CP Utilities

Written by:
Alex Johnson

V. 4.03 08/05/93
V. 4.07 04/20/98
V. 4.08 01/05/98 (New tool; V4.3/6.1 support)

Table of Contents

1   REFS – CONTROL PROCESSOR WORKFILE REFERENCE CHECKER AND REPORTER

Detailed help is printed if one invokes the program without any arguments. The help text is also printed by default when the program is run. To suppress the detailed help use the `-h` *option.*

refs reads on-line or off-line workfiles produced by the Integrated Control Configurator. These files can be for any station type, e.g., CP, FDG, SIP, etc.

A second version of refs *exists,* refsnh. *This version eliminates the help text and can handle larger workfiles. In all other respects its is identical to* refs.

refs lists:

1) With varying degrees of detail, the block connections within a single workfile (on-line or off-line).
2) Non-default settings for ECBs in the workfile ,
3) External, i.e. peer-to-peer, connections,
4) Blocks per compound in block processing order,
5) All blocks in the workfile in alphabetical order with additional information, e.g. description, period, & phase,
6) All blocks with hardware connections, i.e. all I/O blocks, including description, compound, FBM letterbug, and point number,
7) All blocks by type showing the number in each period and the number in each phase for the first four periods (This table is most useful when combined with the I/A Sizing Spreadsheet available from Field Service),
8) Optimal ECB periods based on the rates of the blocks connected to the ECBs,
9) In tabular form all I/O points used on each FBM (This allows one to quickly check for doubly referenced points),
10) Total number of blocks in each period in the workfile,
11) Total number of blocks of each type in the workfile, and
12) A rough indication of the expected load based on both blocks per second and ED 3648. Per company policy, the control (non-ECB) blocks are sized at one-half the CP-10 load if the -CP30 option is used. If ECBs are present in the file, their actual periods will be used for CPU calculations; not the optimal ECB periods. The load attributed to CALC blocks will be pro-rated in the ED 3648 based calculations, i.e. a CALC block with 25 steps will have ½ the attributed load.
13) A "source code" version of the workfile. This listing prints all non-default (or, optionally, all) parameters for all of the blocks and compounds. It is suitable for use with bldcio.

Many of the reports can be printed individually or combined into system-wide reports.

In addition, *refs* can produce a list of (compound, CP Name) pairs. This option can be used to provide a Master Compound list for the entire system. The Master Compound list can be used with my other utilities to more fully document the system.

Please note: *refs* will report :

```
Current name of bt = %d is (%s) not (%s).
 refs is out of date.
 Block type %d name will be wrong because refs cannot store the new name.
 Memory and CPU load sizing will be off because refs does not have the new data.
 Other functions should be correct.
```

for each new block type since *refs* was built, i.e. if you use it on subsequent releases.

The ED values used by *refs* are those from ED3658 dated May 1989. I used the size and timing numbers for the RATIO block for the BIAS block.

I have set the ED 3648 values for the Spectrum Master Gateway blocks, the FDG blocks, the 76x blocks, and the Interspec Integrator blocks to 0.00 since I have no data on them. The correct parameter count will be used and the lack of CPU data has no other effect on *refs* operation.

For ECBs, I have made the following decisions:
1. I have assigned a block size of 1 Block equivalent to all ECBs,
2. I have assigned a CPU requirement of 0.00 to ECB18s, i.e. the ECBs built for each channel on an FBM18,
3. I have assigned a CPU requirement of 2.4% to ECB12s, i.e. the "parent" ECB for each ECB18,
4. I have assigned a CPU requirement of 0.5% for the following ECB types: ECB4
5. I do not have the time to determine all of the other I/O vs. I-Only ECBs. I hope this lapse does not cause too much trouble.
6. I have assigned a CPU requirement of 1.3% for the ECB8, the PLB block ECB,
7. I have assigned a CPU requirement of 0.3% to all other ECBs.
8. ECBs in a CP30 are not any faster because Fieldbus is not faster.

