$F1>CL> or NET>CL>`

`CL OVERLAY`
examples; `$F1>&OV2> or NET>&OV2>`

If necessary, correct the pathnames and press ENTER. The display reappears with all pathnames in blue.

Note that in the removable media pathname examples, above, &ASY is in drive 2 and the other three volumes are in drive 1. To minimize disk/diskette exchanges, the source/object volume and the parameter-list volume can be the same volume (the same disk/diskette), as they are in the examples. You need the overlay disk/diskette, &OV2, to be in drive 1 when you enter CL from the COMMAND PROCESSOR display, and then you can put your user volume (CL in the examples) in drive 1 for the remainder of the session.

3. Use the Text Editor to create the CL-source file on the CL Source/Object volume named in step 2 (see subsection 5.2).

4. Compile the source file as in step 4 under subsection 3.2.1.

5. Perform step 6 under subsection 3.2.1 for each point the generic block(s) is to be linked to. The block(s) is loaded only when linking to the first point in the unit. The separate links establish communication to all of the points.

3.3 MAKE CHANGES TO A CL/AM BLOCK

NOTE

The number of CL slots on an AM point can be increased, without any impact on CL blocks already loaded, by using the DEB to increase the value in parameter CLSLOTS. Once established, the number of CL slots can be reduced only by using the DEB to delete the point and then to rebuild it.

Certain precautions must be observed when deleting points. Refer to subsection 9.5 in the *Engineer's Reference Manual* and to subsection 7.2.8 in the *Data Entity Builder Manual.*

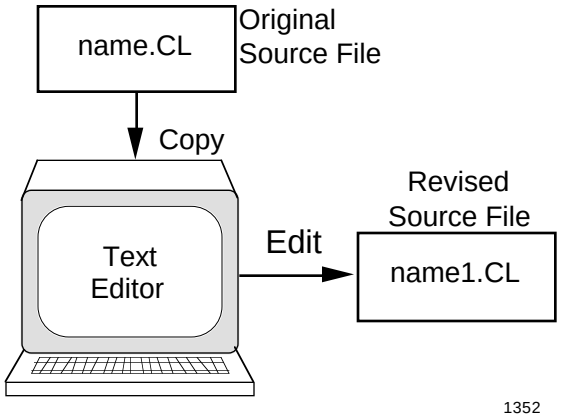
3.3.1 Change a Block or a Package Bound to a Single Data Point

Use the following major steps to change a single CL/AM block, or a package that contains only CL/AM blocks, bound to one AM point.

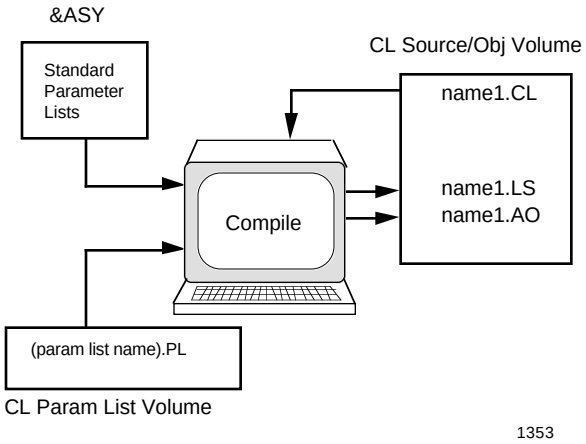
1. Check the volume pathnames as in subsection 3.2.1, step 2.

2. Use the Text Editor to change the source file (see subsection 5.2).

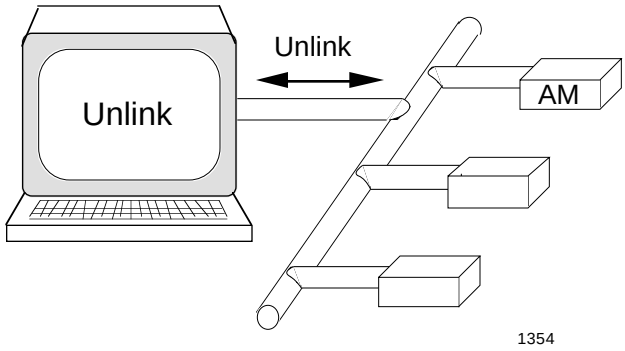
   Note: You can edit the existing source file, or copy and rename it, and then edit the new file. Editing the existing file is more convenient, but it's probably safer to copy and edit a new file, because you won't change the source file for a block that is already working. In either case, use a new name for **each block**. The block name is in the block header in the CL-source file.



3. Compile the new block(s) or the new package (see subsection 5.3.2).

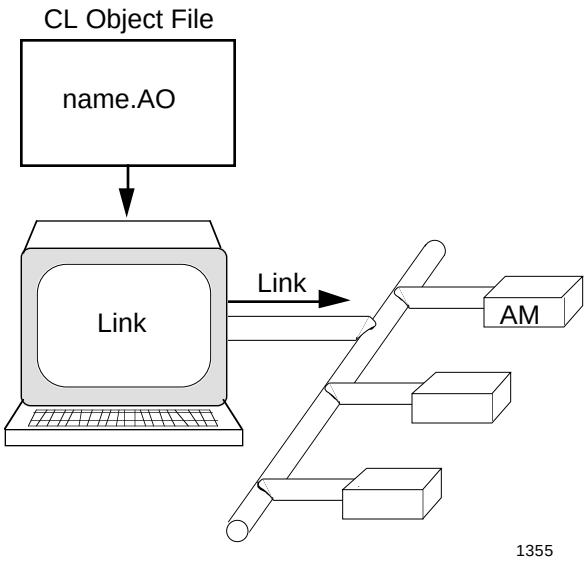


4. Use the Unlink command (see subsection 5.3.2.3) to unlink each block in the package.



5. Link the new block(s) or the new package (see subsection 5.3.2).

6. If you copied and edited a new source file(s), after you successfully link your new block(s), you can use the Utilities' Delete command (see subsection 5.1) to delete the original source file(s).



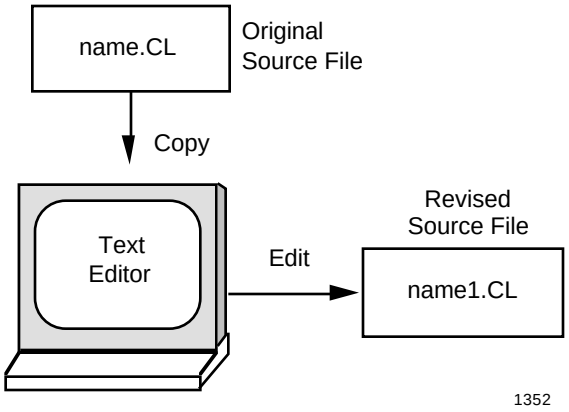
3.3.2 Change Generic CL/AM Blocks Linked to Multiple Points

Use the following major steps to change a single generic CL/AM block, or one or more blocks in a package, that contains CL/AM blocks that are linked to multiple points.

1. Check volume pathnames as in subsection 3.2.1, step 2.

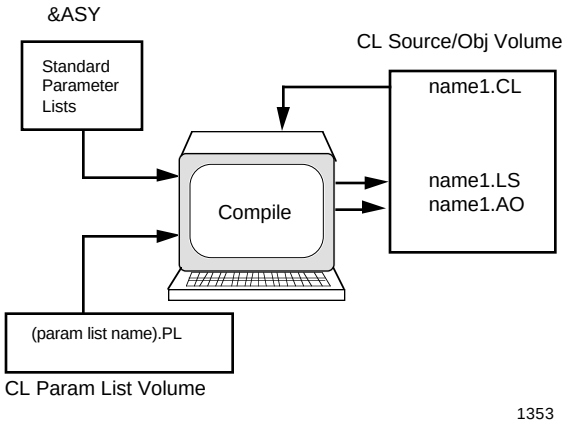
2. Use the Text Editor to change the source file (see subsection 5.2).

   Note: Two different blocks with the same name cannot exist in the same process unit; therefore, when you change a generic block that is linked to more than one point, you should use new block names and a new source-file name. This lets you link the blocks and test them on one point, without affecting the current operation of other points. Also, the original object file is still available if you want to return to the original situation.



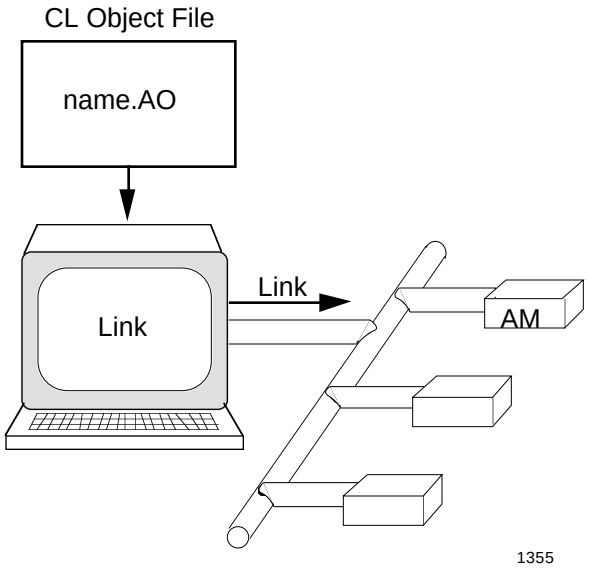
   You can use the Text Editor to copy the original source file, give it a new name, and change the block name or names, then make the changes by editing the new file.

3. Compile the the new block or package (see subsection 5.3.2).

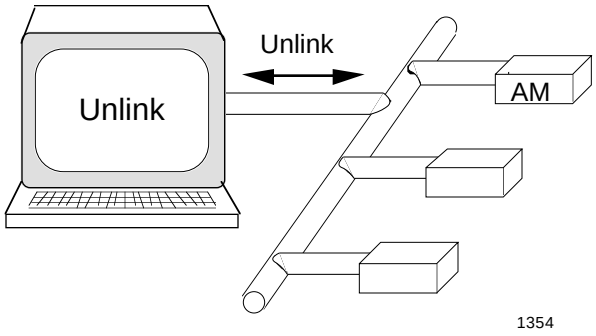


4. Unlink the old block from one of the points and link the new block(s) to the point (see subsection 5.3.2) to test operation with the new block.

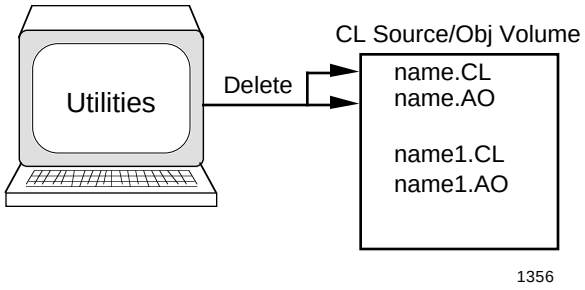
   Note: If you forget to unlink the old block(s), after you link the new block you will have both the old and new blocks on the point, and when the point is active both blocks will be executed.



5. When you are satisfied with the operation of the new block(s), unlink the old block from the rest of the points, one point at a time, and link the new block to each of them.



6. When the original block is no longer needed, use the Utilities to delete its source and object files (see subsection 5.1).



3.4 INSTALL A NEW CUSTOM DATA SEGMENT

Use the following major steps to build, compile, and install a new Custom Data Segment (CDS) in an AM or a CM50/60.

1. From the Engineering Main Menu, select SUPPORT UTILITIES and then select MODIFY VOLUME PATHS. On the Modify Default Volume Path Name Display check that the pathnames for the following volumes are correct (see Table 2-1):

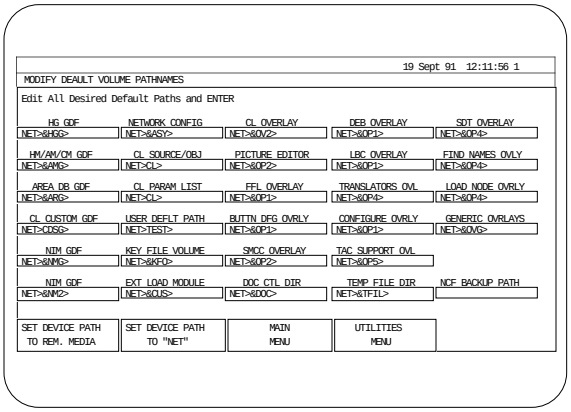
   CL CUSTOM GDF
   examples; $F1>CDSG> or NET>CDSG>

   NETWORK CONFIG
   examples; $F2>&ASY> or NET>&ASY>

   CL SOURCE/OBJ
   examples; $F1>CL> or NET>CL>

   CL PARAM LIST
   examples; $F1>CL> or NET>CL>

   CL OVERLAY
   examples; $F1>&OV2> or NET>&OV2>

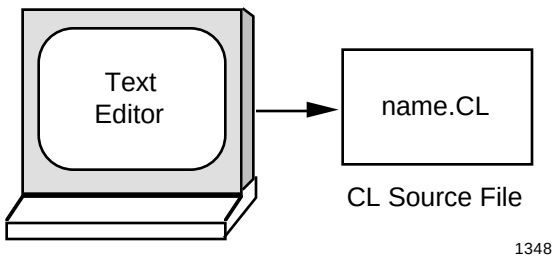


2981

   If necessary, correct the pathnames and press ENTER. The display reappears with all pathnames in blue.

   Note that in the removable media pathname examples above, &ASY is in drive 2 and the other four volumes are in drive 1. To minimize disk/diskette exchanges, the source/object volume, the parameter-list volume, and the custom GDF volume can be the same volume (the same disk/diskette), as they are in the examples. You need the overlay disk/diskette, &OV2, to be in drive 1 when you enter CL from the COMMAND PROCESSOR Display, and then you can put your user volume (CL in the examples) in drive 1 for the remainder of the session.

2. Use the Text Editor (see subsection 5.2) to create the CL source file for the CDS on the CL Source/Object volume named in step 1. You can include other structures, including CL blocks, if you wish.



