Honeywell

Lesson 2

4410S

TotalPlant

Copyright, Notices, and Trademarks

Revision 03 – September 28, 1999

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

FLEXTRAINING and **TotalPlant** are US registered trademarks of Honeywell Inc.

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

Other brand and product names are trademarks of their respective owners.

Honeywell
Industrial Automation and Control
Automation College
2820 West Kelton Lane
Phoenix, AZ 85053

1-800-852-3211

Lesson 2 - Table of Contents

Introduction

LESSON OVERVIEW

Lesson Objectives

Upon completion of this lesson, you will have an understanding of

- CL Block header structures
- System and Custom Parameter Lists
- Background CL scheduling/operations
- Process Point Special usage

Lesson Outline

INTRODUCTION

LESSON OVERVIEW

CL Block Program

BOUND DATA POINT
INSERTION POINTS
CL EXECUTION ORDER
CONDITIONAL STATEMENTS
ACCESS PRIVILEGE

CL "RUNTIME" ERRORS\l 2

Parameter List

SYSTEM
CUSTOM

Ordinal Values

SELF DEFINED ENUMERATIONS

Background CL

SCHEDULING/OPERATION
PRIORITIES
DELAYS

Process Point Special (PPS)

USAGE

CL Block Program

BOUND DATA POINT

- A CL program is defined by a BLOCK name. This name must be unique to an Application Module. This same name must be used to denote the end of the CL program.

Example:

```
BLOCK ABC (POINT ...; AT...)
-- body of program
END ABC
```

- A CL Block must be bound to an AM point for scheduling. This point must be defined specifically or generically in the Block header.

Example:

```
BLOCK ABC (POINT A100; AT ....)

BLOCK ABC (GENERIC ..; AT.....)
```

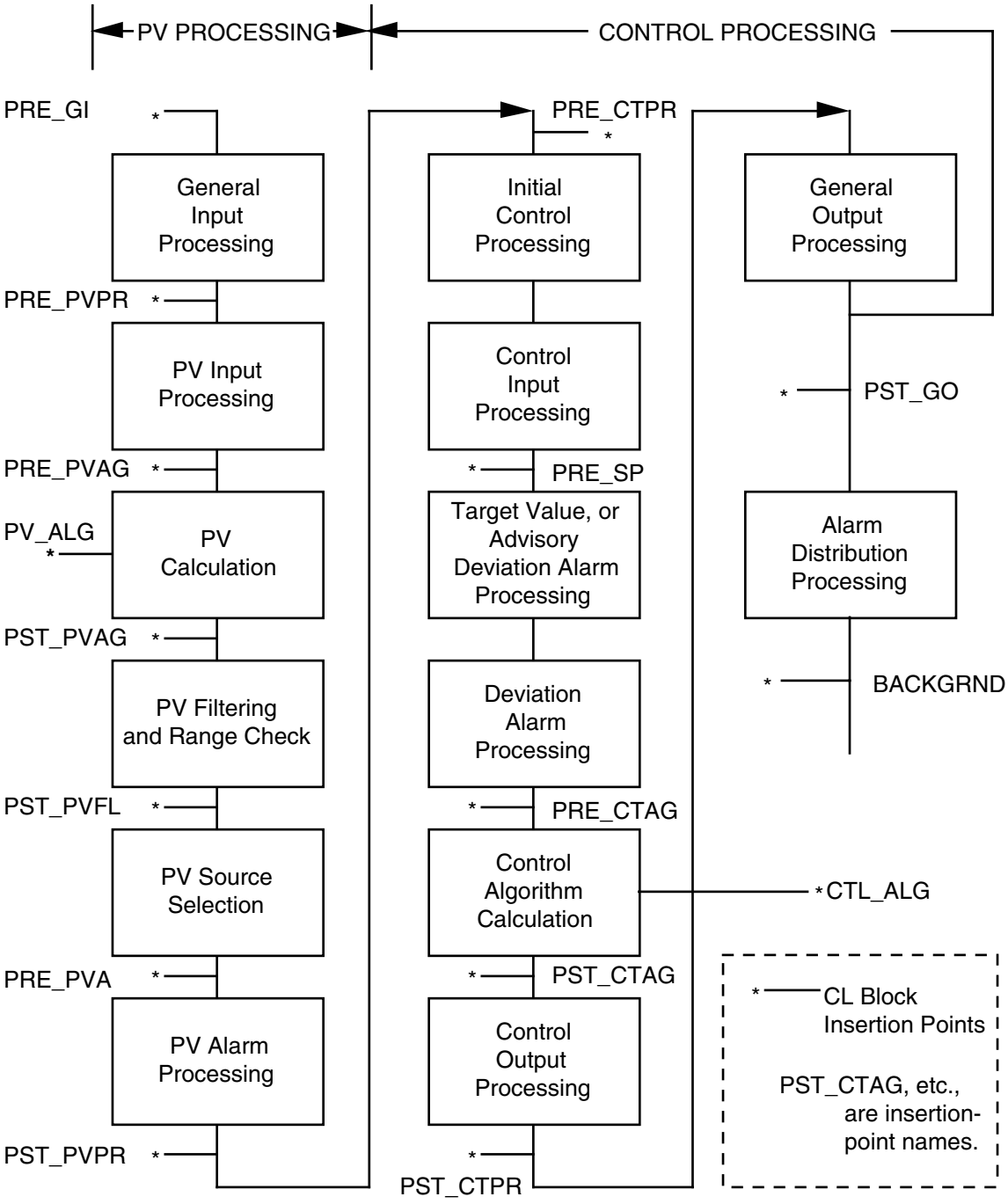
- The Bound Data Point may be either a Regulatory, Custom, or Switch point. A CL Block must specify at what point in the execution of the Bound Data Point the CL Block should be executed. A Regulatory point has 16 pre-defined insertion points.

