CyFlex® Calibration Tables and Utilities

Version 6

February 15, 2024

Developed by Transportation Laboratories

Version History

Version	Date	Revision Description
1	1/19/2016	Initial publication
2	8/23/2018	Format with SGS brand
3	4/3/2020	Retrofit to new template
4	12/3/2021	Revised *Section 8 Table Utilities* on page 13 to remove inline usage content and add hypertext linked cross-references to cyflex.com usage help.
5	6/6/2022	Removed Appendix H. table_report Utility Usage Details. Updated all hypertext linked cross-references to cyflex.com usage help descriptions
6	2/15/2024	Rebrand to TRP Laboratories

Document Conventions

This document uses the following typographic and syntax conventions.

- Commands, command options, file names or any user-entered input appear in Courier type. Variables appear in Courier italic type.

  Example: Select the `cmdapp-`*`relVersion-buildVersion`*`.zip` file….

- User interface elements, such as field names, button names, menus, menu commands, and items in clickable dropdown lists, appear in Arial bold type.

  Example: **Type**: Click **Select Type** to display drop-down menu options.

- Cross-references are designated in Arial italics.

  Example: Refer to *Figure 1*…

- Click intra-document cross-references and page references to display the stated destination.

  Example: Refer to *Section 1 Overview* on page 1.

  The clickable cross-references in the preceding example are *1*, *Overview*, and on page 1.

CyFlex Documentation

CyFlex documentation is available at https://cyflex.com/. View **Help & Docs** topics or use the **Search** facility to find topics of interest.

Table of Contents

1 Overview

1.1 Document Purpose

The purpose of this document is to provide the CyFlex user with an understanding of the basic mechanism which is used for conversion of raw measurement information to engineering values and for defining a two-dimensional (2D) or three-dimensional (3D) relationship between variables that can be used for dynamic interpolation. Calibration tables provide a mechanism to create mathematical representations of the relationship between input and output variables. The tables provide flexible, user-definable relationships that can be easily modified “on-the-fly” without shutting the system down and restarting it. CyFlex supports 2-dimensional (one input, one output) and 3-dimensional two inputs, one output) tables.

1.2 Calibration Table Files

Calibration tables are initially created as disk files and are stored in that form and backed up or archived. The 2D table files must all be stored in the `/cell/tables/` directory and must use the `.tbl` filename extender. The 3D tables are also stored in `/cell/tables/` but have a complicated naming convention. For information about how to access local tables, refer to *Section 7 Using Tables in Computed Expressions* on page 11.

1.3 Loading Tables into Computer Memory

To be used by CyFlex applications, 2D calibration tables must reside in the computer’s shared memory and can then be accessed by any application. 3D tables are loaded into the private memory space of the application which uses the table. All of the 2D tables are loaded into shared memory by a ‘translator’ application called `bld_tables`. This application is normally executed in the `go.scp` system startup script but can run manually from the command line anytime a new table is created or when a table is modified. It is launched from the `tcal_util` calibration utility when a new calibration is to be activated.

2 Table Types

CyFlex supports the following mathematical representations in 2D calibration tables.

2.1 Interpolation

The `INTERPOLATION` table type defines the x-y relationship (input-output) with a series of straight-line segments. The y-value is calculated from the $y=mx+b$ straight-line equation that represents the relationship between the 2 end points that bound the x input value. For example, if a table consists of 3 sets of x-y coordinates representing 2 linear segments. The output value is determined by searching the table to find which segment covers the range of the x input value. The output value is computed from the $y=mx+b$ equation for that segment. Should the input value fall outside the calibration range, the output is computed from the equation of the upper of lower segment, i.e., by extrapolation.

Up to 64 x-y coordinates may be used to represent an `INTERPOLATION` table. Below is an example of the last section of a table file for this type, where there were 3 calibration points that will be represented by 2 linear segments.

Note:

The `INTERPOLATION` type is not a best line fit of the calibration points. Use the POLY to represent a best line fit of more than 2 calibration points

```
#type_code
INTERPOLATION

#number of points
3

#          mv          IN_HG
       0.02189        0.00000
       8.22014      100.00000
      16.28055      200.00000
```

Refer to *Appendix A. Example INTERPOLATION Table Type - @AUG07 Format* on page 17 for an example `INTERPOLATION` table file.