2   REFCNT - PEER-TO-PEER REFERENCE TABLE

Detailed help is printed if one invokes the program without any arguments. The help text is also printed by default when the program is run. To suppress the detailed help use the `-h` *option.*

refCnt reads the Master Compound list produced by *refs* and the list of external references also produced by *refs* and creates a table that resembles a road map mileage chart as shown below:

CP Number	CP Name	0	1	2
0	AACP01		10	
1	AACP02	20		11
2	AACP03			

In this example, CP Number 0, i.e. AACP01, sends 10 values to CP Number 1, i.e. AACP02, and receives 20 values from the same CP.

refCnt also builds a table that lists the number of "hard" or "burned-in" connections between CPs. This table lets one count the number of used connections in a CP easily.

3   LEVEL - GRAPH OF CONTROL BLOCK PHASING

Detailed help is printed if one invokes the program without any arguments. The help text is also printed by default when the program is run. To suppress the detailed help use the `-h` *option.*

level reads on-line or off-line workfiles produced by the Integrated Control Configurator and produces a histogram showing the percentage load in each BPC based on either block counts or ED 3648. The workfiles can be for any station type.

4   LOOPS - BLOCK LINKAGE DIAGRAMS (PSEUDO-LOOP DRAWINGS)

Detailed help is printed if one invokes the program without any arguments. The help text is also printed by default when the program is run. To suppress the detailed help use the `-h` *option.*

loops reads on-line or off-line workfiles produced by the Integrated Control Configurator and produces a set of "loop drawings" for each control loop in the workfile. The workfiles can be for any station type.

The "loop drawing" resembles the following:

```
D353:LV3308
MEAS  <-CP13N:LC3308.OUT
        MEAS  <-CP13N:BAJ1341.PNT_4
               MA     <-CP13N:ALOCK.IN_1
        FBK   <-D353:LV3308.BCALCO *
        HOLD  <-CP13N:BAJ1341.BAD_4 *
        INITI <-D353:LV3308.INIT0 *
        BCALCI<-D353:LV3308.BCALCO *
MA      <-CP13N:ALOCK.IN_1
```

This "loop drawing" shows the control loop L3308. The AOUT block LV3308 has its measurement connected to LC3308's output parameter and LC3308 has its measurement connected to the MAIN block BAJ1341.

The "loop drawing" is generated by starting at "terminal blocks," e.g. AOUT, MCOUT, COUT, REALM, etc, and walking the tree of connections backwards from these terminal blocks. Branches on the tree are pruned if the block shown has been printed earlier in the listing. Pruned branches are shown with an "*" after the block name.

This utility is most useful for verifying drawings against the actual control database.

5   MKBLKINDEX - GENERAL PURPOSE INDEXER FOR WORKFILES

Detailed help is printed if one invokes the program with the `-h` *option. The help text is <u>not</u> printed by default when the program is run.*

mkBlkIndex reads a collection of on-line or off-line workfiles produced by the Integrated Control Configurator and produces a file that contains the following fields for each block in the collection:

    1) Block Type,
    2) Workfile name, i.e., CP Name if the .wf is on-line or a renamed off-line workfile,
    3) Compound:Block name for the block, and
    4) Byte offset into the workfile for the block.

This utility is useful for a number of purposes. For example, a simple *grep*, e.g. "`grep LC3308 BlockIndex`," can give the location of a desired block. However, I use it most often as a front-end for other tools, e.g. *mkHistDB*, *docWF*, and *dumpWF*.

To index all workfiles on an AP or PW and produce a sorted index for use with other tools, issue the following command:

```
mkBlkIndex /usr/fox/ciocfg/*/*.wf | sort -o BlockIndex
```

6   DOCWF - GENERAL PURPOSE WORKFILE DOCUMENTER/QUERY TOOL

Detailed help is printed if one invokes the program without any arguments. The help text is also printed by default when the program is run. To suppress the detailed help use the `-h` *option.*

docWF can be used to query either the on-line CP or a workfile. It works in a manner similar to IMPDBU on the Fox 1/A.