Example:

```
BLOCK ABC (POINT A100; AT PV_ALG)
```

CL Block Program

INSERTION POINTS



CL Block Program

INSERTION POINTS

- Custom and Switch points have only two insertion points: GENERAL AND BACKGRND. (Note that BACKGRND is not misspelled. It cannot exceed eight characters.)

Example:

```
BLOCK ABC (POINT A100; AT   PV_ALG)    -- Foreground
BLOCK ABC (POINT A100; AT  CTL_ALG)    -- Foreground
BLOCK ABC (POINT A100; AT  GENERAL)    -- Foreground
BLOCK ABC (POINT A100; AT BACKGRND)    -- Background
```

- CL programs inserted at a data point's BACKGRND insertion point are referred to as Background CL programs; programs inserted at any other insertion point are referred to as Foreground CL programs.

- Foreground CL programs run as subroutines, and thus should execute within a single processing cycle for the bound data point.

  Background CL programs run in AM "freetime" and can run over many processing cycles.

CL Block Program

CL EXECUTION ORDER

- CL Program scheduling is based on the Bound Data Point execution period (Parameter PERIOD). The selections are: 1, 2, 5, 10 etc. seconds, up to 24 hours, or NoPeriod.

- A maximum of 255 CL Programs (BLOCKS) may be linked to an AM point.

- Once the Bound Data Point begins execution, if it has multiple CL programs linked, the CL Programs execute in the order of insertion.

Example: Regulatory Point

```
Block ABC (Point A100; at Pre_Gi)       -- executes first
Block DEF (Point A100; at Pre_Pvpr)     -- executes second
Block GHI (Point A100; at Pre_Pvag)     -- executes third
                                        -- and so on to
Block XYZ (Point A100; at Backgrnd)     -- executes last
```

Example: Custom Point or Switch Point

```
Block ABC (Point A100; at General)       -- executes first
Block XYZ (Point A100; at Backgrnd)     -- executes last
```

CL Block Program

CL EXECUTION ORDER

- If several CL Program Blocks are inserted at the same insertion point of a Bound Data Point, they will normally execute in the order that they were linked.

Example:

```
Block ABC (Point A100; at Pre_Pvag)
Block DEF (Point A100; at Pre_Pvag)
Block GHI (Point A100; at Pre_Pvag)
```

- If, however, a linked CL Block is unlinked from a point for modification and then recompiled and relinked, it may change the link order which will affect the order of execution .

Example: (Block DEF was modified and re-linked)

```
Block ABC (Point A100; at Pre_Pvag)
Block GHI (Point A100; at Pre_Pvag)
Block DEF (Point A100; at Pre_Pvag)
```

CL Block Program

CL EXECUTION ORDER

- In order to ensure that modified CL Block Programs are executed in the correct order, an entry may be made in the Block header.

Example:

```
Block ABC  (Point A100; at General (1))
Block DEF  (Point A100; at General (2))
Block GHI  (Point A100; at General (3))
```

- In the above example, if Block ABC should be unlinked, modified, recompiled, and relinked, it would relink as the first Block to be executed at insertion point General.

- This link order would also be indicated on the Bound Data Point's CL detail page.

