Appendix

Access to TDC Collector Data

This section describes how to access TDC (TPS) collector data. Collector data is provided by the LCN data server in addition to the point and DispDB data described in The Display Database (DispDB) . Collector data includes alarm and history information. See the "TDC 3000 (TPS 3000) Picture Editor Reference Manual" for more information on collectors.

Collector

A collector is a Honeywell-defined function that accesses data other than process and display database data. A collector can be used in expressions. Collectors cannot be called from actors (a.k.a. action procedures). Collectors can use the data scanning behavior of the data access subsystem to periodically fetch data.

Collector function provides access to collector data

Collector data is accessible from GUS displays by using the collector function It accepts one argument that specifies the desired collector data, and returns the value of the specified collector. The input argument must be a string literal containing a collector and its argument list as specified in Appendix H of the TDC 3000 (TPS 3000) Picture Editor manual.
Notes:

1. This string literal cannot contain any spaces, otherwise a validation error will occur.
2. Use the double quote (") character as an escape character when a double quote character is required within the string literal. For example

   ```
   collector(""$UNITSTS(""01"")")
   ```

   NOTICE: This is equivalent to the TDC syntax: $UNITSTS ("01")
3. You cannot parameterize the argument of a collector. For example, because the type of the return value depends on the specified collector, you cannot script the following:

   ```
   collector("ACKSTAT(dispdb.params.pointr)")
   ```

   The following example shows the proper use of the ACKSTAT collector,

   ```
   ackvalue = collector("ACKSTAT(A100)")
   ```

4. The TDC references within the collector string literal must not contain GUS prefixes DispDB or lcn:

   ```
   CORRECT:  collector("$PNTSTS(A100)")
   WRONG:    collector("$PNTSTS(lcn.A100)")
   CORRECT:  collector("$ADDBCNT(STRING01,ENM01)")
   WRONG:    collector("$ADDBCNT(DispDB.STRING01,DispDB.ENM01)")
   CORRECT:  collector("HOUR_V(A100.PV,3,2,1)")
   WRONG:    collector("HOUR_V(lcn.A100.PV,3,2,1)")
   ```

5. Use quotes (with each quote escaped with another quote) around all references to Primod Names, Unit Ids, and Annunciator Group Titles:

   ```
   collector("$PRIMCNT(""PRIMPNT"",$ALRMSTS:UNACKHI)")
   collector("$PRIMSTS(""PRIMPNT"")")
   collector("$UNITCNT(""01"",$ALRMSTS:ACKEM)")
   collector("$ANNSTS(""GRP001"")")
   ```

6. Do not put collectors in the OnDisplayStartup script. A runtime error will occur if a collector is used in an OnDisplayStartup script.

Appendix

CollectorHistory function initiates processing of history collectors

To initiate collection of history data, you must call the CollectHistory function from within a script. This is typically done in a script handling an operator event, after the operator has entered all the entities and parameters that the history collectors need. The description of the CollectHistory function has more details.

CollectHistory (statement)

Syntax

CollectHistory InitFlag[,ReferenceTime]

Description

This statement initiates the processing of history collectors.

Comments

The CollectHistory statement requires the following parameters:

Parameter Description
InitFlag A Boolean flag that determines whether or not old history values are discarded before collecting new values. If this parameter is TRUE, the value of all history collectors will be invalid until the new collection is complete. Furthermore, a data change event occurs for all history collectors referenced in OnDataChange subroutines indicating that the history collector data is not valid. If this parameter Is FALSE, the value of all history collectors is not changed, and no data change event occurs until valid, new values are collected.
ReferenceTime An optional second parameter that specifies a reference Date/Time. This type of parameter is Date. If this parameter is not present, the current time and date will be used. ReferenceTime establishes a date/time for the history collector. Arguments to the collector can offset the collection period from the reference time.

The CollectHistory statement is typically done in a script handling an operator event, after the operator has entered all the entities and parameters needed by the history collectors.
The following examples show several uses of the CollectHistory statement:

```
rem Use Get Var to establish a reference in a dispdb VAR item
CollectHistory TRUE
CollectHistory TRUE, DispDB.DATIME1
```

Last Page

Appendix