2.2 POLYNOMIAL and POLYNOMIAL_RANGE

The polynomial types use a polynomial equation to represent the relationship. Presumably, the equation was derived from a best-fit regression to a calibration data set, but instead of using the actual x-y points, we are just using a polynomial expression.

$$y = C0 + (C1 * x) + (C2 * x**2) + (C3 * x**3) + (C4 * x**4) + \text{...etc.}$$

There may be up to 8 coefficients to represent the polynomial equation.

Below is an example of the last section of a `POLYNOMIAL` type table file. In this case, the coefficients represent a straight line. The coefficients are listed in the order, `C0`, `C1`, …`CN`.

```
#type_code
POLYNOMIAL

#number of points
2

#millivolts      eng_units
-6.2879E+00
 2.8300E-01
```

The `POLYNOMIAL_RANGE` type is a special form where there is a set of polynomial coefficients in the table for up to four different “ranges”. A `range_index` value is used to select which set is active. To set the index, refer to *Section 8.11 chg_pr_index* on page 15

```
#type_code
POLYNOMIAL_RANGE

# defaults:
#   bias - 0
#        BIAS=
#   gain - 1.0
#        GAIN=
#   virtual settings - 0
#        VZ_X=    VZ_Y=    VS_X=    VS_Y=    CALZ_X=     CALS_X=
#   range_min - 0
#        MIN=
#   range_max - 1.0e6
#        MAX=
#   polynomial coefficients - 0
#        C0=   C1=   C2=   C3=   C4=   C5=   C6=   C7=
#
# NOTE: The RANGE= specification must be the first entry on a line

RANGE=0, C0=15.9834, C1=8.71945, C2=-4.71032e-03, \
         C3=2.22203e-06, C4=-4.09139e-10, C5=3.e-14  MIN=0 MAX=40281
RANGE=1, C0=.123, C1=215.21, C2=12345, C3=.000004, MAX=5000
RANGE=2, C0=.123, C1=430.21, C2=12345, C3=.000004, MAX=10000
```

2.3 Scaled

The SCALED table type is a special polynomial calibration type where only the C1 term is used. C0 and all other coefficients are zero. The equation reduces to:

$$y = C1 * x$$

Stated differently, the output is simply a fixed multiple of the input. This is used to handle accumulating counters where a “change” of input represents a “change” of output or for frequency measurement where a number of pulses sampled over a fixed period of time can be converted to an RPM value. Typically, something such as a flow meter might produce a certain number of pulses per unit of volume. This “calibration” (gallons per pulse) can be used to accumulate a total flow (gallons) over a long period of time. The same calibration coefficient when sampled at a precise rate (gallons per pulse per second) can be used as a non-accumulating rate measurement (gallons per second).

Below is an example of the last section of a table file for this type, where the conversion factor is representing 0.1 engineering units for every increment of 1.0 in raw value.

```
#type_code
SCALED

#conversion factor
.1
```

2.4 Hyperbolic

The hyperbolic table is similar to the polynomial table except that the coefficients represent a different form of the equation.

$$Y = C0 + C1/x + C2 / x**2 + \ldots \text{ etc}$$

Below is an example of the last section of a table file for this type.

```
#type_code
HYPERBOLIC

#number of points
2
# the 2nd coefficient is computed as follows
# (counter frequency in hz) * 60[sec/min] * 16[none]
# / (number of teeth/revolution)

#coefficients for 60 tooth gear
0.000
16000000.0
```

2.5 3D Tables

There are two types of 3D tables:

1. The first are three-dimensional tables located in the shared memory calibration table area. This first type is called an `INTERP_TWO_D` table type.
2. The second type, `LOCAL_INTERP_TWO_D`, are "local" three-dimensional tables. The "local" three-dimensional tables are so-called because the space for these tables is allocated locally in the using application. Local three-dimensional tables are permitted to be larger than shared memory calibration tables.

The `INTERP_TWO_D` shared memory calibration table type is required in order to use a three-dimensional table for analog input calibration purposes. For this table type the table name is placed in the `inpt_specs` file as would be the case for any other calibration table. The raw analog input value is treated as the x-axis variable for this table type. The y and z-axes variables are identified within the table file itself.

The following example is a specification from the `inpt_specs` file for a case in which a `NOx` sensor can provide a `NOx` reading but requires temperature compensation. In this case, the `NOx` sensor's output is the X variable, the temperature is the Y variable, and the output (ppm of `NOx`) is the Z variable. The table name is `nox.tbl`, but only the root name of the file is entered in the spec file as shown below.

```
#type chan interval f_const compen table  label  units format range_x
AI    0      FAS      .9       -   nox     nox   ppm      1       1
```

Three-dimensional table types are used when it is desirable to model a parameter as a function of two other variables. In this case the table is used in conjunction with a function call in a computed variable expression in a `gen_labels` file.