The query is express as a file that specifies the "pattern" to match and the values to print. An example of the file is:

```
SELECT
        TYPE=[PID,PIDXE]
PRINT
        PBAND
        INT
        DERIV
        KD
        MODOPT
        INCOPT
```

The indicated indention is required and must be provided by a single tab character. Sample output resembles:

```
D353:LC3308         AACP01              PID                 D353 LEVEL CNTLR
PBAND = 50.000000 INT   =3.000000 DERIV =0.000000 KD    =10.000000
MODOPT=4            INCOPT=1

D354:LC3309         AACP01              PID                 D354 LEVEL CNTLR
PBAND = 50.000000 INT   =3.000000 DERIV =0.000000 KD    =10.000000
MODOPT=4            INCOPT=1
```

Note:

1) In the SELECT clause, the line

        TYPE=[PID,PIDXE]

   specifies that all blocks in the alphabetical range from PID to PIDXE should be selected. A single block type could be specified by

        TYPE=PID

   However, I have not implemented enumerated values, i.e.

        TYPE=PID, PIDE, PIDX, PIDXE

will not work. Use multiple runs instead.

2) The SELECT clause also supports multiple lines, for example,

```
SELECT
        TYPE=AIN
        HLOP=1
PRINT
        HAL
        LAL
```

would list only those AIN blocks with the absolute alarm option set. Multiple lines amount to an "AND."

3) *docWF* supports the following options:

-pc which causes the output to print in columns with the parameter names at the top of each column and
-sc which separates the columns, if -pc is specified, with the character 'c.'

These options are used to export I/A database information to Lotus or dBase. The default separation character is the tab character.

Before using *docWF*, run *mkBlkIndex*, at least once,:

```
mkBlkIndex /usr/fox/ciocfg/*/*.wf | sort -o BlockIndex
```

docWF is run by typing:

```
docWF BlockIndex <docList          -- typically used to print a report
```

      or

```
docWF BlockIndex -pc <docList      -- typically used to export to Lotus
```

where `BlockIndex` was created by *mkBlkIndex* and `docList` is as described above.

Output is sent to stdout by default.

7   DUMPWF - DUMP THE WORKFILE TO TEXT

Detailed help is printed if one invokes the program without any arguments. By default, the help text is not printed during a normal run to force the detailed help use the `-h` option.

dumpWF dumps all parameters of all blocks in the BlockIndex file to a set of files. A file will be created for each block type represented in the BlockIndex file.

The files will be named `<blockType>.txt.`

To run it type:

Before using *docWF,* run *mkBlkIndex*::

        mkBlkIndex /usr/fox/ciocfg/*/*.wf | sort -o BlockIndex

Then, run *dumpWF*:

        dumpWF BlockIndex

8   MKHISTDB - BUILDS SOURCE FILES FOR HISTORIAN BUILDING 9 TOOLS

Detailed help is printed if one invokes the program without any arguments. The help text is not printed by default when the program is run. To force the detailed help use the -h option.

mkHistDB reads a Historian database definition file and extracts all blocks that match the specified definition. For example, I have used the following definition file to extract all PID MEAS parameters, all AIN PNT parameters, all REALM MEAS parameters, and all AOUT block MEAS parameters:

```
AIN          PNT    RO1
AOUT         *MEAS  RI1
PID          MEAS   RI1
PIDE         MEAS   RI1
PIDX         MEAS   RI1
PIDXE        MEAS   RI1
RATIO        MEAS   RI1
REALM        MEAS   RI1
```

Note the reference to the MEAS parameter of the AOUT block. This means that I want to log the connection to the block; rather than the MEAS parameter of the AOUT block. That is, I will log the output of the PID block connected to the AOUT block's MEAS parameter.

The full syntax for this file is:

```
<block type>  <indirection specifier><parameter name> <units set> <delta in % of scale> <period>
```

where

   <block type> is the name of an I/A block type, e.g. AIN, PID, RATIO, etc,

   <indirection specifier> may be one of (blank, '*', or '+').