1348

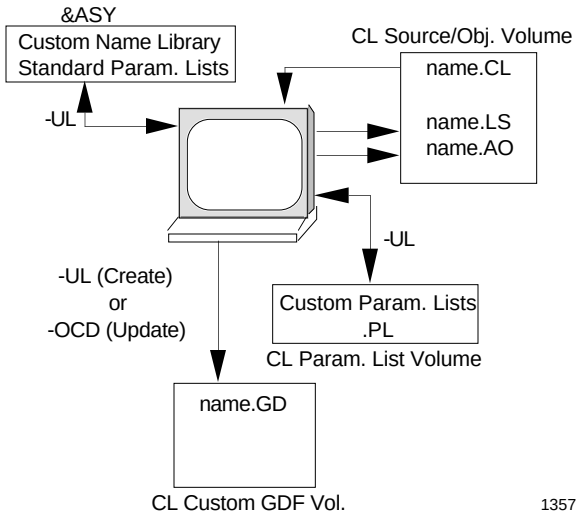
3. Compile (but see the two notes below) the CL source file (see subsection 5.3.2). If no errors are detected, these files are created on the user volume(s):

   `(filename).LS`—the listing file

   `(filename).GD`—the custom GDF

   `(param. list name).PL`—the parameter-list file, one for each parameter list

   Custom names contained in the CDS are added to the Custom Name Library in volume &ASY (if the -UL option was used).
   See Table 3-1.



Note 1: Because there is a limit to the number of each type of custom name, and because they cannot be deleted, you should take care not to cause extra entries because of misspelled names. To avoid this, we recommend that you compile twice, as follows:

First compile without the -UL option. Instead of generating the files listed above, the Compiler generates a .LE file that lists all custom parameter names not now on &ASY.

Use a Print command (see subsection 5.1) to see the contents of the .LE file. Check if there are misspellings of names that should already be in the library, and correct them as necessary.

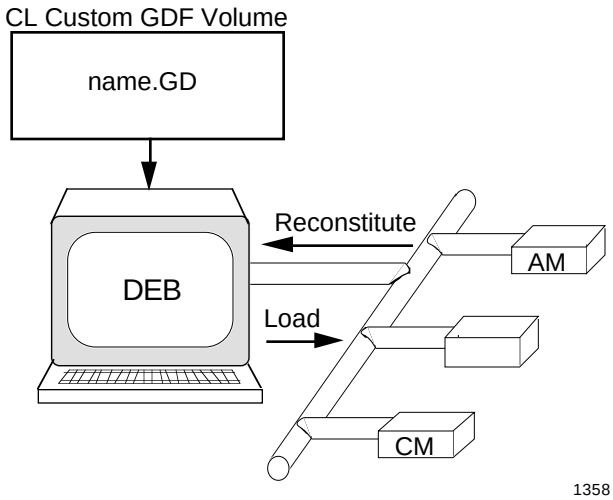
Compile again, using the -UL option. This adds new names to the Custom Name Library.

Note 2: If you are changing a custom data segment, you must use the Compile command -OCD option to overwrite the custom GDFs. See subsection 5.3.1.2.

Note 3: The error message "Cannot create .GD file, FMS error: INSUFFICIENT MEDIA STORAGE SPACE" can be caused by too many memory resident schematics. The reason is that the compiler creates the .GD file in the memory volume ($MEMORY). This is also where memory resident schematics are stored. If there are too many schematics loaded in the US, the compiler can run out of space and be unable to create the .GD file.

4. Use the DEB to add the CDS to the appropriate data point (see the *Data Entity Builder Manual* in the *Implementation /Engineering Operation - 1* binder).

   This may be the initial building of the point or it may already be built. If the point is already loaded in its node, reconstitute it, or read it from its IDF. Key in data for the following parameters:



   PTDISC—This parameter applies to AM regulatory points. It is on the AM Point Assignment Display. Select Full and press enter. This adds the Custom I/O Display to the PED set. The following parameters are on that PED.

   NOPKG—The maximum number of CL packages you expect to be bound to this point. Key in at least 1. You can key in more than you expect to use to avoid having to make a change in the future. Maximum is 10.

   PKGNAME—Key in the name of your CL source file(s). If you entered more in NOPKG than actual packages, after you press ENTER you will have some blank ports, which are acceptable.

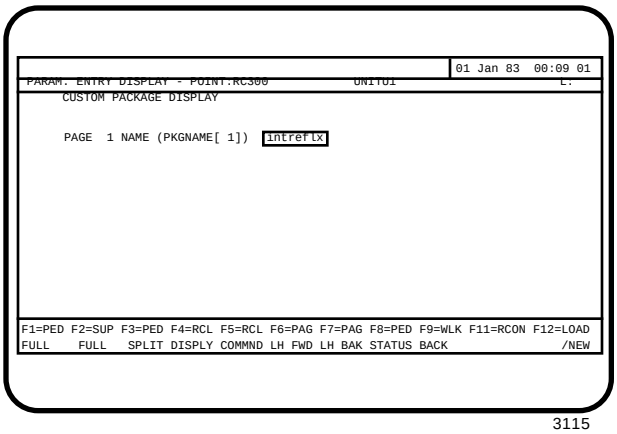
5. Press ENTER. If no errors are detected by the DEB, the PED is redisplayed with all values in blue. This also adds your custom parameters to the PED set (the CL Custom GDF volume must be available on an HM or in a disk/diskette drive).

WARNING

Sometime in the future, you may need a backup copy of your custom-name library files because, for some reason, you are unable to use the data on an HM to restart a node; therefore, when you install a new CDS or update one, you should copy the resulting custom-name files onto your current &ASY disk/diskette. Execute-Command files ASY_BKUP.EC and CLNCFBKUP.EC are provided on the Honeywell &ASY disk/diskette to help maintain these files (see your latest *Customer Release Guide*). Immediately after copying these files, use the Command Processor's Protect command to protect them from being accidently overwritten or deleted (of course, you will need to unprotect them the next time you make such a backup copy).

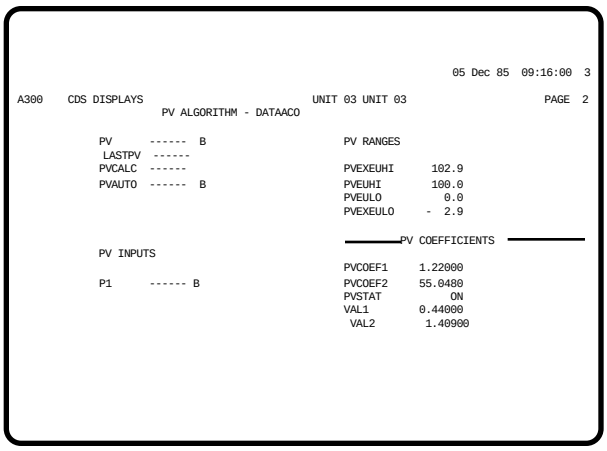
You should also make a backup copy of your custom GDF (.GD) files on a disk/diskette, so that if those files become unavailable from the HM you still have GDFs to use in rebuilding or reloading points with CDSs. These backup files should also be protected.

6. Key in values for your custom parameters, as required, key in data and make selections for other parameters, as necessary, and ENTER your data. When the PED status line shows no errors, no unENTERED data, and no missing data and after checking the checklist under subsection 2.5, write it in an IDF and load the point with its new CDS into its node.



3115

You can see the data for your CDS on the CDS page(s) of the Detail Display for the point, at a US with the Operator Personality.



3116

3.5 CHANGING CUSTOM DATA SEGMENTS

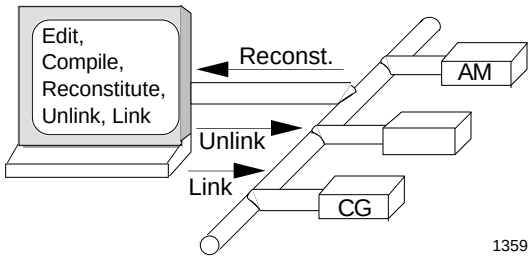
3.5.1 Change a CDS Used on a Single Data Point

Use the following major steps to change a CDS that is used on only one data point. The point can be in an AM or a CG.

1. Check volume pathnames as in subsection 3.4, step 1.

2. Use the Text Editor to change the CDS source file (see subsection 5.2).

   Note: You can edit the original file, retaining its name and the CDS-description name, or you can copy the original file, rename it, and edit the new file. Where the CDS is used on just one point, you can save some work by editing and recompiling the original file, which overwrites the original custom GDFs and retains the file name.



1359

3. Compile the revised source file as in step 3 under subsection 3.4.

4. Use the DEB to reconstitute the data point, key in and ENTER data, and reload the point, as in steps 4 through 6 under subsection 3.4. If you used the original file name, any parameters in the new version of the CDS that were also in the original CDS don't have to be reENTERED. When the point is reloaded, the original CDS in the point is deleted and the new one takes its place.

5. See the WARNING at the end of subsection 3.4.

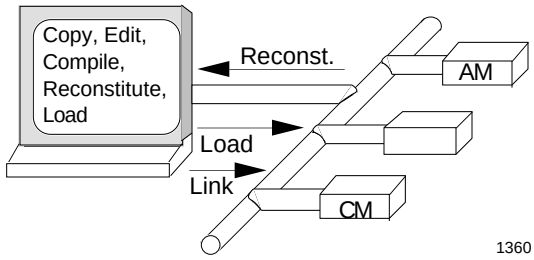
3.5.2 Change a CDS Used on More Than One Data Point

Use the following major steps to change a CDS that is used on more than one data point. The points can be in an AM or a CM60.

1. Check volume pathnames as in subsection 3.4, step 1.

2. Use the Text Editor to copy the original source file, give the copy a new name, and make the changes to the new file (see subsection 5.2).

   Note: As with generic CL blocks, this lets you retain the original CDS description, which several points may be using, while testing the new one. This is especially important because the custom GDF (.GD) must be available if it is necessary to use the DEB to reconstitute, change, and reload any of these points.



   We recommend that you use the last character in your CDS source-file name as a revision indicator. Keep the number of revisions as low as possible; if possible, no more than two or three, because there is a limit of 1000 CDS-description names on the &ASY volume and unused names can't be deleted without rebuilding all CL structures, including all related data points.

3. Compile your new CDS source file as in step 3 under subsection 3.4. Because you used a new name, you have both the original compiler-generated files and a new set of these files with the new file name. Retain the original files, in case you need to use them again, because of problems with your new CDS.

4. Reconstitute the first point to receive the new CDS, as in step 4 under subsection 3.4. In parameter PKGNAME, be sure to key in the new source-file name, enter parameter values (it's a new CDS, so key data in all parameters that appear), ENTER it, and reload the point, as in steps 5 and 6 under subsection 3.4.

5. The first point will be operating with a completely new CDS. After you are satisfied with the operation of the new CDS on the first point, repeat step 3 for other points that will use the new CDS.

6. When the CDS is operating on all points, use the Delete command (see 5.1) to delete the original .GD file from the CL Custom GDF volume.

7. See the WARNING at the end of subsection 3.4.

3.5.3 The "REV NO MISMATCH" Error

The name stands for Revision Number Mismatch. This is an error that can occur when loading a CDS to a point from the PED. The error occurs if one or more other points in the node have a different version of the CDS (same name but a different time stamp). The error message appears at the bottom of the Param. Entry Display screen.

The first time you compile a CDS with the Update Library (-UL) option, a time-stamped Global Descriptor (.GD) file is created in the directory specified under CL CUSTOM GDF on the Volume Paths display (normally the CDSG directory). The .GD file contains the names of the parameters in the CDS and the information that will be required by the PED to create the extra pages for entry of these parameters. When you use the PED to attach the CDS to a point, the system uses the .GD file to determine this format information.

To give an example, suppose that you have a CDS attached to several points and that you decide to add another parameter to the CDS. You make the change and recompile the CDS source without changing the name, using the Overwrite Custom Data (-OCD) option to overwrite the information in the .GD file (this adds the name of the new parameter to the .GD file. Compiling with the -OCD option creates a new time stamp on the .GD file. This in itself does not create a problem, but when you use the PED to attempt to load the modified CDS to a point, the system will check to see if there are any points in the node that have a different copy (different time stamp) of the CDS. If any are found, you will get the REV NO MISMATCH error in the PED when you try to do the load, and the load will not be performed. The system will not allow you to load a point with a CDS if another point or points in that node have a CDS of the same name with a different time stamp (even if the parameters are exactly the same).

To recover, change the name of the new CDS and recompile (with the -UL option). Then use the Find Names command (from the Command Processor) or Documentation Tool (from the Engineering Main Menu) to determine which points in the node have the CDS attached. Then replace the old version with the new (renamed) version one point at a time, as described in subsection 3.5.2.

You may not wish to rename the CDS because you are limited to 2000 CDS names in the Custom Names Library. If you do not rename the new CDS, you must delete the old CDS from **each** point in the node **before** you attempt to load the new version to **any** point. For each point with the old CDS attached:

- Make the point INACTIVE
- Reconstitute the point in the PED
- Delete the old CDS slot
- Load the point

After the old CDS is removed from **each** point to which it was attached, use the following procedure to attach the new CDS to each point:

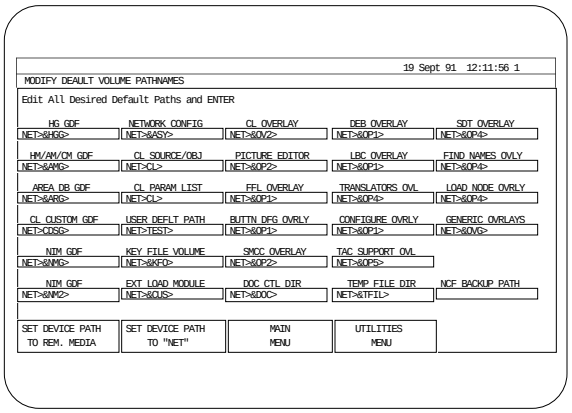
- Reconstitute the point in the PED
- Add a CDS slot
- Attach the new CDS
- Reload the point
- Make the point ACTIVE

3.6 INSTALL A CUSTOM ENUMERATION OR PARAMETER LIST

3.6.1 Install a New Custom Enumeration or Parameter List

Use the following major steps to build and compile a new custom enumeration or a new parameter list.