Example:

```
                                                    18 Mar 93 10:06:09  6

A100                              39 UNIT 39                 CL BLOCK PAGE   1
                  CL BLOCK INFORMATION - ERRSUM NOERROR

BLOCK BLOCK    ACTIVITY INSERTN  INSERTN BLOCK    LOC   BLOCK TIME
INDEX NAME               POINT   ORDER   ERROR
----- -------- -------- ------- ------- -------- ----- --------------
    1 ABC        ACTIVE GENERAL       1 NOERROR      0 18Mar93 10:03
    2 DEF        ACTIVE GENERAL       2 NOERROR      0 18Mar93 10:03
    3 GHI        ACTIVE GENERAL       3 NOERROR      0 18Mar93 10:03
```

CL Block Program

CONDITIONAL STATEMENTS

- CL Block Programs may be specified to execute only if certain conditions are true. These conditions are included in the BlockHeader.

Example:

```
Block ABC (Point A100; at Pre_Pvag(1); when pv>50)
```

-- or

```
Block ABC (Point A100; at Pre_Pvag(1); when B100.sp<20
&    and pv<>sp)

External B100  -- must be declared external when used
               -- in condition statement
--
--
End ABC
```

CL Block Program

ACCESS PRIVILEGE

- CL Blocks have defined access privileges. The access key of **Contin_Ctrl** is the default and permits full privileges but, must adhere to system enforced restrictions.

Example:

```
Block ABC (Point EC100; at xxxxxx;)
External A100       -- A100 is a controller point
Set A100.sp = 50     -- Valid  if A100.mode is in CAS
Set A100.op = 45     -- Valid if A100.mode in CAS, algo is
                          -- DDC, and RCASENB contains On
```

- The other access key is **Program**, which inhibits stores to certain parameters unless the value of the target point's Modattr parameter is Program.

Example:

```
Block ABC (Point EC100; at xxxxxx; Access Program)
External A100       -- A100 is a controller point
Set A100.sp = 50    -- Valid  if A100.mode is in Auto and
                    -- Modattr is in Program
Set A100.op = 45     -- Valid if A100.mode is in Man and
                    -- Modattr is in Program
```

- The parameters affected by Access Privilege are:

```
MODE
SP          OP
RATIO       BIAS
```

- Stores to Ratio and Bias, of the PID Ratio/Bias Algorithms, require the Modattr Parameter to be **Program** for CL stores.

CL Block Program

CL "RUNTIME" ERRORS

- There are three classes of Runtime errors; **CL Error Conditions**, **CL Failures**, and **Minor Errors**.

- **CL Error Conditions** result from CL programming errors ("bugs") or other errors that require CL coding changes for successful recovery. The error conditions are:

  CNFERR (Configuration Error): referenced point does not exist, parameter data type mismatch, indirect reference thru a null data-point, etc.

  KEYLEVEL (Key-Level Access Error): an attempted store could not take place because of a key-level restriction.

  PROGERR (Program Error): CL execution reached some illogical condition that indicates an error in the CL program.

  RANGE (Range Error): An attempt was made to store a value outside the fixed range of a parameter that has a fixed value range.

  ARRAYLIM (Array Limit): An attempt was made to access outside the bounds of an array.

  BRANCHV (Backward Branch Violation): Legal number of backward branches has been exceeded. There is no backward branch limit for Background programs.

  ARITHERR (Arithmetic Error): An attempt was made to store an infinite number of seconds into a time variable or time parameter.

CL Block Program

CL "RUNTIME" ERRORS

- **CL Failure Conditions** result from equipment failures, such as a sensor failure, a box, module, or gateway failure, or a communication failure. Also included in this class is an explicit abort from execution of a CL Abort statement. The error conditions are:

  ABORT (Abort Statement): Execution of an Abort statement.

  COMABORT (Communications error abort): Attempted use of a parameter, other than type number, that had a communication error on the attempt to fetch the value.

  BADVALST (Bad value store): Attempt to store a bad value without using the ALLOW_BAD or the SET_BAD subroutine, or the use of a bad value in making a decision.

- **Minor Errors** result from conditions in the process or in the control strategies, such as modes that preclude stores or limits clamping the stored values. No alarms or aborts occur. The error conditions are:

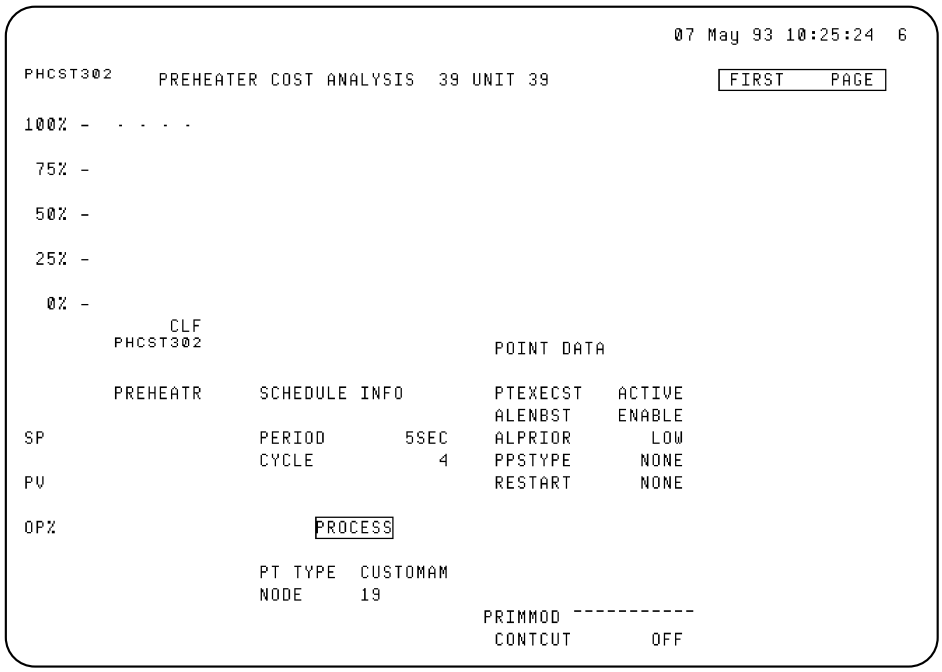
  COMMERR (Communications error): A network fetch or store failed because of a communications error or because the device accessed was off line.

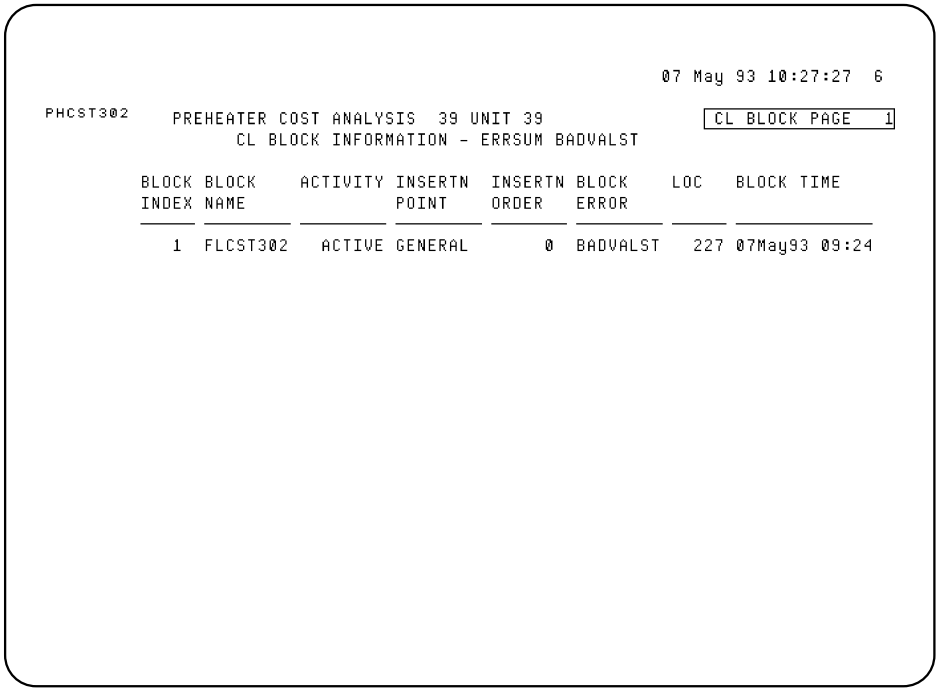
  RIGHTS (Rights error): A temporary restriction prevents the storing of a value. Examples are MODE and MODATTR restrictions on stores to SP, and PVSOURCE restrictions on stores to PV.

  LIMVIOL (Limit Violation): An attempt to store a value that exceeds SP Limits or OP Limits.