      A blank means that the parameter specified in the next field is logged
         directly, i.e. the logged reference for

            AIN PNT RO1

         will be something like

            CMPD:AIN.PNT

      '*' means that whatever is connected to the parameter specified is logged
         and if nothing is connected then nothing is logged, i.e. the logged
         reference for

            AOUT *MEAS RI1

         will be something like

CMPD:PID.OUT

'+' means that if the parameter is connected the connection will be logged otherwise the parameter is logged directly. Thus for
AOUT *MEAS RI1

CMPD:PID.OUT

will be logged if the AOUT block is connected to a PID block, but

CMPD:AOUT.MEAS

will be logged if it is not connected.

<units set> is the name of the set of items that defines the high scale, low scale, change delta, and engineering units for the parameter. RI1 means that the variable is associated with HSCI1, LSCI1, DELTI1, and EI1. RO1 means that the variable is associated wit HSCO1, LSCO1, DELTO1, and EO1.
is the name of a valid parameter for the specified block type,
<period> is the minimum time between the writes to the collection group, the allowed values are 2, 4, 10, and 20 seconds. Twenty seconds is the default, and
<delta in % of Scale> specifies the amount of change delta required to generate a write for this parameter. The default value is the change delta % specified in the block multiplied by the span specified in the block.

Before using *docWF,* run *mkBlkIndex*::

```
mkBlkIndex /usr/fox/ciocfg/*/*.wf | sort -o BlockIndex
```

A typical invocation of this program is :

```
sort histList | mkHistDB BlockIndex > APHistDB
```

The file APHistDB would be fed to Steve Johnson's program, *add_histdb*.

There is a script in cpUtil/exe called *mkAPHistDB* that can be used as an example of how to run the program. In cpUtil/data, you will find a sample histList that can be used as a basis for extracting the Historian Database Configuration Definition data required by Steve Johnson's program.

Notes on the Solaris Version

The Solaris Version will build output for either Steve Johnson's AP-20 Historian Tools or the AP-50 Historian builder, /usr/fox/hstorian/bin/cfgpts. The options are '-ap20' and '-ap50'. Documentation for *cfgpts* can be found in /usr/fox/hstorian/bin/cfgpts.doc.

9   MKLISTCHECK - SYNTAX CHECKER FOR MKHISTDB

Detailed help is printed if one invokes the program with the `-h` *option. The help text is <u>not</u> printed by default when the program is run.*

mkListCheck checks the syntax of a histList file, i.e. a Historian Database Configuration Definition file.

A typical invocation is

```
mkListCheck <histList &&
      (sort histList | mkHistDB BlockIndex >APHistDB)
```

10   MKAPHISTDB - BUILD A HISTORIAN DATABASE CONFIGURATION DEFINITION FILE

mkAPHistDB is a script that builds the Historian Database Configuration Definition file. The script says:

```
mkBlkIndex /usr/fox/ciocfg/*/*.wf | sort -o BlockIndex
mkListCheck <histList &&
        (sort histList | mkHistDB BlockIndex >APHistDB)
```

11   MKEXTREFLIST - MAKES AN EXTERNAL REFERENCE LIST FOR A PARTICULAR APPLICATION

refs and *refCnt* are nice, but they do not cover any application programs that a user has written.

Given a list of the tags in those programs, *mkExtRefList* will generate such a list. The output of *mkExtRefList* is compatible with the `-prefs` options of *refs*.

Try this as an example:

```
mkExtRefList a1hs01 A1AP01 /opt/fox/hstorian/sample/__tdir cmpds >>sorted.refs
```

Where `sorted.refs` is produced by *refs*.

Type:

```
mkExtRefList
```

for more information.

12   CHECKING THE CONTROL STATION CONFIGURATION ON-LINE

12.1 OVERVIEW

There are three locations where information about the contents of a Control Station are kept:

1) The Control Station's RAM,
2) The Integrated Control Configurator's (ICC) Workfile, and
3) The Control Station's Checkpoint file.

In addition, the Compound Summary Access (CSA) database keeps a list of CS Compounds and Blocks.