1. From the Engineering Main Menu, select SUPPORT UTILITIES and then select MODIFY VOLUME PATHS. On the Modify Default Volume Pathnames Display check that the pathnames for the following volumes are correct:



2981

   NETWORK CONFIG
   examples; $F2>ASY> or NET>&ASY>

   CL SOURCE/OBJ
   examples; $F1>CL> or NET>CL>

   CL OVERLAY
   examples; $F1>&OV1> or NET>&OV2>

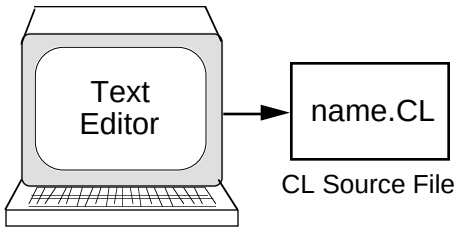
   CL PARAM LIST
   examples; $F1>CL> or NET>CL>
   (not required for enumerations)

   If necessary, correct the pathnames and press ENTER. The display reappears with all pathnames in blue.

   Note that in the removable media pathname examples above, &ASY is in drive 2 and the other three volumes are in drive 1. To minimize disk/diskette exchanges, the source/object volume and the CL parameter-list volume can be the same volume (the same disk/diskette), as they are in the examples.

   You need the overlay disk/diskette &OV2, to be in drive 1 when you enter CL from the COMMAND PROCESSOR display, and then you can put your user volume (CL in the examples) in drive 1 for the remainder of the session.

2. Use the Text Editor (see subsection 5.2) to create the source file for the enumeration or parameter list.



3. Compile the CL source file. If no errors are detected, the following files are created:

   For custom enumerations,

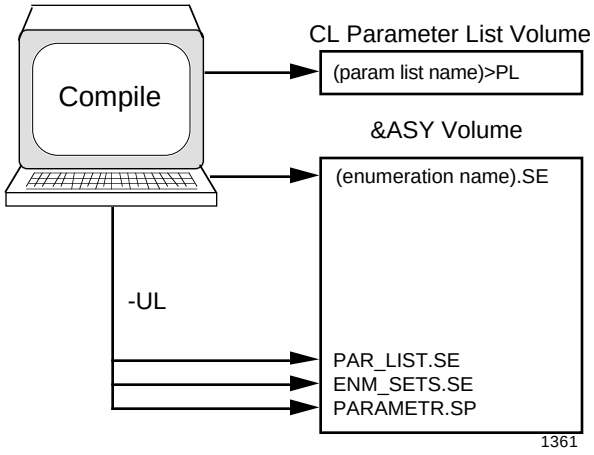
   (enumeration name).SE

   is created on the &ASY volume. It lists the state names for all of the enumerations (in ASCII).

   For custom-parameter lists,

   (param list name).PL

   is created on the CL parameter-list volume. It lists all parameters and their types, for this parameter list.



   If the -UL option was used, the following custom-library files are updated:

PAR_LIST.SE	Contains the names of all custom-parameter lists.
ENM_SETS.SE	Contains names of all custom enumerations.
PARAMETER.SP	Contains names of all custom parameters.

   Note that once a custom enumeration is added to the system through a successful compilation with the -UL option, it can be neither modified nor deleted.

4. See the WARNING at the end of subsection 3.4.

3.6.2 Change a Parameter List

Use the following major steps to change a custom-parameter list.

1. Check the volume pathnames as in subsection 3.6.1, step 1.

2. Use the Text Editor (see subsection 5.2) to modify the parameter-list source file.

   Note: The original file can be edited. If the parameter list is in a package with Custom Data Segments, also pay close attention to the guidelines provided under subsections 3.5.1 and 3.5.2.

3. Compile the source file as in subsection 3.6.1, step 3. If the compilation is successful, the original parameter list is overwritten with the new definition.

4. See the WARNING at the end of subsection 3.4.

3.7 INSTALLING AND CHANGING COMBINED CL/AM PACKAGES

CL packages that consist of combinations of CL structures, including blocks, CDSs, enumeration definitions, and parameter-list definitions, can be installed by combining the steps and the topics covered by notes under subsections 3.3 and 3.4.

When you recompile generic packages that contain a custom data segment (CDS) and you use the -OCD option, the CDS is overwritten. If you have not made any changes to the CDS, recompile the package WITHOUT the -OCD option.

3.8 CUSTOM NAME LIBRARY

Table 3-1 describes the custom-name library. This library includes the internal-to-external conversion information for the names it contains. It is maintained on all Universal Stations on the LCN and on the &ASY volume.

The Compiler's -UL option enables updates to this library.

Internal names are used in the CL object code (.AO file), in the custom GDF (.GD), and in AM and CG/PLNM checkpoints that contain custom data.

It is not possible to delete or alter the names in this library.

You can use the Print command (see subsection 5.1) to see the content of the files in this library.

Parameter list names cannot conflict with enumeration-set names. An enumeration state can be in any number of enumeration sets and each counts against the 1000-name limit.

Standard parameter names can be used in CDSs, if the point the CDS is attached to doesn't already contain the standard parameter. For example, parameter PV can be used as a CDS parameter on an AM-custom data point but not on an AM-regulatory data point. Standard parameter names used in custom data segments don't count against the size limitations in Table 3-1.

Table 3-1 — The Custom Name Library

Type	Max. Names	File Name
Custom Parameter Names	2000	PARAMETR.SP
CDS Description Names*	2000	SEGMENTS.SP
Parameter List Names	250	PAR_LIST.SE
Enumeration Set Names	200	ENM_SETS.SE
Enumeration State Names	1000	(name).SE
New standard/existing custom parameter name matches	N/A	Duplicat.SP
*Names of CL-source files that contain custom-data descriptions		

3.9 MOVING CL APPLICATIONS BETWEEN SYSTEMS

CL structures can be moved from one LCN-based TDC 3000^X System to another in only CL source-file form. Object files, IDFs, and GDFs contain internal representations of names that apply to only the system they were created on. Specifically,

- **Custom-data GDFs** must be regenerated on each destination system by compiling the appropriate CL-source files. This installs any custom names in the destination systems' Custom Names Libraries.
- **Data points** in the destination system must be built using the **DEB's Exception Build** command.
- **CL blocks** must be recompiled on the destination system to create loadable object (.AO) files for that system.

It's likely that before you do any of these activities you will need to edit the CL-source files on the destination system, to change point names and to tailor the source file for the destination system.

3.10 CUSTOM NAME LIBRARY SYNCHRONIZATION

The custom-name library is maintained on all Universal Stations and on the Network Configuration Volume, &ASY. A revision code and time stamp are maintained on all copies of the library to assure synchronization.

When a US is started up, its custom-name library is loaded from the &ASY volume. The revision code of the custom-name library on &ASY *must* match that of all other USs on the LCN; otherwise, the US won't start up (its status will be (NameRev). See below for custom name library reconstruction steps.

If during CL compilation, names are added to the custom-name library (the -UL option is used) the revision code of the &ASY volume selected on the Modify Default Volume Pathnames Display must match the revision code in the USs. Otherwise, the compilation will fail.

We recommend the following:

- If you keep the &ASY volume on an HM, that version should always be used and removable media should not be used (the default pathname should be "NET>&ASY>"). This keeps the HM version of &ASY synchronized with the USs, so that it can be used in starting up USs.

- If &ASY is not maintained on an HM, multiple users of CL must be sure that they all use the same &ASY disk/diskette. You must take particular care for systems with more than one Console (probably, most such systems have &ASY on an HM).

- If the &ASY volume that is synchronized with the USs should be lost or destroyed, the current custom-name library can be reconstructed on an &ASY volume (removable media or HM) from the information in the USs, using the Custom Name Save function. Refer to the *Process Operations Manual*, Section 20.

- If all USs are lost and you can't reconstruct the custom-name library, the latest &ASY must be used, but it is extremely important to immediately rerun all CL compilations that have added custom names since the time of the old &ASY. Anything that uses those names, such as data points, CL structures, and custom displays, should be re-established. Otherwise, old and probably invalid internal names would be used with unpredictable results.

WARNING

If the status of the US is CNAMREV, indicating that the custom-name library for this US is at a different revision level than those of the other LCN nodes, the Custom Names Save function **WILL NOT take place**. A WARNING status appears on the US node status display and an associated message appears in that US's Status Detail display.

ENGINEER'S KEYBOARD
Section 4

Use this section when you don't know or are not sure of the effect of pressing a key on the Engineer's Keyboard.

4.1 CL DATA ENTRY KEY FUNCTIONS

The Engineer's Keyboard is required for interaction with the Engineering Personality's Utilities, Text Editor, and CL Programs activities. Most of the keys function as you would expect. Figure 4-1 clarifies the functions of several of the keys.

Several keys have dual functions and dual markings—symbols on the top of the key and other symbols on the front edge of the key. The secondary function, indicated on the front edge of these keys, is obtained when one of the two CTL keys is held down while the function key is pressed.

The generic-function keys, F1 through F12, are used to execute Text Editor commands, as described in Section 3 of the *Text Editor Operation* manual. These keys are not used in the Utilities and CL Programs activities.

The color and behavior keys (BKGND, BLINK, INTEN, WHT, BLK, CYAN, BLUE, MAGN, RED, YEL, GRN) don't operate with the Utilities, Text Editor, and CL Programs activities.

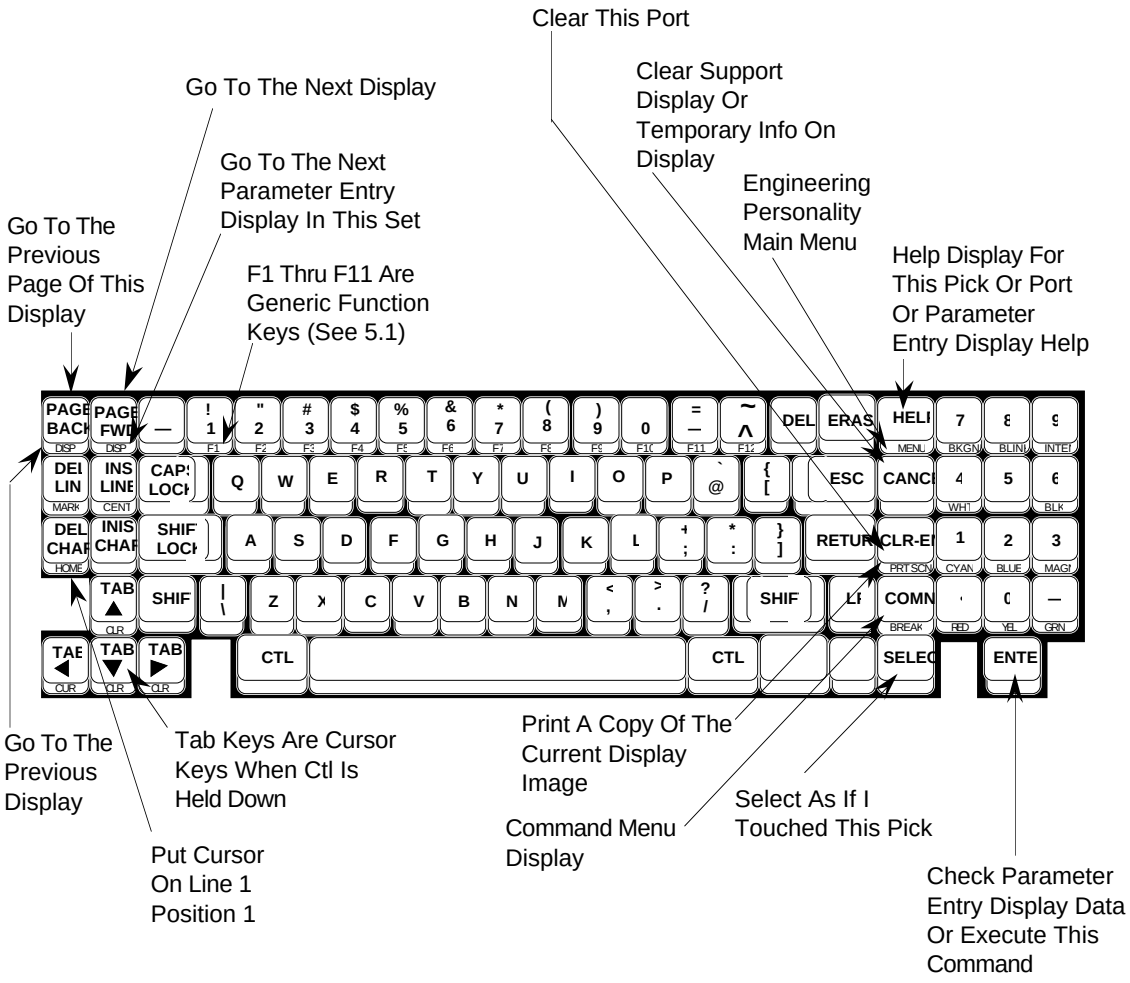


Figure 4-1 — Engineer's Keyboard Special Key Functions

1196

COMMANDS
Section 5

This section provides

- *An overview of Text Editor and Utilities commands that are related to CL-data entry (see subsections 5.1 and 5.2).*
- *Detailed descriptions of the CL Compile and Link commands (see subsection 5.3).*

Section 3 describes how these commands are used to install and to modify CL programs. Section 6 provides guidelines for recovering from errors indicated as the Compile and Link commands are executed.