CL Block Program

CL "RUNTIME" ERRORS





CL Block Program

CL "RUNTIME" ERRORS

```
                                                        07 May 93 10:31:18  6
                        Source File :  NET>S302>PACK302.CL
                                              PACK302.LS  LINE 23
?
 CL V40.31     PACK302                          05/07/93 09:24:27:9111     Page


 Line     Loc  Text

    1          PACKAGE
    2
    3          ENUMERATION DESU302 = DOL_DAY/DOL_HR/DOL_MIN/DOL_BBL/DOL_GAL
    4
    5          CUSTOM
    6
    7           PARAMETER UNITS: DESU302
    8           ACCESS OPERATOR
    9
   10           PARAMETER FUELCOST
   11           EU "DOL/KSCF"
   12           VALUE 1.0
   13
   14           PARAMETER HTR_COST
   15           ACCESS PROGRAM
   16
   17          END CUSTOM
 F1    F2     F3    F4     F5    F6     F7      F8
QUIT  BLOCK  JUMP  MACRO  EDIT   SET   FILE   SBUFFER
```

```
                                                        07 May 93 10:29:41  6
                        Source File :  NET>S302>PACK302.CL
                                              PACK302.LS  LINE 43
   15           ACCESS PROGRAM
   16
   17          END CUSTOM
   18
   19          BLOCK FLCST302 (POINT PHCST302; AT GENERAL)
   20
   21            LOCAL EF,PF  -- EU FACTOR AND PRODUCT FLOW
   22            EXTERNAL  FC71302,FC70302
   23
   24      13    IF UNITS=DOL_DAY THEN (SET EF=24;SET PF=1)
   25      44     ELSE IF UNITS=DOL_HR THEN (SET EF=1;SET PF=1)
   26      76     ELSE IF UNITS=DOL_MIN THEN (SET EF=0.0167;SET PF=1)
   27     108     ELSE IF UNITS=DOL_BBL THEN (SET EF=0.024;SET PF=FC70302.PV)
   28     152     ELSE (SET EF=0.000571;SET PF=FC70302.PV)
   29
   30     174    SET HTR_COST =FUELCOST*FC71302.PV*EF/PF
   31
   32          END FLCST302
   33
   34          END PACKAGE

*******  No errors detected
 F1    F2     F3    F4     F5    F6     F7      F8
QUIT  BLOCK  JUMP  MACRO  EDIT   SET   FILE   SBUFFER
```

Parameter List

SYSTEM

- A Parameter List is required when creating a Custom Data Segment parameter of the type ENTITY.

Example:

```
Custom
     Parameter Tin : $Reg_Ctl
End Custom
```

- A Parameter List may also be used to declare the parameters that will be used in a CL Block bound to a Generic point.

Example:

```
Block ABC ( Generic $Reg_Ctl; at xxxxxxx)
     if Pv > 50 and Mode = Man then ..........

         -- the alternative would be --

Block ABC ( Generic; at xxxxxxxxx)
parameter Pv
parameter Mode:Mode
     if Pv > 50 and Mode = Man then ..........
```

- **$Reg_Ctl** is the only usable **System** Parameter List.

Parameter List

SYSTEM

$REG_CTL Parameters and Types

Parameter Name	Parameter Type	Parameter Name	Parameter Type
ARWDI	ENM: POLARITY	INITMAN	LOGICAL
ARWNET	ENM: WINDUP	INITREQ [1.. 4]	LOGICAL
ARWOP	ENM: WINDUP	INITTYPE	ENM: INITTYPE
ASP	NUMBER	INITVAL	NUMBER
ASPP	NUMBER	K	NUMBER
AV	NUMBER	K1	NUMBER
AVCOUNTS	NUMBER	K2	NUMBER
AVDEV1FL	LOGICAL	K3	NUMBER
AVDEV2FL	LOGICAL	K4	NUMBER
AVP	NUMBER	KEXT	NUMBER
AVTV	NUMBER	KFF	NUMBER
AVTVFL	LOGICAL	KGAP	NUMBER
B	NUMBER	KLIN	NUMBER
B1	NUMBER	KNL	NUMBER
B2	NUMBER	LASTPV	NUMBER
B3	NUMBER	MODE	ENM: MODE
B4	NUMBER	NAME	STRING
BADCTLFL	LOGICAL	NMODE	ENM: MODE
BADPVFL	LOGICAL	OFFNDIAK	LOGICAL
BFF	NUMBER	OFFNDIRQ	LOGICAL
BIAS	NUMBER	OP	NUMBER
BYPASS	LOGICAL	OPEU	NUMBER
CASREQ	ENM: CASREQ	OPHIFL	LOGICAL
CIACCSTS	ENM: PASTATUS	OPHILM	NUMBER
CIACTSTS	ENM: IOACTSTS	OPLOFL	LOGICAL
COACCSTS	ENM: PASTATUS	OPLOLM	NUMBER
COACTSTS [1.. 8]	ENM: IOACTSTS	OPMCHLM	NUMBER
COMMAND	ENM: COMMAND	OPROCLM	NUMBER
CONTCUT	LOGICAL	ORFBSEC	NUMBER
CTRLINIT	LOGICAL	PATHIND	ENM: PATHIND
CV	NUMBER	PIACCSTS	ENM: PASTATUS
CVEUHI	NUMBER	PIACTSTS	ENM: IOACTSTS
CVEULO	NUMBER	PPS	LOGICAL
CVTYPE	ENM: CVTYPE	PPSREQ	ENM: PPSTYPE
DEVHIFL	LOGICAL	PREVMODE	ENM: MODE
DEVLOFL	LOGICAL	PTDESC	STRING
ESWAUTO	LOGICAL	PTEXECST	ENM: PTEXECST
ESWCAS	LOGICAL	PTINAL	LOGICAL
ESWMAN	LOGICAL	PTORST	ENM: ORSTATUS
FSELIN	ENM: PINP	PV	NUMBER
GAPHI	NUMBER	PVAUTO	NUMBER
GAPLO	NUMBER	PVAUTOST	ENM: PVVALST
GIACCSTS	ENM: PASTATUS	PVCALC	NUMBER