As described in the I/A Series documentation, the CS's RAM holds the actual executing code, the checkpoint file is a snapshot of the CS's RAM at a particular time, and Workfile is "source code" for the control block parameters, i.e., it is the file maintained by the ICC.

During normal plant operations, it is common for the contents of these files to "drift," i.e., for the same parameter in a Workfile, Checkpoint File, and CS RAM to have a different value. Such "drift" is possible only for control block parameters that are "settable" and, therefore, the biggest cause of "drift" are the changes made by the operator to alarm limits and setpoints.

Drift of this type can lead to confusion and ambiguity about the state of the system. On more than one occasion, the checkpoint file that was restored during a reboot did not match the customer's expectations. Therefore, FoxCTS provides a utility for examining the degree of drift among the files, `CompareCSFiles`.

Please note that it is not possible, under normal operation, for the three files to have different compounds, blocks, sequence block code, or ladder logic. Differences of these types can only arise if a system failure has occurred that required the restoration of one or more of these files from old tapes. These utilities do not address the problems caused by such failures.

12.2 THE EVALUATION TOOLS

Two tools are provided to assist in the evaluation of Control Station Drift:

1)	CompareCSFiles and
2)	ReportChanges.

The first tool grabs the current checkpoint, RAM image, and workfile for one or more Control Stations and stores them. The second tool examines the stored files and reports variations among the files.

CompareCSFiles retrieves:

3)	A workfile by using *iccprt* to generate an ASCII version of the file,
4)	A checkpoint file by copying the file to a local directory from the station's host, and
5)	A copy of the RAM by archiving the current checkpoint file, performing a checkpoint, archiving the new file, and restoring the original file.

12.3 MAN PAGES

12.3.1 CompareCSFiles

NAME

CompareCSFiles – Archives copies of the three versions (Workfile, RAM, and Checkpoint
file) of a Control Station's database

SYNOPSIS

```
CompareCSFiles [<cpLbug>...] [-h]
```

```
where
      <cpLbug> is an optional Control Station letterbug
      -h is a help flag used to get this synopsis
```

DESCRIPTION

CompareCSFiles obtains copies of the three Control Station databases to an archive directory for
processing by ReportChanges.

CompareCSFiles accepts an optional list of Control Station Letterbugs whose files should be
archived.

If such a list is not present, CompareCSFiles will check in the data directory for a file called CSs.
This file should contain a set of CS letterbugs with one name per line.

If such a file is not present, CompareCSFiles will use the contents of the file /etc/cplns which
is generated by the I/A Series Configuration process.

In general, this script installed in /opt/foxind/cms/CompareCSFiles. The archived files are
stored in /opt/foxind/cms/extras/CompareCSFiles/data/<csName>.

12.3.2 ReportChanges

NAME

>   ReportChanges – Examines the archives created by `CompareCSFiles` and reports any
>   parameter value mismatches.

SYNOPSIS

>   **ReportChanges <cpLbug> [...] [-h]**
>
>   where
>   <cpLbug> is a Control Station letterbug
>   -h is a help flag used to get this synopsis

DESCRIPTION

`ReportChanges` examines the files obtained by `CompareCSFiles` and generates a mismatch report
for each station on it command line.

In general, this script installed in `/opt/foxind/cms/CompareCSFiles`. The archived files are
stored in `/opt/foxind/cms/CompareCSFiles/data/<csName>`.

`ReportChanges` may be run at any time.

12.3.3 stationInfo

NAME

       stationInfo – Data file that contains a list of all station type related information required by
           CompareCSFiles.

SYNOPSIS

```
Pre-Installation:
     /opt/foxind/cms_install/extras/CompareCSFiles/data/stationInfo
Run-Time:
     /opt/foxind/cms/CompareCSFiles/data/stationInfo
```

DESCRIPTION

This file contains one line per supported station type. It may or may not have all of the lines
required to support any particular I/A Series Station. However, a knowledgeable user can edit the
file to add support for missing station types.

The file consists of comment lines and data lines.

Comment lines have a # as their first character.