5.1 USING UTILITY COMMANDS IN CL DATA ENTRY

The default-volume pathnames and utilities are used in CL data entry. The Modify Default Volume Pathnames display is called up by selecting the `SUPPORT UTILITIES` pick on the Engineering Main Menu and then selecting `MODIFY VOLUME PATHS` on the Support Utility Menu. (Alternately, you can select the `COMMAND PROCESSOR` pick on the Engineering Main Menu, and then type SP and press ENTER.) The utilities are called up by selecting the `COMMAND PROCESSOR` pick on the Engineering Main Menu.

The following are the functions that you will probably use most in CL data entry (command-file utilities Execute Command, Prompt Out, Pause, and End are described under subsection 5.4):

- **Modify Default Volume Pathnames**—This function is used to set up the pathnames to the files that are created and used in CL data entry. Examples of such pathnames are provided in this publication under the following headings:

  3.2.1—Compile and Link to a Bound Data Point

  3.2.2—Compile and Link Generic CL/AM Blocks

  3.4—Install a New Custom Data Segment

  Instructions for calling up the Modify Volume Pathnames Display are under subsection 4.3 in the *Command Processor Operation* manual. Instructions for changing the default pathnames are in Section 6 of that publication.

- **Copy**—Used to copy a file or files from one volume to another, or on the same volume. We recommend that you copy all of your files to removable media, in case a file is lost or damaged. The Copy and Copy Volume commands are described under subsections 5.6 and 5.7 in the *Command Processor Operation* manual.

- **Create**—Used to name a volume on a floppy diskette or cartridge disk and to initialize the disk so that it can be used on the TDC 3000^X System. This command is used to create user volumes that you use in CL data entry or on which you make backup copies of HM files. This command is described under subsection 5.8 in the *Command Processor Operation* manual.

- **Create Directory**—Used to create a directory under an existing volume. An example is described in subsection 3.9.2 in this publication, Prepare to Load the MO Object File.

- **List Names**—Used to list the attributes of a volume and its files. You can use this command to determine if specific files are in a volume, the time and date they were created, their size, and other information. For your records, you might want to use the Data Out command to direct the list to a printer as well as to the screen. The List Names command is described under subsection 5.13 in the *Command Processor Operation* manual.

- **Data Out**—Directs the output generated by a subsequent Print Command to a printer or file as well as to the display screen, or to the screen only. This command is executed directly from the Command Processor. To direct the output to go to the screen and a printer, on the Control Language Compiler/Linker display, key in a command as in this example:

  ```
  DO $Pn
  ```

  and press ENTER. In this command, n is the printer number (which is available on the Console Status Display). To direct that the output is to go only to the screen (this is the default situation), key in DO and press ENTER.

- **Print**—Used to display or to print the content of a file. Use this command to look at your listing (.LS) and error (.LE) files. This command is executed directly from the Command Processor. Key in a command as in this example:

  ```
  PR NET>CL>VLR_CALC.LE
  ```

  and press ENTER. When the whole content of the requested file has been displayed, "Print Complete" appears at the bottom of the display area. If a Data Out command has also directed the output to a printer, the content of the file is simultaneously displayed and printed.

- **Delete**—Deletes a file from a volume. If you make changes to CL source files, you may want to copy the original file to a file with a slightly different name, make the changes in the new file, install those changes, test them, and when the modified structure is operating properly, delete the original file. For example, you might name your original file CALC1 and the new file CALC2. When CALC2 is compiled, installed, and successfully running, you could delete CALC1. The delete command is described under subsection 5.10 in the *Command Processor Operation* manual.

- **Break**—Aborts an in-process CL compile or CL link (but not a CL unlink). The break is initiated by pressing the combination of the “CTL” and “BREAK” keys. (“BREAK” is an alternate function of the “COMND” key.)

5.2 USING TEXT EDITOR COMMANDS IN CL DATA ENTRY

The following procedures describe the Text Editor functions you will probably use most frequently in CL data entry. For more detailed descriptions of Text Editor functions, refer to *Text Editor Operation* in the *Implementation/Engineering Operations - 1* binder.

5.2.1 How to Create a CL Source (.CL) File

1. You access the Text Editor through the COMMAND PROCESSOR pick on the Engineering Main Menu display.

2. Key in ED followed by the pathname for the file to be created. Be sure to include the .CL suffix. For example,

   ```
   ED NET>CL>CALC1.CL
   ```

   The file name cannot begin with "$."

3. Press ENTER. A blank Text Editor display appears, ready for you to key in your source-file information.

4. Key in your source file. The form and content of source files for all CL structures are in the *Control Language/Application Module Reference Manual*.

5. After you have keyed in and visually checked your entire source file, end the Text Editor session by pressing LF, then E, and RETURN (or CTL F1 followed by CTL F2). The source file is transferred from the Text Editor's temporary file to the user volume that you named in step 2, and your source (.CL) file is now ready to compile.

5.2.2 How to Copy a File, Edit it, and Store it in a New File

Often, you will use several similar CL structures, and you can save time by copying a source file, editing the copy, and storing the result in a new file.

1. You access the Text Editor through the COMMAND PROCESSOR pick on the Engineering Main Menu display.

2. Key in ED followed by the pathname for the file to be copied. Be sure to include the .CL suffix. For example:

   ```
   ED NET>CL>CALC1.CL
   ```

3. Press ENTER. The first 23 lines of the file are displayed.

4. Edit the copy of the file that is now on the Text Editor display to make the changes that are needed in the new file. When you finish, leave the cursor on the first character you want in the new file.

5. Hold CTL and press F7. Four new prompters appear (`F1=Define`, `F2=Get`, `F3=Put`, and `F4=UnProt`).

6. Hold CTL and press F1. A prompter appears that asks for the pathname for the new file.

7. Key in the pathname of the new file and press ENTER. Here is an example of the pathname:

   ```
   CALC2.CL
   ```

8. Hold CTL and press F3. A prompter appears that asks how many lines to output. If you want to store the whole file, key in 9999 (this is more lines than the largest file). Otherwise, key in the number of lines you want to store in the new file.

9. Press ENTER. You now have two copies of the file, the original one that you copied and the new one that you named in step 7. In this example, they are both on the same volume. If a different volume is needed, precede the filename in step 7 with the entire pathname.

5.3 CL COMPILER/LINKER COMMANDS

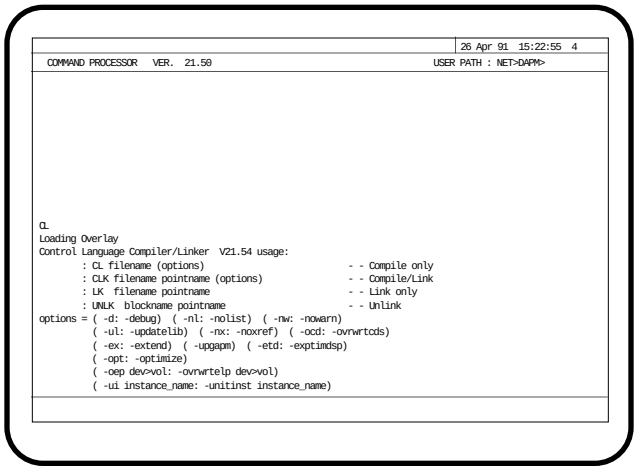
The CL Compiler and Linker commands are available through the `COMMAND PROCESSOR` pick on the Engineering Main Menu. The Compiler reads CL source (.CL) files and creates object files from them. The object files are then linked to a data point and loaded into the module or gateway (node) in which the point resides.

Object files are files that contain the CL structures in the form that is executed by the destination node.

The Compiler also establishes Custom Data Segments and user-defined enumerations and parameter lists.

5.3.1 Compiler/Linker Command Descriptions

This Control Language Compiler/Linker Display appears when you type `CL` on the COMMAND PROCESSOR display and press ENTER. When the display first appears, the cursor is in the command-entry/prompter region, as shown here. The area between the command-entry/prompter region and the title line displays the results of command execution.



2989

5.3.1.1 Compiler/Linker Commands

The following commands can be executed from this display. Execution is requested by keying in the command line with any options and pressing ENTER.

- **Compile**—Compiles CL/AM source files, creating object and listing files, or if errors are detected, an error file. Form:

  ```
  CL (filename) -(options)
  ```

  For Custom Data Segments, the Compiler also creates custom GDFs. For custom enumerations, (enumeration name).SE is created (see 3.6). For custom-parameter lists (param list name).PL is created (see 3.6).

- **Link**—Links CL/AM object files (.AO) to the data point(s) and loads the program in the AM. Form:

  ```
  LK (filename) (tag name)
  ```

- **Unlink**—Removes the linkage from a CL/AM object file to a data point so that, if required, a revised version of the program (object file) can be linked again to the point. Form:

  ```
  UNLK (block name) (tag name)
  ```

- **Compile and Link**—Combines the Compile and Link command functions in one command line. Form:

  ```
  CLK (filename) (tag name) (options)
  ```

In the command forms above, "(filename)" is the name of the CL source file. The source file must have a .CL suffix, but use of the suffix in the commands is optional. "(block name)" is the name of the CL block to be unlinked. A source file can contain more than one block, so block names are not necessarily the same as the source-file name, though they may be. "(tag name)" is the name of the bound data point that is configured in the point's NAME parameter. Letters on the command line can be uppercase or lowercase or a mixture of upper- and lowercase. (filename) cannot begin with "$."

5.3.1.2 Compile Command Options

The following options are available for the Compile command. The options are requested by keying in "-" followed by the option name for each option. More than one option can be included, and each is separated by a space. For example, a command line that requests every CL/AM option would have this form:

```
CLK CALC2 -D -UL -NW -NL -NX -OCD -ETD
```

Following is the full set of CL compile options (note that some do not apply to CL/AM):

- **Debug**—Compile %DEBUG lines in the source file. See subsection 3.3.3 in the *Control Language/Process Manager Reference Manual.* Form:

  `-D (or -DEBUG)`

- **Update Library**—Update the AM library files (see subsection 3.2.1). Form:

  `-UL (or -UPDATELIB)`

- **No Warnings**—Do not list notes and warnings in .LE or .LS files. Form:

  `-NW (or -NOWARN)`

- **No Listing**—Do not generate a full listing file (.LS); list only error lines. Form:

  `-NL (or -NOLIST)`

  The -NL option automatically causes the -NX option to be selected; therefore, the CL/APM cross reference is omitted from the listing and both -NL and -NX are displayed as selected options at the bottom of the listing.

- **No Cross Reference**—Omit printing the CL cross reference (symbol table) at the end of the listing. Form:

  `-NX (or -NOXREF)`

s

- **Overwrite Custom Data**—Overwrite existing custom GDFs. Required to change the .GD file if it already exists. Form:

  `-OCD (or -OVRWRTCDS)`

- **Expanded Time Display**—Time values will be formatted in the expanded time format allowing the display of time values greater than one day. When this switch is not used, time values are formatted as HH:MM:SS. When this switch is used, time value format depends on the time value. If the value is less than one day, it will be formatted as HH:MM:SS. If the value is one day or greater, but less than 1000 days, it is formatted as DD HH:MM:SS. If the value is greater than 1000 days, it is formatted as DDMMMYY HH:MM:SS. Form:

  `-ETD (or -EXPTIMDSP)`

NOTE

To terminate a compilation in progress, press the combination of the "CTL" and "BREAK" keys on the engineering keyboard. The CL Compiler recognizes the Break key during any of its three passes or during the listing generation. The CL Compiler will also let you break out of a CL link (CLK or LK <filename><pointname>). However, it will not let you break out of a CL unlink (UNLK <blockname><pointname>).

5.3.1.3 Results of Custom GDF (.GD) Compilation

A custom GDF (.GD) can be created with or without specifying the -OCD option in the command line, but to change (overwrite) an existing .GD file you must use the -OCD option.

The results of a successful compilation are reported both on the screen and in the listing (.LS) file (no files are modified on an unsuccessful compilation). In addition to reporting the status of custom data descriptors, the Compiler also reports the status of object files and of custom parameter list files.

The following are examples of the messages that report compilation status:

For a compilation that creates a .GD file for the first time;

```
File OCDST1.GD created
```

For a compilation where the .GD file exists and the user has not specified the -OCD option;

```
File OCDST2.GD not overwritten, per command
```

For a compilation where the .GD file exists and the user has specified the -OCD option;

```
File OCDST3.GD overwritten
```

For a compilation that results in a new AM object file (.AO) and parameter list file (.PL);

```
File OCTST5.AO created
File OCDST2.PL created
```

For a compilation that overwrites existing .AO and .PL files;

```
File OCDST1.PL overwritten
File OCDST4.AO overwritten
```

5.3.2 How to Invoke Compiler/Linker Commands

5.3.2.1 Compilation Precaution

Node Resources—A typical compilation, for example, compiling a package with a CDS, a parameter list, and one or more CL blocks can cause up-to-10 files to be open at once, which can cause problems with the resources in USs or HMs in a busy system. The symptoms are file-management errors; or messages saying, "`LRNs exhausted`," or "`Memory exhausted`," when a file is opened or created. To avoid such problems, verify that the USs and HMs involved will not be performing other filing functions while the compilation is taking place.

5.3.2.2 Volume Handling and Pathnames

The Command Processor display doesn't provide prompters to request that required volumes be mounted in a disk/diskette drive for a CL compilation. If you don't make a needed volume available, an error message appears, indicating that the required volume or file could not be found. After you mount the required volume, retype your command line and press ENTER.

The pathnames used in executing Compiler and Linker commands are the default pathnames on the Utilities' Modify Default Volume Pathnames display.

The descriptions of typical CL data entry sessions in Section 3 provide examples of volume pathnames, and suggestions for handling disk/diskettes. Table 2-1 lists all volumes involved in CL data entry.