The following examples are of specifications from the `gen_labels` specification file for the short (from long calibration table area) and long (a locally allocated three-dimensional table) table types. In these examples the X variable is the variable whose label is identified as the first calling argument of the function. The identity of the Y and Z variables are located in the three-dimensional table file. The table file to use is specified by the second argument in the function call. For the `shrt_3d_comp()` function, call the table file to be used is `two_d_2.tbl`. For the `long_3d_comp()` function, call the table file to be used is `loc_two_d_1.tbl`.

```
#label          units       format    initial_value      interval
oil_press3D     psi          2        -                  1000
@shrt_3d_comp( RPM, 2[none] )

int_man_t3d     deg_f        2        -                  1000
@long_3d_comp( TORQUE, 1[none] )
```

Refer to *Appendix G. Example of a Local Three-Dimensional Table File* on page 27 for an example of a 3D file.

3 Table Storage and File Types

Calibration tables must be located in the `/cell/tables/` directory so that they may be automatically loaded into shared memory when a CyFlex system is started.

All files located in this directory which have the `.tbl` filename extender are considered to be calibration tables. Each of these files will be read by the `bld_tables` utility and, depending on the `table_type` loaded into a special shared memory area which is designated exclusively for this purpose. Files of 3D table type are not loaded into this shared memory area. Instead, they will be loaded into the local memory of the applications which use them.

The `/cell/tables/` directory will also contain files with different filename extenders. These other files contain historical records of previous calibrations and processes related to calibration.

Files with the `.save` extender contain a record of calibration operations from the past. They are referred to as “archival” files. These files grow each time a calibration operation is performed using `tcal_util` by appending information about the most recent operation to the end of the file. The archival files are used as input to the `table_report` utility which primarily reports on calibration drift from one calibration to the next.

4 Table Access Methods

Calibration tables may be located in a shared memory area or in the local memory of an application. Most tables are in shared memory and have been placed there by the `bld_tables` application which runs in the `go.scp` startup script ().

4.1 Shared Memory Tables

These tables are normally associated with a sensor channel defined in the `inpt_specs` file. The transfer layer application such as `ai_transfer` or `fici_xfer`, will create the association between the raw sampled values for that channel and the conversion to engineering units defined in the calibration table.

These same shared memory tables can be accessed by any computed expression that is created in a `gen_labels` file, a `gp_test` procedure, or anywhere else in the system that supports computed expressions. The methods for doing this is are user functions:

- The function for 2D tables is called `@cal_table(value, table base name)`.
- For 3D tables, the function is, `@shrt_3d_comp( value, index)`.

Example:

```
@cal_table( raw_value_label, 'cmp_in_p' )
```

where `raw_value_label` is the label of a real variable that contains a value with the same units as the raw units in the file, `/cell/tables/cmp_in_p.tbl`. Refer to *Appendix A. Example INTERPOLATION Table Type - @AUG07 Format* on page 17 for an example of an interpolation table type.

4.2 Local Memory Tables

Tables which are stored in the local memory of an application are referred to as “local” tables and have a particular naming convention. Local tables are used for very large 3D tables. Three-dimensional (3D) calibration tables are tables in which an input-output relationship is established for the case in which there are two input variables and a single output variable.

Three dimensional tables are useful in the following applications:

- Analog input channel calibration.
- Parameter modeling for performance monitoring and safeties.
- Parameter modeling for establishing target test conditions.

Refer to *Section 7.2 Using 3D Tables in Shared Memory* on page 11 for information about how to access local tables.

5 Table Versions

Table “versions” refers to the format of calibration table files. CyFlex supports backward compatibility of the formats, so that tables generated many years ago in early versions of ASSET or CyFlex can still be read and understood. The very first line of a table file will contain a keyword with an @ symbol prefix, such as those below to identify the format.

It is possible that a very early (pre-1994) calibration table might be found without an @-prefixed keyword. If the file contains the basic information shown below, it can probably be handled properly.

The basic format found in every version includes the following information:

- User comment
- Last calibration date
- Expected calibration interval
- Calibration method – usually the QIE number of the calibrator
- Technician name
- A description of the sensor
- Engineering units
- Raw units
- Calibration Table Type; refer to *Section 2 Table Types* on page 2.
- Calibration data – format varies by table type

5.1 @941105 and @010301

These numbered formats refer to calibration files which contain the basic information listed above. The actual number after the @ symbol is not significant.