Data lines have four fields as shown in the following table:

Field Name	Description	Example
StationMnemonic	Shorthand name for a type of Control Station	CP10, CP30, ABS, MI
StationType	Numeric code used by the I/A Series System Configuration software to represent a particular station	202 for a CP10
OSImage	The Operating System image used for the particular Control Station	OS1UC for a CP10
OSMapFile	Linkage editor map file required by the tool dbvu	OS1UC.mp2 for a CP10

New station types can be added by obtaining the required information and adding it to the Station
Information file.

12.3.4 CSs

NAME

CSs – Control Station Names file.

SYNOPSIS

```
Pre-Installation:
      /opt/foxind/cms_install/extras/CompareCSFiles/data/CSs
Run-Time:
      /opt/foxind/cms/CompareCSFiles/data/CSs
```

DESCRIPTION

This file is read by `CompareCSFiles` if `CompareCSFiles` is started without any arguments.

It should contain the names (letterbugs) of the Control Stations whose data is to be archived for later differences reporting.

A typical file would resemble the contents of `/etc/cplns` which holds all Control Station letterbugs.

12.4 EXAMPLE USAGE

In the following example, notes are have been added in italics intermixed with the standard output.

```
1AW51A# CompareCSFiles 1CP40C && ReportChanges 1CP40C
```
This invocation tells CompareCSFiles to get the information on 1CP40C. ReportChanges will run only after CompareCSFiles successfully completes.

```
CP10 201 OS1UC OS1UC.mp2 :  :
CP30 203 OS1C30 OS1C30.map :  :
ABST 204 OS1AB4 OS1AB4.map :  :
CP40 205 OS1C40 OS1C40.map : 1CP40C :
AB30 2003 OS1AB3 abgw30_3.map :  :
MG30 3002 OS1MG3 mggw30_3.map :  :
```
Note that the script has listed each station type that it understands and the stations of that type that it will process. The stations are between the colons.

```
+ rexec 1AW51C /tmp/MkDBLists 1AW51A 1CP40C OS1C40.map OS1C40
```
CompareCSFiles is gathering the required data from the host of 1CP40C, i.e., 1AW51C.

```
        Processing 1CP40C with dbvu (checkpoint file)...
        Grabbing current data...
        Processing 1CP40C with dbvu (current data)...
        Processing 1CP40C with iccprt...
```
As it runs, the script provides a progress report. This step is not very fast since it requires a checkpoint (grabbing current data) along with two executions of dbvu *and one of* iccprt.

```
Comparison report for 1CP40C
Mon Nov 2 19:33:41 GMT 1998
```
This header is provided to show when the report was run. It does not say when the data was gathered.

```
Notes:
   R: Value as read from the RAM of the Control Station
   C: Value as read from the checkpoint file of the CS
   W: Value as read from workfile of the CS
   B: Value is the same in the CS's RAM and checkpoint file
```
This key provides the information required to read the report. ReportChanges only shows mismatches to minimize the output that the user needs to review. Therefore, if all three sources match, the line is not reported. If the RAM and Checkpoint files match, a 'B' is used to represent their value.

```
1CP40C_ECB
      INITON
         B: 1
         W: 2
1CP40C_STA
```

```
        INITON
            B: 1
            W: 2
    STATION
        CKPTIM
            R: 1998-11-02 19:32:04
            C: 1998-11-02 19:19:28
            W:
AMI_TEST
    AIN_OOR
        INHALM
            B: 0x1000
            W: 0x0000
```
Note that all packed Boolean and packed long parameters are printed in hexadecimal.

```
    IND1
        ACTIVE
            B: F
            W: T
```
Note that all Boolean parameters are printed as T for true (set) or F for false (reset).

```
    MSGGR1
            B: 1
            W: 0
CDUVDUOP
    COUNTDOWN
        II0001
            B: 120
            W: 0
        II0002
            B: 300
            W: 0
        RI0001
            B: 120.0
            W: 0.0
        SN0001
            B: CDUVDU
            W:
        SN0002
            B: 1CP40C
            W:
        SN0003
            B: 2
            W:
        SN0004
            B: 1CP40C_STA:STATION.RESVL2
            W:
        SN0005
            B: 1CP40C_STA:STATION.TIMVL2
            W:
        SN0006
            B: CDUVDU:DMCOP.CTLINT
            W:
```

```
FALLBACK
   BI0002
      B: T
      W: :MASTERWDT.BI0001
MASTERWDT
   BI0001
      B: T
      W: F
   BI0002
      B: T
      W: F
   BI0003
      B: T
      W: F
   SN0001
      B: 1CP40C_STA:STATION.FLBRQ2
      W:
   SN0002
      B: NONE
      W:
   SN0003
      B: NONE
      W:
   SN0004
      B: NONE
      W:
   SN0005
      B: NONE
      W:
   SN0006
      B: NONE
      W:
   SN0007
      B: NONE
      W:
   SN0008
      B: NONE
      W:
   SN0009
      B: NONE
      W:
   SN0010
      B: NONE
      W:
 PULSE
   STEP01
      B: IN BI01 I
      W: IN BI01 INPUT DMCON
```