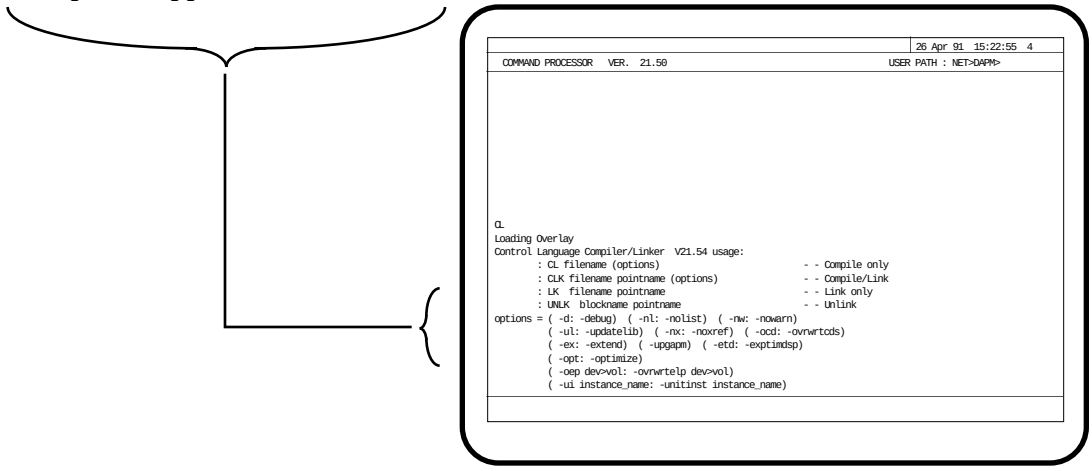
5.3.2.3 Invoking a Compile Command

1. If you are using some Engineering Personality activity other than the COMMAND PROCESSOR, hold CTL and press MENU to return to the Engineering Main Menu.

2. To invoke CL:

   Select the COMMAND PROCESSOR, type `CL` and press ENTER (Engineering Personality Overlay volume &OV2 must be available on a disk/diskette or an HM). The Control Language Compiler/Linker display appears.

3. The summary of CL commands and options appear as shown here.



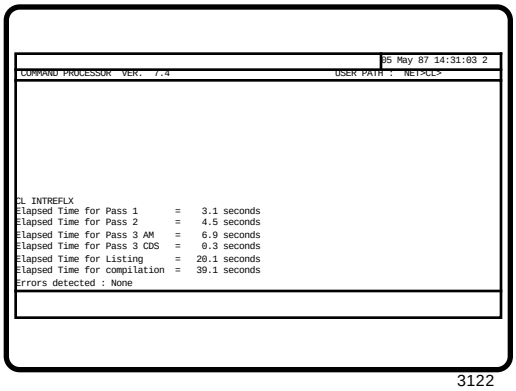
2989

4. Check that your CL source file is available on the CL Source/Obj volume and that other volumes are available, as needed, for the type of structure you are compiling. To determine which are needed, refer to major steps and the sketches in Section 3.

   Key in a Compile command line, and any options needed (see subsection 5.3.1.2). Here we keyed in "`cl intreflx`." "Intreflx" is the name of our source file.

5. Press ENTER. The Compiler makes several passes through the source file and each of these is noted on the display.

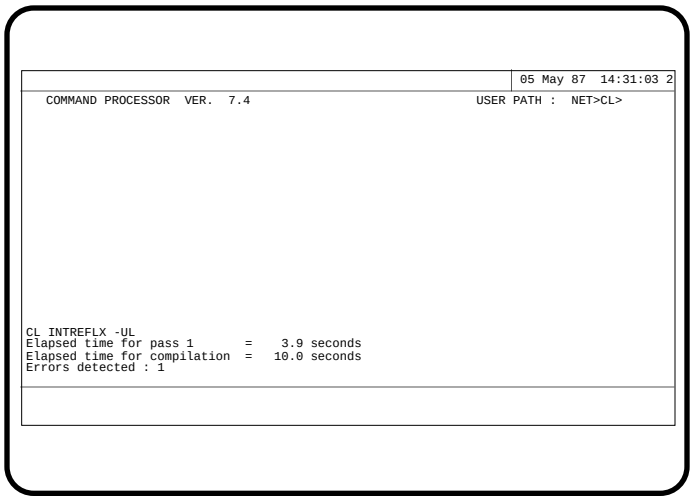
   As each pass is processed, the line number of the current line being compiled is displayed on the Command Line to show progress of the compilation.



3122

   Comment lines in the CL program are skipped and their line numbers are not displayed. During compiler processing of internal executive routines, a zero (0) line number is displayed. Here the compiler made four passes and generated the listing in 39.1 seconds and it detected no errors. The listing file (.LS) and the object file (.AO) are now available on the CL Source/Obj volume.

If errors are detected, they are indicated on a line like the one at the bottom of this display, and the listing and object files are not created; instead, an error file (.LE) is created on the CL Source/Obj volume. Section 6 provides guidelines for recovering from errors.



3120

5.3.2.4 Invoking a Link Command

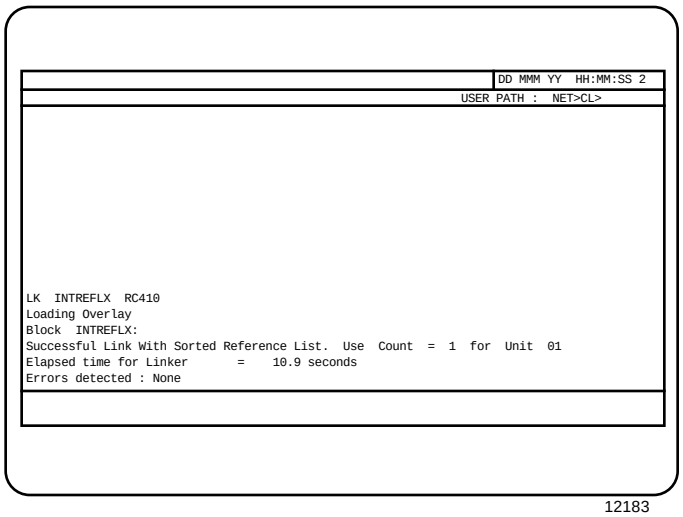
1. Check that the object file (.AO) is available on the CL Source/Obj volume, then key in a Link-command line. Here, we keyed in "`LK INTREFLX RC410`." "INTREFLX" is our source-file name and RC410 is the tagname for the point INTREFLX is to be linked to.

2. Press ENTER. During the linking process, as errors are detected, error messages appear on the display. Linker error messages and guidelines for recovery are in Section 6. If any error, other than errors that cause notes and warnings is detected, linking cannot be completed.

3. When the program is linked to the point and loaded in the AM, an "Elapsed time for Linker" message is displayed. In the first example below, this message indicates that linking and loading took 10.9 seconds. This message is followed by a message that indicates how many errors were detected. In this case, none were.

4. The CL Compiler generates a list of parameters referenced by the CL program. This list is called the Reference List , and it is included in the object file generated by the compiler. As part of the AM Performance Enhancements available in release 401 and later, the CL Linker sorts the Reference List as it links the CL object to a point in order to maximize AM performance. When the program has been linked to the point and loaded in the AM, one of three successful link messages may appear:

   The first message, "**Successful Link With Sorted Reference List**," indicates that the CL Linker put the generated Reference List in an order that will take advantage of some of the AM Performance Enhancements.



12183

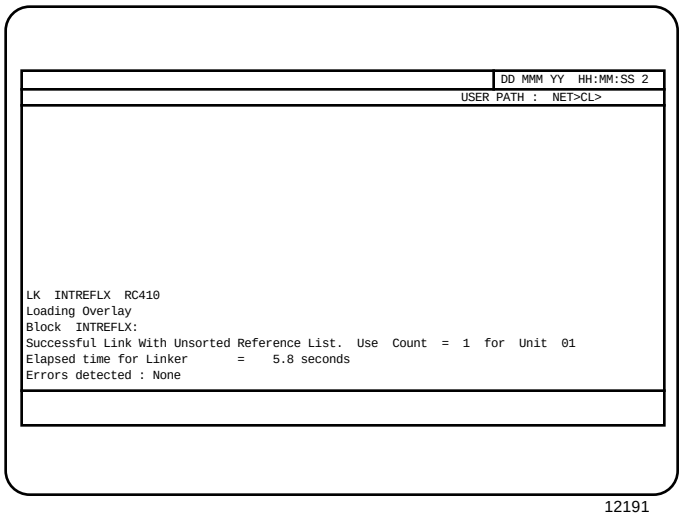
The second message, "**Successful Link With Unsorted Reference List**," indicates that:

(a) the link was to an AM that did not contain the AM Performance Enhancements,

(b) the block is already linked to a point in the same unit in an unsorted manner (the CL Linker cannot sort the Reference List if the CL object was linked to a point or points by an older version of the linker—see Note below), or

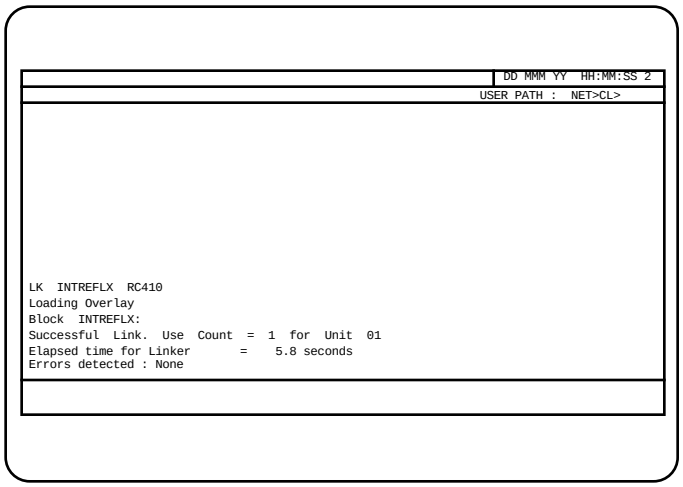
(c) the block does not contain any reference list information.

Regardless of the reason, the block is linked and loaded into the AM in such a manner that it is unable to take advantage of the AM Performance Enhancements.



12191

If the message is displayed as "**Successful Link**," this indicates that the link was with an EP or a UP that did not contain the AM Performance Enhancements.

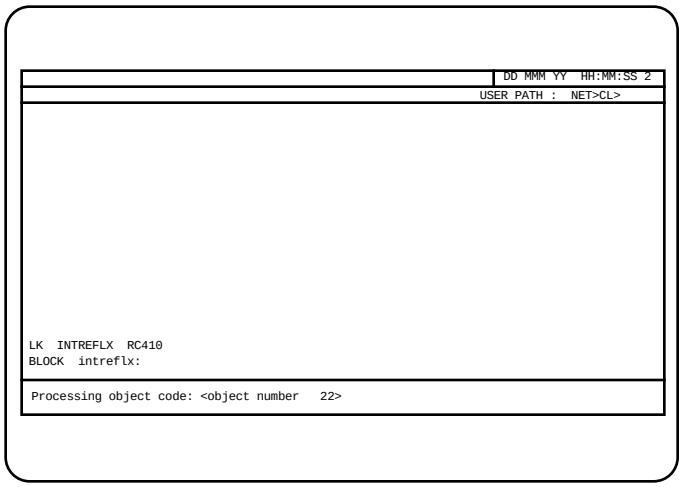


12190

NOTE

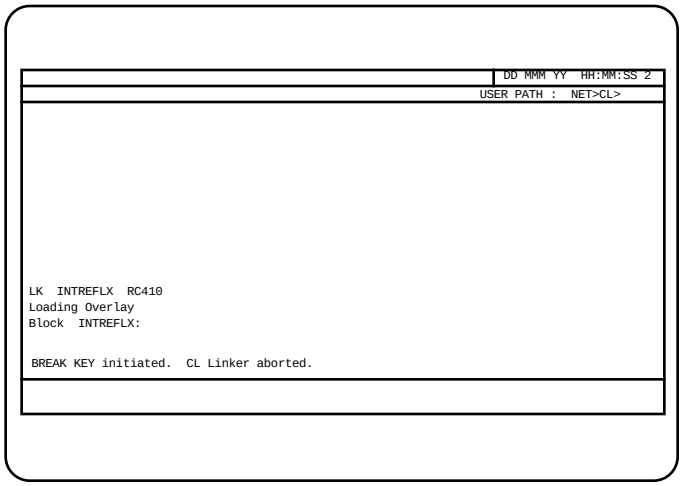
In order to take full advantage of all the AM Performance Enhancements, all CL Blocks with Reference Lists in AMs containing the Performance Improvements should be linked in a sorted manner. If blocks are already linked unsorted, unlink all the blocks in a unit and then relink. You should see the "Successful Link With Sorted Reference List" message on the screen.

5. As each portion of the link is processed, a status line about that portion of the processing is displayed on the Command Line. The intended use of this line is to display the progress of the link. Because of the various ways in which CL programs are structured, some messages may be displayed for a few seconds, while others may appear briefly.



11903

6. If you wish to abort the link, you can use the BREAK function. Hold down the CTL key and press the BREAK key (same as the COMND key).



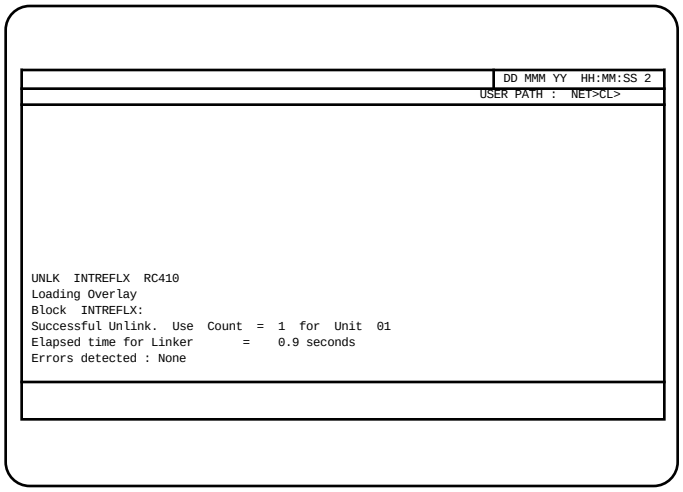
11904

5.3.2.5 Invoking an Unlink Command

If the program is bound to more than one point, unlink it one point at a time.