Refer to the following for examples:

- *Appendix B. Example POLYNOMIAL Table Type* on page 18
- *Appendix F. Example POLYNOMIAL_RANGE Table Type with @010301 Format* on page 24

5.2 POLY2 and @MAY04

The `@POLY2` and `@MAY04` formats add the following fields to the basic format:

- Sensor model
- Sensor serial number
- Sensor range units
- Sensor range lower limit / Sensor range upper limit

5.3 @AUG07

The `@AUG07` formats include an addition to the basic fields that includes that following fields as well as the fields that were added to `@MAY04`:

```
MUA Instance name
```

Refer to Appendix A. Example INTERPOLATION Table Type - @AUG07 Format on page 17 for an example of this file format.

5.4 @JAN10

The `@JAN10` format included information about the tolerance settings that are related to the `tcal_util` calibration GUI. That extra information is not required and the format has been obsolete. However, it can still be successfully read in as basically the same as `@AUG07`.

5.5 @POLY3

The `@POLY3` tag is for `POLY_RANGE` (polynomial ranges) table types. This table includes the possibility of several polynomial expressions that usually represent different gains or “ranges” of emission analyzers. An index variable allows selection of which range is active. Refer to *Appendix C. Example POLYNOMIAL_RANGE Table Type with @POLY3 Format* on page 19 for an example of this type of file.

6 Calibration Process/Generating Table Files

Generate calibration tables with the `tcal_util` calibration GUI. Refer to *CyFlex Calibration Utility*.

7 Using Tables in Computed Expressions

Calibration tables are used in CyFlex to relate one value to another through a mapping process. These tables are typically used to map a raw input channel in the transfer layer to the corresponding analog input real variable value. There is also the capability to use calibration tables as part of computed variable expressions. In this case the relationship in the table may be used to map any real variable value to another real variable.

7.1 Using 2D Tables in Shared Memory

Use the function `@cal_table( label, table_name )` in a computed expression in any application that supports computed expressions. The `table_name` refers to a table in `/cell/tables/` that has a `*.tbl` filename extender. The `label` refers to a CyFlex REAL_VARIABLE that will be used as the x-value to compute the corresponding y-value. How this is performed depends on the table type. The result is the return value from the expression.

Example:

> `“@cal_table( Engine_speed, ‘ThrotUprBnd’ )”`
>
> where `Engine_speed` is the label of a REAL_VARIABLE and `ThrotUprBnd.tbl` is the table filename.

7.2 Using 3D Tables in Shared Memory

Use the function `@shrt_3d_comp( label, table_index_number )` in a computed expression in any application that supports computed expressions.

3D tables have a particular naming convention that must be used. The table name for the short 3D tables in shared memory begins with `two_d_`, followed by an integer (*N*) and has the `*.tbl` filename extender. Accessing the table in a computed expression is done by calling the `@shrt_3d_comp()` function, where the integer (*N*) used in the filename is the second argument to the function.

> `two_d_`*N*`.tbl`
>
> where *N* is a integer number such as `0`, `1`, 2, etc.

Example:

> `two_d_1.tbl`
> `“@shrt_3d_comp(  xvalue, 1[none] )”`
>
> where the `1` in the function call references the `two_d_1.tbl` file that resides in the `/cell/tables/` directory.

7.3 Using Large 3D Tables in Local Memory

Tables which are stored in the local memory of an application are referred to as “local” tables and have a particular naming convention. These are used for very large 3D tables. The table files must reside in the `/cell/tables/` directory.

Local tables with table number:

`loc_two_d_`*`N`*`.tbl`

where *N* is a integer number such as `0`, `1`, `2`, etc.

Example:

`loc_two_d_9.tbl`

Example (access):

`“@long_3d_comp(  value, 9[none] )”`

where the `9` in the function call references the `loc_two_d_9.tbl` file that resides in the `/cell/tables/` directory.

8 Table Utilities

8.1 bld_tables

Use the `bld_tables` utility to process calibration table files into shared memory tables that are then available to the transfer layer applications such as `ai_transfer` and `fici_transfer`. The shared memory tables are also accessible to computed expressions which contain the user function `@cal_tables()`. An additional function performed by `bld_tables` is to set a refresh time stamp parameter which will cause applications that are using "local" tables to update the tables in their local memory by re-reading the local table files. This is equivalent to using the `update_loc` application, which is described in a section below.