Parameter List

CUSTOM

- A Custom Parameter List may be created and used when there is a need to declare Custom Parameters that are not found in the System Parameter List $Reg_Ctl.

Example:

```
Param_List My_list
   Parameter Pv, Op  : Logical
   Parameter Message : String
   Parameter Specvals: Array (1..10)
   Parameter Backup  : $Reg_Ctl
End My_list

Block ABC ( Generic My_list; at xxxxxxx)
   if Pv = Op and Specvals(3) = 25 then (set Backup.Sp =
55;
&  set Message = " This is a special run ")
```

- A Custom Parameter List may also be used in creating Custom Parameters of the type Entity in a Custom Data Segment.

Example:

```
Custom
   Parameter Tin : My_List
End Custom
```

- Only one Parameter List name may be used per declaration statement.

Example:

```
Custom
   Parameter Tin : My_List
   Parameter Tout: $Reg_Ctl
   Parameter Temp: $Reg_Ctl, My_List  -- not valid
End Custom
```

Ordinal Values

SELF DEFINED ENUMERATIONS

- In dealing with Self Defined Enumerations, it is sometimes more expedient to deal with the State rather than the State Name.

  It would be difficult to deal with these Digital, Flag and Switch points whose state names for PV may be different, eg. On/Off, Stop/Run, etc.

  It is easier to deal with the states ; State 0, State 1, or in CL vernacular ; **$ord0**, **$ord1**.

Example:

```
Package
     Param_list My_List
     Parameter PV: $2ords          -- up to max $5ords
     Parameter Flag_Pt: My_List Array(1..10)
     End My_List

Custom
     Parameter Flag_Pt: My_List Array(1..10)
End Custom

     Block ABC(Generic; at General)
%Relax Linker_SDE_Checks
     If Flag_Pt(1).PV = $ord1 and ............
     ..............
     End ABC
End Package
```

- Normally , when compiling references to self-defined enumerations, the State Names must match those found on the referenced point.