CALC block RAM and checkpoint files do not report the entire comment field so there will generally be many mismatches on well-documented CALC blocks.

```
   STEP04
      B: OUT BO01 P
```

```
      W: OUT BO01 PUT OUT OSP ~DMCON
   STEP05
      B: IN BI01 I
      W: IN BI01 INPUT DMCON
   STEP07
      B: OUT BO02 P
      W: OUT BO02 PUT OUT OSP DMCON
   STEP08
      B: IN BI02 I
      W: IN BI02 INPUT DMCXIT
   STEP10
      B: OUT BO03 P
      W: OUT BO03 PUT OUT OSP DMCXIT
   STEP11
      B: IN BI03 I
      W: IN BI03 INPUT WATCHDOG (T=GOOD)
   STEP14
      B: OUT BO04 P
      W: OUT BO04 PUT OUT OSP ~WDT
   STEP20
      B: IN II01 I
      W: IN II01 INPUT CTLINT
   STEP21
      B: BIZ 24 I
      W: BIZ 24 IF ZERO GO AROUND
   STEP22
      B: RCL II01 I
      W: RCL II01 INPUT CTLINT AND CLEAR
   STEP23
      B: OUT IO01 P
      W: OUT IO01 PUT CTLINT ON OUTPUT
   STEP24
      B: IN IO01 P
      W: IN IO01 PUT CTLINT ON STACK
   STEP25
      B: BIZ 27 I
      W: BIZ 27 IF ZERO GO AROUND
   STEP26
      B: DEC IO01 D
      W: DEC IO01 DECREMENT IO01
WATCHDOG
   II0001
      B: 120
      W: 0
   RI0002
      B: 300.0
      W: 0.0
   SN0001
      B: CDUVDU
      W: DMCPLUSD
   SN0002
      B: CDUVDU:DMCOP.CNTDWN
      W:
   SN0003
```

```
            B: CDUVDU:DMCOP.ONREQ
            W:
CRUDE_FEED1
   11TIC102
      BCALCI
            B: 104.0
            W: 0.0
DMCPLUS_COL1
   WATCHDOG
      BI0002
            B: T
            W: F
      II0001
            B: 1
            W: 0
      RI0003
            B: 12.0
            W: 0.0
      RI0004
            B: 100.0
            W: 0.0
      RI0005
            B: 1.0
            W: 0.0
DMCPLUS_COL2
      II0001
            B: 1
            W: 0
      RI0002
            B: 150.0
            W: 0.0
      RI0003
            B: 5.0
            W: 0.0
      RI0004
            B: 100.0
            W: 0.0
      RI0005
            B: 1.0
            W: 0.0
DMCPLUS_COL3
      RI0003
            B: 5.0
            W: 0.0
      RI0004
            B: 10000.0
            W: 0.0
      RI0005
            B: 1.0
            W: 0.0
DMCPLUS_COL4
      BI0002
            B: T
            W: F
```

```
RI0003
   B: 255.0
   W: 0.0
RI0004
   B: 10000.0
   W: 0.0
RI0005
   B: 1.0
   W: 0.0
DMC_COLLECT1
   II0001
      B: 0
      W: 10
   II0002
      B: 10000
      W: 0
   RI0004
      B: 0.0
      W: :WATCHDOG.BI0001
   RI0006
      B: 17.02
      W: 0.0
DMC_COLLECT2
   II0001
      B: 0
      W: 10
   II0002
      B: 10000
      W: 0
   RI0004
      B: 0.0
      W: :WATCHDOG.BI0001