1. Key in an Unlink command. Here we keyed in "`UNLK INTREFLX RC410`" to unlink the block we linked under subsection 5.3.2.4. "INTREFLX" is the block name and RC410 is the point's tagname.

2. Press ENTER. When the program is unlinked (is no longer bound to the point), an "Elapsed time for Linker" message appears. Here that message says unlinking took 0.9 seconds. The "Elapsed time" message is followed by an "Errors detected" message. Here that message says no errors were detected.



12184

During the unlinking process, as errors are detected, error messages appear on the display. These messages are described in Section 6. Section 6 also provides guidelines for error recovery.

5.3.2.6 Invoking a Compile and Link Command

This is a combination of the Compile command (see subsection 5.3.2.3) and the Link command (see subsection 5.3.2.4). If the compiler detects errors other than errors that cause notes and warnings, the link doesn't take place. To request a compile and link, key in the command in the form of this example and press ENTER:

```
CLK INTREFLX RC410
```

You can also include any of the options. See subsection 5.3.1.2.

5.4 HOW TO USE COMMAND FILES

CL and Command Processor commands can be included in command files to cause them to be executed without human interaction. The Command Processor commands that facilitate the execution of command files are available through the Control Language Compiler/ Linker Display.

`DO (pathname)`	Data Out
`P (pathname)`	Print
`EC (pathname)`	Execute Command File
`PO (pathname)`	Prompt Out
`Pause`	Pause (stop execution) until the operator presses ENTER.
`End`	End Command File Execution and this Utilities session.

where: (pathname) is a complete file pathname for either disk/diskette or HM.

See subsection 5.1 for a description of the Data Out and Print commands.

The EC command directs the Universal Station to take commands from the specified file, rather than from the keyboard. The commands are displayed on the screen as they are read from the file. The responses to the commands also go to the screen, just as if they were being executed through interaction with an operator. Command execution continues until an End command is encountered or the end of the file is reachedand you are returned to the Command Processor.

You can abort EC file execution by holding CTL and pressing COMND. When the command being executed is complete, execution stops and you are returned to the Command Processor.

"&P" can be included in a command file to indicate comments that are not considered as commands. "&P" and the text that follows it appear on the screen as they are encountered and, if so specified by a PO command, they are also printed or appended to a file.

The PO command directs the US to copy the commands and responses as they appear on the screen, to the specified printer, or to append them to the specified file. This continues until an End command or the end of the file terminates execution, or until a PO command with no pathname is executed.

The PAUSE command causes command execution to cease. Command execution continues when the ENTER button is pressed. The primary purpose of the PAUSE command is to allow you to mount a disk/diskette during the execution of an Execute file.

We recommend that you use a .EC suffix on your command files to identify their purpose. The EC and PROMPTOUT commands don't require this specific suffix, but you will find a .EC suffix a help in remembering the purpose of the file.

As an example, you could use the Text Editor to create a file named ECFILE in a volume named of EC that contains the following:

```
PO NET>CL>SESSION.EC
&P This command file compiles two CL blocks
CL PUMP1

CL PUMP2
END
```

Then on the Control Language Compiler/Linker display, key in

```
EC NET>EC>ECFILE.EC
```

and press ENTER. The PO command directs the commands, the &P line, and all responses to file SESSION.EC. The two CL commands are then executed, and the End command terminates the session and returns you to the Command Processor.

If you wanted to review the commands and responses that were stored in SESSION.EC, you could key in

```
PR NET>CL>SESSION.EC
```

Then press ENTER. The content of the file appears on the display.

ERROR INDICATIONS AND RECOVERY
Section 6

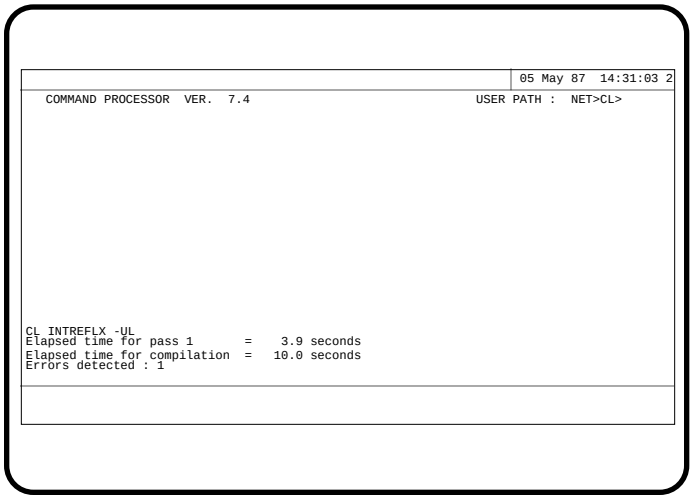
Use subsections 6.1 and 6.2 to recover from errors detected by the Compiler and Linker commands. Subsection 6.3 provides information that will help you to recover from errors that are detected as a CL program is running in an Application Module.

6.1 HOW ERRORS ARE INDICATED

CL data entry errors are indicated two ways:

- On the Command Processor Display.
- In error listing (.LE) files that the Compiler generates when it detects errors.

As each CL command (compile, link, compile and link, and unlink; see Section 5) is executed, messages appear on the Control Language Compiler/Linker display that indicate progress as the command is executed and that provide information that may or may not indicate errors. The content of these messages must be considered, to determine if corrective action is needed or not. For example, this display indicates that a Compile command was executed; however, the last message on this display indicates that one error was detected during the compilation. That error can be found in the error-listing (.LE) file.



3120

Errors detected by the Link command appear on only the Control Language Compiler/Linker display.

6.1.1 CL Compiler Error Messages

An alphabetical list of all of the error messages generated by the compile, link (CL/AM only), and unlink (CL/AM only) commands is provided below. Note that errors shown as P3MC or P3PM also apply to CL/APM. The information in the Source/Severity column has the following meaning:

- **Source**:

  CX – Communication and data access messages

  LK – Linker messages (CL/AM only)

  P1, P2, P3, P3CDS, P3MC, P3PM, P3APM – Compiler messages generated on pass 1, pass 2, or pass 3. These messages appear in the error listing (.LE) file.

The major activity in each of the Compiler passes is as follows:

Pass 1—CL syntax checking

Pass 2—Parameter type checking

Pass 3—Object code generated for a CL/AM compilation

P3CDS—Custom GDF generated (CL/AM only)

P3MC—Object code generated for a CL/MC compilation

P3PM—Object code generated for a CL/PM compilation

P3APM—Object code generated for a CL/APM compilation

- **Severity** (listed from most severe to least severe):

  Fatal—An error that prevents further command execution or that prevents the CL Compiler from executing beyond the point where the error occurred

  Severe—A severe, but not fatal error

  Error—An error that did not prevent command execution

  Note—The Compiler has done something that may be surprising or noteworthy

  Warning—The user has done something noteworthy.

Messages	Source/Severity
%INCLUDE_EQUIPMENT_LIST directive is not supported for CL/AM	P2-Fatal
%INCLUDE_EQUIPMENT_LIST directive only alowed as the first line	P1-Error
%INCLUDE_EQUIPMENT_LIST is not declared as the first line	P1-Error
&ASY volume not found	CX-Fatal
(filename) could not be created:<reason>	CX-Fatal
(filename) could not be deleted <reason>	CX-Fatal
(filename) could not be read: <reason>	CX-Fatal
(filename) Handler actually defined as (filename)	P2-Error
(filename) Not Found	P2-Error
(tagname.param) compile type does not match actual type	CX-Fatal
(tagname.param) does not exist	LK-Fatal
(tagname.param) is an array of point IDs that do not all refer to the same LCN	LK-Fatal
<name> cannot be redeclared	P2-Error
<name> expected	P1-Error
<name> is not a legal %RELAX option	P2-Error
<name> not found	P2-Error
<name> was not found in the system and -UL was not selected	CX-Error
A local variable is required here	P2-Error
A Loop can have only one REPEAT statement	P2-Error
A number cannot be divided by a time	P2-Error

(Continued)

Messages (Continued)	Source/Severity
A remote variable is required here	P2-Error
ALARM time must be an integer constant (0-9999) for MC	P3MC/PM/APM-Error
Aliasing of Bound Data Point	P2-Error
Alias Name does not exist in Equipment List Object File	P1-Error
Allowed only before first Block in a Package or in Block data declarations	P2-Error
All enabled abnormal handlers are disabled	P2-Note
Argument multiply defined	P2-Error
Argument must be a Data Point Param of type Data Point	P2-Error
Argument must be a Data Point Param or a LOCAL	P2-Error
Argument must be an unsubscripted data point parameter	P2-Error
Argument must be of type ENUMERATION	P2-Error
Argument must be of type String Literal	P3APM-Error
Array expected	P2-Error
Array index must be of type number	P3MC/PM/APM-Error
Array too large	P3-Fatal
Arrays cannot be compared	P2-Error
Attempt to fetch element of off node array of strings: <pointname>.<parameter>	LK-Error
Attempt to redefine (name) as a parameter name	P2-Error
Attempt to redefine parameter (name) with different type	P2-Error
Attempt to store to (tagname.param) whose access lock is View Only	P2-Error
Attempted Aliasing	P2-Error
Attempted indirection	P2-Error
Attribute previously defined	P2-Error
Background Block Active on Bound Data Point—retry Link or Unlink	LK-Fatal
BACKGROUND is allowed only in Platform Subroutines and/or Functions	P2-Error
Block already exists at another Insertion Point	LK-Fatal
Block exceeds maximum number of prefetches allowed per block	LK-Fatal
Block insertion point must be BACKGRND when calling background routines	P3-Error
Block not on the Bound Data Point	LK-Fatal
Both KEEPENB and abnormal handlers not allowed in a Phase heading	P2-Error
Bound Data Point at maximum CLs	LK-Fatal
Bound Data Point has already been processed in a previous compilation	P2-Error
Bound Data Point is active	LK-Fatal
Bound Data Point not configured for CLs	LK-Fatal
C-Link data inconsistent in READ/WRITE: may indicate that no C-Link exists	P3MC-Error
Cannot create .GD file, FMS error: <reason>	CX-Fatal
Cannot declare a data type with REFERENCE_N	P2-Error
Cannot fetch values from an OUT argument	P2-Error
Cannot access this parameter from this sequence program	P3PM/APM-Error
Cannot generate code for this point reference	P3PM/APM-Error
Cannot get default pathname	CX-Error

(Continued)

Messages (Continued)	Source/Severity
Cannot have LOCAL data definitions with the data type of Entity ID in a CL Block	P2-Error
Cannot INITIATE non Process Module Point	P3MC/PM/APM-Error
Cannot INITIATE own normal sequence	P3MC/PM/APM-Error
Cannot Link to a Library subroutine	LK-Fatal
Cannot loop in Phase Heading	P3MC/PM/APM-Error
Cannot loop in Step Heading	P3MC/PM/APM-Error
Cannot modify a local constant or enumeration literal	P2-Error
Cannot modify a Point ID—use Data Entity Builder	P2-Error
Cannot modify an expression	P2-Error
Cannot modify an IN argument	P2-Error
Cannot modify function	P2-Error
Cannot modify read-only parameter	P3MC/PM/APM-Error
Cannot multiply time by time	P2-Error
Cannot pass a member of an array of data Points	P2-Error
Cannot Read Pointer from Global memory	CX-Fatal
Cannot Read/Write subroutine argument	P3MC/PM/APM-Error
Cannot Read/Write this parameter	P3MC/PM/APM-Error
Cannot Reference @1.@2	LK-Fatal
Cannot reference data points with names greater than 8 characters across LCNs	P2-Error
Cannot reference Off-LCN $AMIF Point	P2-Error
Cannot reference off-LCN points in a sequence program	P2-Error
Cannot Restart in emergency handler	P3MC/PM/APM-Error
Cannot SEND Enumeration types with blank state names	P3MC/PM/APM-Error
Cannot SEND Enumerations	P2-Error
Cannot SEND mode type value	P3MC-Error
Cannot send Arrays	P2-Error
Cannot send Conditions	P2-Error
Cannot store to an IOL parameter in a READ statement	P3PM-Error
Cannot use a parameter which requires a Prefetch in a CALL statement	P3PM-Error
Cannot use an entire array	P3MC/PM/APM-Error
Cannot use WHEN ERROR clause when INITIATE own handler	P3MC/PM/APM-Error
Checksum Error on Linker/Loader communication	LK-Fatal
Code size exceeds maximum of 12288 words	P3APM-Error
Compiler memory overflow! Program too large or statement too complex	CX-Fatal
Composite Data Point must reside in a MC	P3MC/PM/APM-Error
Constant is too big	P3MC/PM/APM-Error
Corrupted Data Base	LK-Fatal
Code Size exceeds maximum of 8192 words	P3PM-Fatal
Could not finish reading file (filename) <reason>	CX-Fatal
Could not get file attribute (filename) <reason>	CX-Fatal
Could not get memory in AM	LK-Fatal
Could not go to End of File (filename) <reason>	CX-Fatal
Could not open %Include_Set file @1: @2	CX-Error
Could not open file (filename) <reason>	CX-Fatal
Could not write to file (filename):<reason>	CX-Fatal
Custom Set @1 is not loaded in the AM	LK-Error

(Continued)