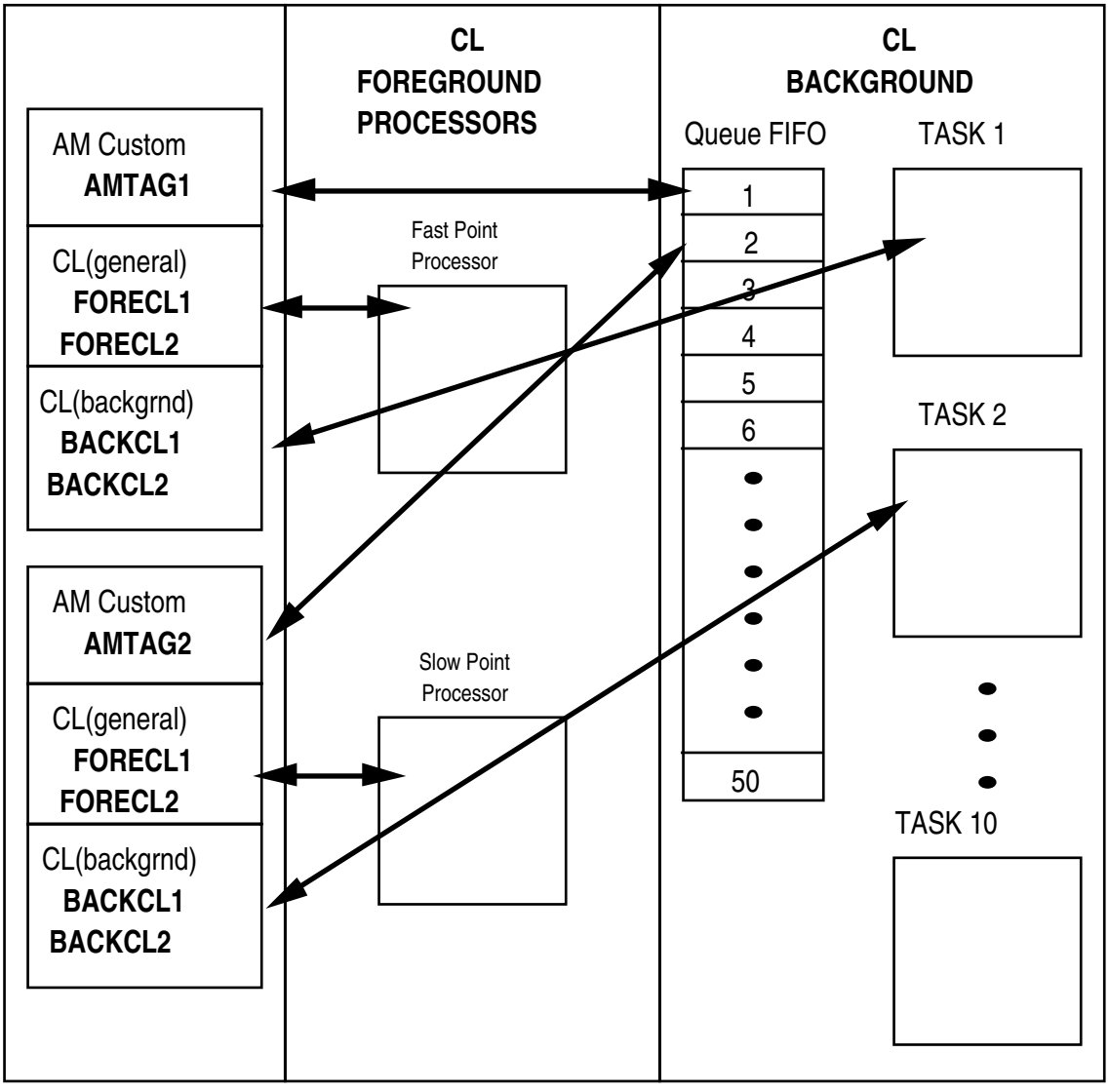
  The **%Relax Linker_SDE_Checks** directive relaxes this requirement and allows the CL code to operate directly on the State of the Parameter rather than the State Name.

  This option applies only to self-defined enumerations (e.g., Digital, Flag, or Switch Points) referenced indirectly (parameter is defined in a parameter list).

Background CL

SCHEDULING/OPERATION

- CL programs inserted at the **Backgrnd** insertion point of a Regulatory, Custom, or Switch point execute only during AM "Freetime".



Background CL

SCHEDULING/OPERATION

- Number of CL Background tasks ranges from 0 to 10. This may be adjusted via the NCF.

- CL Background task size must be set to support the largest background program on the Application Module. CL Background task size is adjustable via the NCF. Default size is 15000 words, with a range of 700 to 32000 words.