   RI0006
      B: 13.26
      W: 0.0
DMC_COLLECT3
   II0001
      B: 0
      W: 10
   II0002
      B: 10000
      W: 0
   RI0004
      B: 0.0
      W: :WATCHDOG.BI0001
   RI0006
      B: 13.29
      W: 0.0
DMC_COLLECT4
   II0002
      B: 10000
      W: 0
   RI0004
      B: 0.0
```

```
        W: :WATCHDOG.BI0001
NEWDEMOOP
   COUNTDOWN
      II0001
         B: 60
         W: 0
      II0002
         B: 150
         W: 0
      RI0001
         B: 60.0
         W: 0.0
      SN0001
         B: NEWDEMO
         W:
      SN0002
         B: 1CP30A
         W:
      SN0003
         B: 4
         W:
      SN0004
         B: 1CP30A_STA:STATION.RESVL4
         W:
      SN0005
         B: 1CP30A_STA:STATION.TIMVL4
         W:
      SN0006
         B: NEWDEMO:DMCOP.CTLINT
         W:
   FALLBACK
      BI0002
         B: F
         W: :MASTERWDT.BI0001
   MASTERWDT
      SN0001
         B: 1CP30A_STA:STATION.FLBRQ4
         W:
      SN0002
         B: NONE
         W:
      SN0003
         B: NONE
         W:
      SN0004
         B: NONE
         W:
      SN0005
         B: NONE
         W:
      SN0006
         B: NONE
         W:
      SN0007
```

```
         B: NONE
         W:
     SN0008
         B: NONE
         W:
     SN0009
         B: NONE
         W:
     SN0010
         B: NONE
         W:
 PULSE
     STEP01
         B: IN BI01 I
         W: IN BI01 INPUT DMCON
     STEP04
         B: OUT BO01 P
         W: OUT BO01 PUT OUT OSP ~DMCON
     STEP05
         B: IN BI01 I
         W: IN BI01 INPUT DMCON
     STEP07
         B: OUT BO02 P
         W: OUT BO02 PUT OUT OSP DMCON
     STEP08
         B: IN BI02 I
         W: IN BI02 INPUT DMCXIT
     STEP10
         B: OUT BO03 P
         W: OUT BO03 PUT OUT OSP DMCXIT
     STEP11
         B: IN BI03 I
         W: IN BI03 INPUT WATCHDOG (T=GOOD)
     STEP14
         B: OUT BO04 P
         W: OUT BO04 PUT OUT OSP ~WDT
     STEP20
         B: IN II01 I
         W: IN II01 INPUT CTLINT
     STEP21
         B: BIZ 24 I
         W: BIZ 24 IF ZERO GO AROUND
     STEP22
         B: RCL II01 I
         W: RCL II01 INPUT CTLINT AND CLEAR
     STEP23
         B: OUT IO01 P
         W: OUT IO01 PUT CTLINT ON OUTPUT
     STEP24
         B: IN IO01 P
         W: IN IO01 PUT CTLINT ON STACK
     STEP25
         B: BIZ 27 I
         W: BIZ 27 IF ZERO GO AROUND
```

```
    STEP26
        B: DEC IO01 D
        W: DEC IO01 DECREMENT IO01
WATCHDOG
    II0001
        B: 60
        W: 0
    SN0002
        B: DMCPLUSD:DMCOP.CNTDWN
        W:
    SN0003
        B: DMCPLUSD:DMCOP.DMCON
        W:
```