Messages (Continued)	Source/Severity
DA error adding name to global directory, Build status=(value)	CX-Fatal
DA error converting entity name, minor_stat: (value)	CX-Error
DA error getting enumeration definition	CX-Fatal
DA error getting Point Parameters, se_stat: (value)	CX-Error
DA ext/int conv error, <reason>	CX-Fatal
DA int/ext conversion error, codes=(values)	CX-Fatal
Data Access Disk I/O Error:<reason>	CX-Fatal
Data Access error: Enumeration Set not found	CX-Error
Data Point @1 does not exist. Rebuild Equipment List.	P2-Fatal
Data Point @1 does not exist	LK-Fatal and Error
Data Point @1 not found—all nodes not operational	LK-Fatal and Error
Data Point Identifier expected	P2-Error
Data Point is too large	LK-Fatal
Data Point must be a MC point	P3MC-Error
Data Point must be in different MC from Bound Data Point	P3MC-Error
Data Point must be in the same UCN node as the bound data point	P3PM/APM-Error
Data Point must be in the same MC as Bound Data Point	P3MC-Error
Data Point must be on same Hiway as Bound Data Point	P3MC-Error
Data type DATA_POINT_ID must be followed by REFERENCE	P2-Error
Data Type name may not appear here	P2-Error
Data Type was not found in system	P2-Error
Declared size exceeds actual array bounds	P2-Error
Declared with invalid Data Type	P2-Error
Default value exceeds Port Size	P3CDS-Warning
Defined but not referenced	P2-Warning
Delaying for busy Loader	LK-Note
Destination must be a Data Point with a Mailbox	P2-Error
Different version of object exists in Unit <number> Usecount = <number>	LK-Fatal
Disk/Memory revision mismatch on Global Name File	CX-Fatal
Does not match Heading Name	P2-Error
Duplicate Declaration	P2-Error
ELSE cannot begin a Statement	P2-Error
Empty Custom Data Segment not permitted	P2-Error
Empty Object File	LK-Fatal
Empty SEND statements are not permitted in sequence programs	P3MC/PM/APM-Error
End of File expected	P1-Error
End of Line expected	P1-Severe
Entity Name could not be converted	CX-Error
Enumeration <name> is not valid on the UCN	CX-Error
Enumeration (name) not found—all nodes not operational	LK-Fatal
Enumeration literal expected	P2-Error
Enumeration literal may not appear here	P2-Error
Enumeration literal not implemented	P3MC-Error
Enumeration literal not in this Type	P2-Error
Enumeration name conflicts with Parameter List Name in Custom Library	CX-Error
Enumeration Set does not match actual enumeration set	CX-Error
Enumeration set <name> was not found in custom name library	P3CDS-Error
Enumeration state <name> not found - all nodes not operational	LK-Fatal

(Continued)

Messages (Continued)	Source/Severity
Enumeration state not valid for this sequence program	P3PM/APM-Error
Enumeration type of this object is invalid	P3MC-Error
Enumeration types with blank state names are not supported	CX-Error
Equipment List Object version mismatch - Translate or rebuild Equipment List	CX-Error
Error attempting to get parameter information; S:@1 M:@2 PA:@3	CX-Fatal
Error during attempt to get parameter information; PA Status:<status>	CX-Fatal
Error during attempt to get SDE state names; Minor Status:<reason>	CX-Fatal
EU string truncated	P3CDS-Warning
Exceeded maximum of 3000 characters allowed for Value statements	P3CDS-Fatal
Expression is invalid in READ or WRITE statement	P3MC/PM/APM-Error
Expression is not constant or cannot be evaluated	P2-Error
Expression too complicated for evaluation	P3-Fatal
Fetching UCN Box Point parameters NN, FL, or TIME is not allowed in foreground CL	P3/LK-Error
File Manager Request Block—Allocation Error	CX-Fatal
File suffix is not Equipment List Object Suffix (QO)	P1-Error
Foreground fetch of Array Point <tagname>.TIME which has > 160 elements	LK-Error
Function Result must be scalar or string	P2-Error
Global Name File full	CX-Error
Handler already enabled for this condition	P2-Error
Handler contains multiple RESTARTS	P2-Error
Handler expected	P2-Error
Hardware location is required	P3MC/PM/APM-Error
Hardware location must be NN or FL	P3MC/PM-Error
Hardware location must be NN, FL, TIME, STR8, STR16, STR32, or STR64	P3APM-Error
Heading expected here	P1-Error
Heap memory requirements exceed available heap memory	CX-Fatal
I/O card type of bound MC is inconsistent	P3MC/PM/APM-Error
Identifier cannot be resolved in this context	P2-Error
Identifier expected here	P1-Error
Identifier length error	CX-Error
Identifier must be PM, APM, or MC—defaulted to PM	P2-Error
Identifier must be within 8 characters	P3MC-Error
Illegal character	P1-Error
Illegal compiler directive	P1-Severe
Illegal external Name	CX-Error
Illegal use of "%"	P1-Error
Incorrect format string	P3APM-Error
Index Type mismatch	P2-Error
INITIATE Point.Param not allowed in this sequence program	P2-Error
INITIATE SEQ not allowed in this sequence program	P2-Error
Insertion Point invalid for build type	LK-Error
Internal communication error (IDB) - Contact Honeywell TAC	CX-Fatal

(Continued)

Messages (Continued)	Source/Severity
Internal error; Notify your Honeywell Representative:	
Converting LCN Name	LK-Fatal
Invalid access key	CX-Error
Invalid Array Base type	P2-Error
Invalid CALL argument	P3MC/PM/APM-Error
Invalid constant data type	P3MC/PM/APM-Error
Invalid Declaration	P3MC/PM/APM-Fatal
Invalid declaration for a Sequence Program	P2-Error
Invalid File Name	CX-Fatal
Invalid File Pathname or Pathname not specified	P1/CX-Error
Invalid in a Sequence	P2-Error
Invalid in Blocks	P2-Error
Invalid in this context	P2-Severe
Invalid %Include_Set file name, must be 6 characters or less	P1-Error
Invalid Insertion Point (name)	LK-Error
Invalid insertion point for configured algorithm	CX-Error
Invalid mode enumeration for this point	P3MC/PM/APM-Error
Invalid Module Number, Entity does not exist	CX-Error
Invalid parameter on this point	P3MC/PM/APM-Error
Invalid point for this sequence	P2-Error
Invalid point type for this compilation	LK-Error
Invalid point type for this compilation	P2-Error
Invalid SEND destination	P3MC/PM/APM-Error
Invalid SEND item	P3MC/PM/APM-Error
Invalid SEND WAIT option for this Destination	P3MC/PM/APM-Error
Invalid Slot Number, Entity does not exist	CX-Error
Invalid State Name	P2-Error
Invalid subscript for this parameter	P3MC/PM/APM-Error
Invalid time expression	P3MC/PM/APM-Error
Invalid to indirect through an off-node point	LK-Fatal
Keylock error on attempt to store to (tagname.parameter)	LK-Error
Labels must be unique within a Program Block	P2-Error
Library full of Names/Strings, id not added	P3PM-Error
Library full of Sequence/Phase/Step id, id not added	P3MC/PM/APM-Error
Library full of Strings, String not added	P3MC/PM/APM-Error
Library Subroutine not permitted in a Package	P2-Error
Library Subroutine Object corrupted	LK-Fatal
Linking/Unlinking to points on other LCNs is not supported	LK-Fatal
Loader communication error	LK-Fatal
Loader still busy after one retry	LK-Fatal
Loader time out error	LK-Fatal
Loader/Data Access communication problem	LK-Fatal
Loader/Kernel communications problem	LK-Fatal
Local Subroutine cannot have a Bound Data Point	P2-Error
Local time not synchronized with network	CX-Note
Local variables may not be data point identifiers	P2-Error
Logical condition expected	P2-Error
Logical conditions cannot be compared	P2-Error
Logical or Enumeration type expected here	P2-Error
Loop Index not declared	P2-Error

(Continued)

Messages (Continued)	Source/Severity
Loop was never repeated	P2-Warning
Lower bound must be less than upper bound	P2-Error
Malformed Identifier	P1-Error
Malformed number	P1-Error
Malformed Point Identifier	CX-Fatal
MAN, AUTO, CASC, and COM may be stored but not compared	P3MC-Error
Missing Abnormal Handler ID	P1-Error
Missing Closing Quote	P1-Error
Missing END	P1-Error
More than 12 IOL parameter fetches in one step	P3PM/APM-Error
MSF Error - Code:@1	P1/P2-Fatal
MSF FM Error - @1:@2	P1/P2-Fatal
MSF.LO is not installed in this node	P1-Fatal
Multiple locals assigned to this hardware location	P2-Error
Multiple stores of strings are not permitted	P2-Error
Must be a BDP Parameter	P2-Error
Must be a Parameter List	P2-Error
Must be a positive integer	P2-Error
Must be an integer	P2-Error
Must be APM resident point	CX-Error
Must be in the RESTART section of an ABNORMAL Handler	P2-Error
Must be MC resident point	CX-Error
Must be PM resident point	CX-Error
Must contain executable statements	P2-Error
Must link to point (name)	LK-Fatal
Name/String not found in Library and -UL was not selected	P3PM-Error
Name too long	P3-Error
Nested CALL invalid	P3MC/PM/APM-Error
Nested PACKAGE heading ignored	P2-Severe
Network in the EL Object does not match with compilation target	P2-Fatal
No available entries in Library for Sequence names, Sequence name not added	P3PM/APM-Error
No corresponding item for this Variable	P2-Error
No response from the remote AM node	LK-Fatal
Not a parameter	P2-Error
Not a Process Module point	CX-Error
Not a Subroutine	P2-Error
Not Enough Contiguous Memory Error	LK-Fatal
Not enough memory	CX-Fatal
Not supported	All-Error
Not supported on target machine	P3-Fatal
Not valid for AM	P2-Error
Null point ID only permitted in point ID arrays	LK-Fatal
Number of values differs from number of array elements	P2-Error
Numeric literal expected here	P1-Error
Object code memory requirement exceeds maximum allowable object size	LK-Fatal
Object corrupted in transmission from US to AM	LK-Fatal
Object File Access Error	LK-Fatal
Object File Corrupted	LK-Fatal

(Continued)

Messages (Continued)	Source/Severity
Object file FMRB allocation error	LK-Fatal
Obsolete syntax - this name is ignored	P2-Error
Old File (filename) cannot be safely renamed (filename)	CX-Fatal
Only one %INCLUDE_EQUIPMENT_LIST directive is allowed	CX-Error
Only one Custom Data definition is permitted	P2-Error
Only valid in a Custom Data Segment	P2-Error
Parameter (name) not found—all nodes not operational	LK-Fatal
Parameter (name) not on this Data Point	P2-Error
Parameter conflict	P2-Error
Parameter Declaration expected	P2-Severe
Parameter Description truncated	P3CDS-Warning
Parameter List <name> was not found in custom name library	P3CDS-Error
Parameter List File (filename) is empty	P2-Error
Parameter List name conflicts with Enumeration Set Name in Custom Library	CX-Error
Parameter name conflicts with Segment name in Custom Library	CX-Error
Parameter name not allowed in this context	P2-Error
Parameter references used in a subscript must be On-Node	P3APM-Error
Parameters data type not supported	P2-Error
Passing enumeration literals as arguments in a CALL is not allowed	P3MC-Error
Phase Heading contains multiple ALARM clauses	P2-Error
Point ID change only valid after unlink and link	P3-Warning
Point Identifiers cannot be compared	P2-Error
Point is not NIM resident	CX-Error
Point must be declared in an EXTERNAL statement prior to the LOCAL declaration	P2_Error
Point must be on the same UCN as the Bound Data Point	CX-Error
Point must be resident on the UCN	CX-Error
PPS Parameter not logical	P2-Fatal
Program is too large	P3MC/PM/APM-Error
PRTFL parameter of composite point contains only the status of the input slot	P3MC-Warning
Quoted string is too long. Maximum length allowed is <value>	P1-Error
Reference List entry created for each element of this array	P3-Note
References an off-LCN point; must link to IPP or Backgrnd Insertion Point	LK-Fatal
Remote variable in READ/WRITE must be in different box from Bound Data Point	P3MC/PM/APM-Error
Remote variable in READ/WRITE/INITIATE must be on same C-LINK as Bound Data Pt	P3MC-Error
RESUME must have PHASE destination	P3MC/PM/APM-Error
Routine must begin with step	P2-Error
Routine Out of Order	P2-Error
Same object exists with different access key in Unit <name>. Use count = <value>	LK-Fatal
Segment name conflicts with standard parameter or param name in Custom Library	CX-Error
Segment size exceeds maximum by <number> bytes	P3CDS-Error
Sequence must begin with a PHASE	P2-Error
Sequence/Phase/Step ID must be defined in Library	P3MC/PM/APM-Error

(Continued)

Messages (Continued)	Source/Severity
Source File contains no code	P1-Fatal
Source File is empty	CX-Fatal
Statements not valid in Custom Data Segments	P2-Error
String exceeded maximum allowable length and was shortened to <value> characters	P3CDS-Warning
String is valid only in SEND statement	P3MC/PM-Error
String literal expected here	P1-Error
String literal type can only be general (G), integer (I), or real (R) format	P3APM-Error
String must be defined in Library	P3MC/PM/APM-Error
String must be within 8 characters	P3MC/PM/APM-Error
Subroutine/Function name must be less than or equal to 80 characters	CX-Error
Subscript must be an integer constant	P3MC/PM/APM-Error
Subscript outside array bounds	P3-Error
Subscript used too deep in an indirect reference	P3-Error
Syntax Error	P1-Severe
System Name conflict	CX-Fatal
Tagname access references only the upper subslot of this dual digital parameter	P3MC-Warning
Target does not define a loop	P2-Error
Temp File (filename) could not be renamed:(filename)	CX-Fatal
The AM must be configured with at least one BACKGROUND task	LK-Error
The argument being passed in is not a Data Point Parameter	P2-Error
The argument being passed in is not an Enumeration type	P2-Error
The Data Point Name @1 is invalid	LK-Fatal
The loop counter must be a LOCAL variable	P2-Error
The referenced parameter list (filename) was not found	P2-Error
The target of a CALL statement must be a subroutine	P2-Error
There are too many parameter references in this block	LK-Fatal
This array limited to one dimension	P2-Error
This data type may only be compared equal/unequal	P2-Error
This Include Set file has an error on line @1, column @2	CX-Error
This is not an array	P2-Error
This LOCAL variable is mapped to a Serial Interface IOP parameter	P3APM-Warning
This parameter name cannot be used in a Custom Data Segment	P3CDS-Error
Time expression may not be followed by a WHEN clause	P2-Error
Time may be wrong	CX-Note
Too few arguments	P2-Error
Too many arguments	P2-Severe
Too many Array elements	P3CDS-Fatal
Too many array elements individually initialized with the VALUE statement	P3CDS-Error
Too many Build Visible Parameters	P3CDS-Fatal
Too many constants in program	P3MC/PM/APM-Error
Too many enumerations	P2-Error
Too many Links to Unit	LK-Fatal
Too many Parameters in Segment, Entity Builder Limit	P3CDS-Error
Too many Segments in Package, Entity Builder Limit	P3CDS-Error