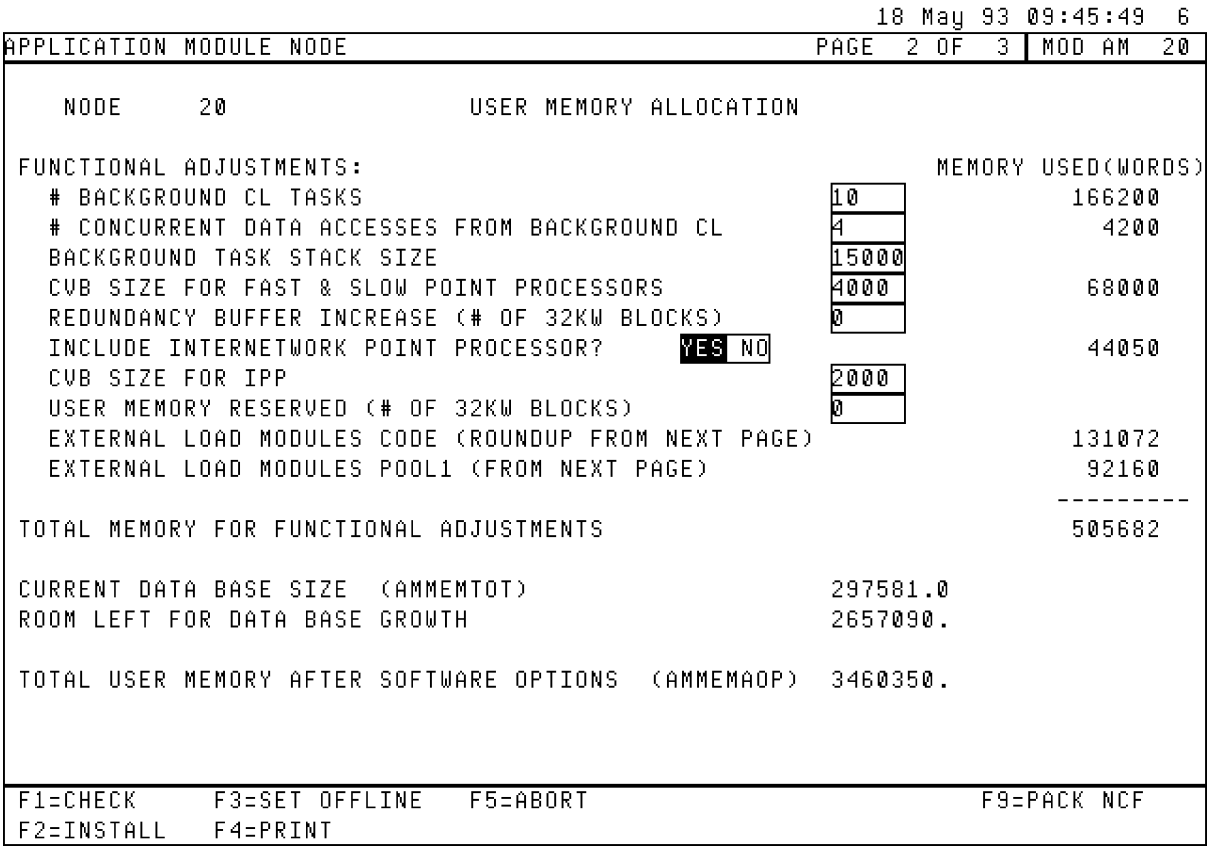
- Background programs are executed in 50 millisecond time slices per task.

- CL Background programs do not pre-fetch/post-store access data as foreground programs do. Fetch and stores are executed as they occur.

  Up to four concurrent data accesses may be executed at a time by a Background program. This is very inefficient. A more efficient method to access data from background is to fetch data in foreground and store in CDS parameters for access by background CL.

Background CL

SCHEDULING/OPERATION



Background CL

PRIORITIES

- Background CL programs may run at any of three priority levels: Low, Medium, or High. High priority is the default when a background CL is initiated.

- The priority may be changed via the BKG_Change_Priority subroutine. This subroutine may only be called by a background CL Program and may only change the priority of the calling program.

Example:

```
Block ABC (Generic; at Backgrnd)
Local prior: $Bkgprty

Call Bkg_Change_Priority (Low)
----------
----------
Set prior = High
Call Bkg_Change_Priority (prior)
End ABC
```

- A background CL in High priority will run to completion without interruption from Medium or Low priority programs. The same holds true for Medium priority programs over Low priority programs.

- Regardless of priority, background CL programs are always superseded by foreground CL Programs or functions, e.g.,. Checkpointing, Point Building, CL linking, etc.

Background CL

DELAYS

- Background CL programs may be delayed up to a maximum of one hour. This is accomplished with the BKG_Delay subroutine.

Example:

```
Block ABC(Generic;at Backgrnd)
     Local tconst : Time
     Set tconst = 1 Mins 5 Secs
     Call Bkg_Delay (2 Mins)
     Call Bkg_Delay (20 Secs)
     Call Bkg_Delay (tconst)
End ABC
```

- The BKG_Delay subroutine may only be called from a Background CL Program, and may only delay the calling program.

Process Point Special (PPS)

USAGE

- CL Programs execute at the selected scheduled time (Period) of the Bound Data Point. The selected execution time Period may be from 1,2,5,10,.....Seconds to once every 24 Hours.

- An alternate schedule selection is **No Period.** That is, the Bound Data Point has no fixed schedule and relies on an external source to trigger execution. This is called a Process Point Special (**PPS**).

- Three sources of triggering a PPS are:

  1. Another AM point with a CL program executing a PPS on the Non-Scheduled point.

Example:

```
Block ABC(Point A100; at xxxxxx)
     External B100            -- non-Scheduled AM Point
     Set B100.PPS = on
End ABC
```

  2. An operator initiated PPS via the PROCESS target on the Detail display or Graphics display.

  3. Event Initiated Processing triggered from a Hiway point or a UCN point. This is accomplished by entering the name of the Non-Scheduled AM point as the value for the EIPPCODE parameter of the Hiway or UCN point.