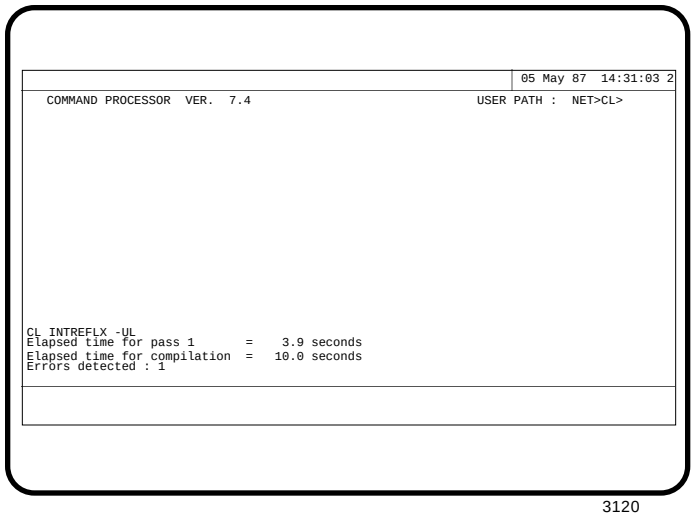
(Continued)

Messages (Continued)	Source/Severity
Too many SEND items	P3MC/PM/APM-Error
Too many statements in a Step/Routine	P3MC/PM/APM-Error
Too many variables in this statement	P3MC/PM/APM-Error
Total Variables exceeds available memory	P3-Fatal
Type (name) expected	P2-Error
Type mismatch between variable type and hardware location	P3MC-Error
Type mismatch	P2-Error
Undefined Enumeration	CX-Error
Undefined Link Error	LK-Fatal
Undefined Symbol	P2-Error
Unexpected Attribute Statement	P2-Error
Unexpected End of File	P1-Error
Unexpected Record	LK-Fatal
Unexpected text after end of program	P1-Severe
Unit Instance name does not exist in Equipment List Object File	P2-Fatal
Unlink required—another block at specified Insertion Point	LK-Fatal
Unlink required—block already on Bound Data Point	LK-Fatal
Unmatched END statement ignored	P2-Severe
Unrecognizable Point Type	CX-Error
Unresolved forward reference	P2-Error
Use standard enumeration @1 instead of declaring enumeration states	P2-Error
Value cannot be Converted	P3-Fatal
Value does not match parameter type	P2-Error
Variables requiring prefetch are not permitted in the Phase alarm clause	P3APM-Error
Wrong Data Type for this Operator	P2-Error
Wrong number of Dimensions	P2-Error

6.2 CORRECTING CL SOURCE FILE ERRORS

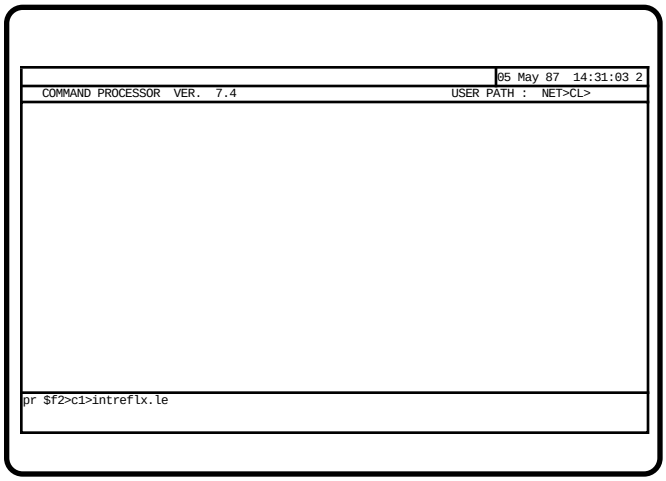
The following are the steps used to correct an actual, but simple, CL source file error. A similar process can be used to correct all CL source file errors.

1. This is how the display appeared when compilation of the CL source file was completed. The last line on this display indicates that the Compiler detected one error. We need to look at the content of the error listing (.LE) file to find the error.



3120

2. We keyed in "`PR $f2>cl>intreflx.le`" followed by ENTER, to request that the error listing be displayed.



3121

3. The entire .LE file is displayed, line-by-line. When the end of the file is reached, a "Print Complete" message appears. You can now use PAGE FWD and PAGE BACK to page through the file to find the error messages.

```
                                                                05 May 87  14:31:03 2
  COMMAND PROCESSOR  VER.  7.4                                  USER PATH :  NET>CL>
      47  SET x = reflux.pv/feed.pv * (1.0 + (cp * delta.t/hv))
      48  IF BADVAL(x) THEN GO TO badv
      49  SET pvcalv = x
      50  SET pvautost = normal
  CL V14.2   INTRREFLX                                                  14 10 85 14:25
:023646    Page        2
      51  --
      52  IF init + off THEN EXIT
      53  ESLE (SET CTRLINIT = ON:
      54  &     EXIT)
      55  --
      56  badpv : CALL SET BAD(pvcalc)
      57  SET pvautost = bad
      58  END Reflextl
      59
      60  End Intreflx
      61
  ********  Number of errors detected:   1
Print Complete
```

3123

4. We found our error on the first page of the .LE file. The `**ERROR ** Missing Closing Quote` message is just below the line with the error. The "^" is pointing to the beginning of the field with the error. The Compiler found that there was only one quote mark on the `EU "POINT` line. It couldn't determine where the second or closing quote mark should be. We need to add a quote mark after the `T` in `POINT`.

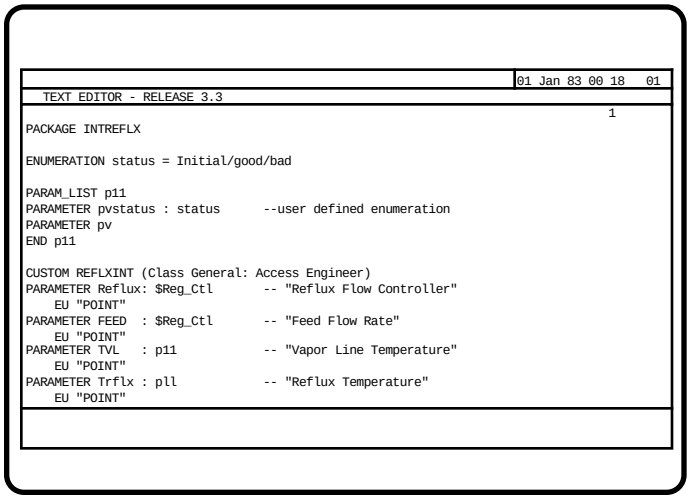
```
                                                                05 May 87  14:31:03 2
  COMMAND PROCESSOR  VER.  7.4                                  USER PATH :  NET>CL>
       2
       3  ENUMERATION status = Initial/good/bad
       4
       5  PARAM.LIST p11
       6  PARAMETER pvstatus : status     --User defined status
       7  PARAMETER pv
       8  END p11
       9
      10  CUSTOM REFLXINT (Class General: Access Engineer)
      11  PARAMETER Reflux: $RegCtl       --Reflux Flow Controller
      12      EU "POINT
                 ^
**ERROR ** MISSING CLOSING QUOTE

      13  PARAMETER FEED  : $REG.CTL      --FEED FLOW RATE
      14      EU "POINT"
      15  PARAMETER TVL   : p11           --Vapor line Temperature
      16      EU "POINT"
      17  PARAMETER Trflx : pll           --Reflux Temperature
```

3124

5. Type ED followed by the pathname of the source file that needs to be corrected, example: `$f2>cl>intreflx.CL`, and then press ENTER (be sure that the .CL file is available in the floppy drive or on an HM).

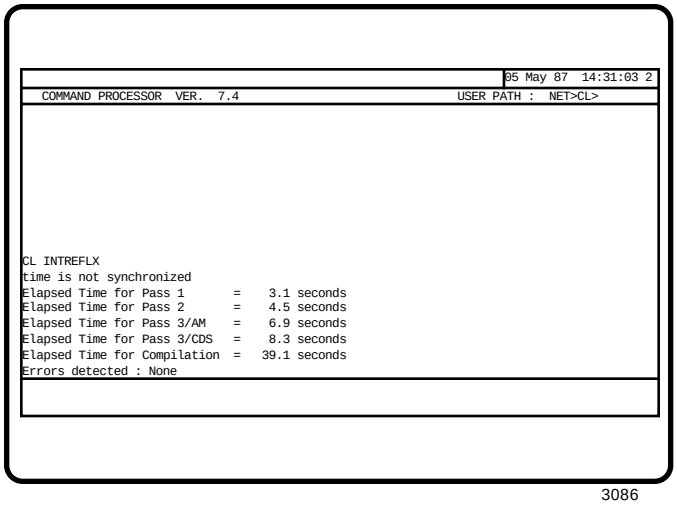
6. The first 23 lines of the .CL file appear. You can page forward and backward through this display. Our error is about halfway down the first page on the '`EU POINT`' line. We keyed in a quote mark after the word "`POINT`" to correct . the error. Then we pressed LF, followed by E, and RETURN, to transfer the corrected file to the volume it came from.



3125

7. The display returns to the Command Processor display.

8. Type `cl intreflx`, press ENTER to recompile the CL-source file. This display shows that this time the compilation took 39.1 seconds, and there are now no errors.



3086

In the example of steps 1 through 7, an error was detected on the first Compiler pass, and passes 2 and 3 did not take place. For each pass to complete, the preceding pass must be error free. Once you have corrected an error in an early pass, you may still have more errors to correct in subsequent passes.

Most errors detected by the Link command are fatal; therefore, if you correct one such error, there may still be others to correct after you attempt to link again.

6.3 HOW TO DETERMINE THE LOCATION OF A CL RUNTIME ERROR

The CL listing (.LS) file includes program-code location information. This information is found in the "Loc" column in the example provided below.

For CL/AM programs, the number in the "Loc" column for each statement is the offset in words, relative to the start of the object code for the CL block. When a CL error or failure occurs at runtime, the location counter of the machine instruction where the error occurred is placed in the CL Segment's CLERRLOC parameter array. There is one CLERRLOC value for each CL slot in the CL segment (see subsection 4.1.5 in the *Application Module Control Functions* manual). These values are displayed in the "Loc" column of the Point Detail Display-CL Block page.

After you have determined the error location from CLERRLOC, refer to the "Loc" column of the listing file for the CL block. The statement that caused the error is the one whose location is the largest number less than the value in CLERRLOC. The first statement of a block is seldom located at the start of the block. There is usually a code segment that performs initialization of local variables at the beginning of each block, and this segment is transparent to the user.

You can determine error locations with better resolution if, when you write the source code, you continue long statements across multiple lines, using the continuation character "&." Also note that the location numbers increase throughout the block and its subroutines, but start over at the next block.

```
Line      Loc          Text

  1               package
  2
  3               block block1 (generic; at general)
  4
  5               local a, b
  6
  7       25      set a = 1
  8
  9       30      set b = 2
 10
 11       35      if a > b then set a = sin(3.14)
 12       62      else if a = b then (set a = 0;
 13       79      &                   set b = 1;
 14       84      &                   call sub1)
 15
 16       95      send   "pctest:  a + b = ", a + b
 17
 18               end block1
 19
 20               subroutine sub1
 21
 22               local i, j
 23
 24       168     set i =1
 25
 26       173     set j = 1
 27
```

Line	Loc	Text
28	177	lp: loop for i in 1..2
29		
30	203	set j = i
31		
32	207	repeat lp
33		
34		end sub1
35		
36		block block2 (generic; at general)
37		
38		local c, d
39		
40	24	set c = 1
41		
42	29	set d = 2
43		
44	34	if c = d then
45	46	& send: "equal"
46		else
47	86	& send: "unequal"
48		
49		end block2
50		
51		end package

Index

Index

READER COMMENTS

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

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

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

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

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

Title of Publication: **Control Language/AM Data Entry** Issue Date: **9/95**

Publication Number: **AM11-585**

Writer: **Maria Nelson**

COMMENTS: _____

_____

_____

_____

_____

_____

_____

_____

RECOMMENDATIONS:_____

_____

_____

_____

_____

_____

_____

_____

NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

TELEPHONE _____ FAX _____

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

Communications concerning technical publications should be directed to:

Automation College
Industrial Automation and Control
Honeywell Inc.
2820 West Kelton Lane
Phoenix, Arizona 85023-3028

FOLD FOLD

From: _____

NO POSTAGE
NECESSARY
IF MAILED
IN THE USA

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ....

Honeywell

Industrial Automation and Control
2820 West Kelton Lane
Phoenix, Arizona 85023-3028

Attention: Manager, Quality

FOLD FOLD

Cut Along Line

Additional Comments:

Honeywell

Industrial Automation and Control
Honeywell Inc.
16404 North Black Canyon Highway
Phoenix, Arizona 85023-3099

Helping You Control Your World