The `bld_tables` application can be used to process individual tables or to scan the `/cell/tables/` directory and process all files with the `*.tbl` file extender. Individual tables are handled by listing them on the command line.

Examples:

```
bld_tables air_mtr0_p cmp_in_p
```

Process both the `/cell/tables/air_mtr0_p.tbl` file and `/cell/tables/cmp_in_p.tbl`

```
bld_tables
```

Process all table files found in the `/cell/tables/` directory

Refer to cyflex.com usage help for `bld_tables` for supplemental information.

8.2 table_report

The `table_report` application produces a summary of the currently active calibration in comparison to the previous calibration. A deviation greater than the 1% at any calibration point is flagged. The 1% deviation is the default tolerance and can be changed from the command line.

The report is based on files found in the `/cell/tables/ directory`. The current calibration is read from the `*.tbl` file and the previous data is derived from the `*.save` (archive) file. Refer to cyflex.com usage help for `table_report` for more information.

8.3 cal_rpts

The `cal_rpts` utility is a script that is intended for use by Operations personnel who perform periodic calibrations of sensors on a test system. After doing a sequence of calibrations, this script will run the `table_report` application twice, with and without the `+exceptions` options.

The output for each execution of `table_report` is stored in the `/cell/tables/reports/` directory.

Each file has a `time_stamp` included in the filename so that subsequent runs do not over-write the files. Thus, record of all the executions of `cal_rpts` will remain on the disk. Each report is automatically emailed to the `q5` email distribution list established in the `/etc/postfix/virtual` file.

A report named `cal_times` is also generated, showing the sequence of calibration operations that was performed during the calibration session.

Refer to cyflex.com usage help for `cal_rpts` for supplemental information.

8.4 table_find

Use this utility is identify calibration tables that are currently in use and those which are in the `/cell/tables/` directory but not in use. It also has a `+r` command line option which will delete the unused `*.tbl` table files and associated `*.save` archive files. Refer to cyflex.com usage help for `table_find` for supplemental information.

8.5 active_tables

Use the `active_tables` utility to produce a report listing all active tables. Active tables are those which are currently specified in `inpt_specs`. The report includes information found in the tables such as the sensor serial number, sensor range, last calibration date and, optionally the technician name. There is also a `–typeE` option to exclude from the report all thermocouple channels, since they generally share a common calibration table. Using the `–typeE` option will exclude all tables with the names `typeE.tbl`, `GantypeE.tbl`, and `opto_deg_f.tbl`.

Refer to cyflex.com usage help for `active_tables` for supplemental information.

8.6 set_table_units

Use the `set_table_units` utility to modify the engineering units field in a calibration table file. This does not change the sensor units or sensor range units, nor does it change any of the calibration values. This application will be able to read any of the old table formats, but when the modified file is written back out, it will be in the newest format (which is currently `@AUG07`). Consequently, it can be used as an update mechanism. However, the values newer fields such as `MUA_instance_name` are not updated.

Refer to cyflex.com usage help for `set_table_units` for supplemental information.

8.7 new_tables

Use the `new_tables` utility to identify calibration table files that are newer than a specified age (in days). The file timestamp determines the file age. This also has the option of sending the report to an email address specified on the command line. Thus, a process could be set up to run by the `cron` utility to notify an individual if any calibration files have been changed within a specified period.

Refer to cyflex.com usage help for `new_tables` for supplemental information.

8.8 conv_table_units

Use this utility to perform engineering units conversion from the units previously used in the table to the new units specified on the command line. Note that the units used previously in the table must be in the same units dimension as the new units (pressure to pressure, etc.). Running this command modifies the table file but does not load it into `shared_memory`.

Refer to cyflex.com usage help for `conv_table_units`.

Note:

To make the table active run `bld_tables`:

```
bld_tables int_mnf_p
```

8.9 cal_x_units

Use the `cal_x_units` utility to identify the units of the x-variable (raw units) in the calibration table file. This allows the `@cal_table()` function to properly perform the conversion. Any table this is used in an expression with `@cal_table()` must have the raw units in the table.

Refer to cyflex.com usage help for `cal_x_units`.

8.10 update_loc

Use this command to cause the tables that are referenced in computed expressions by `@long_3d_comp()` and `@short_3d_comp()` functions to be updated in the applications which are using those functions. Typically, these functions are specified in a `gen_labels` file and are being computed by the `compvar` application. This command has no options.

8.11 chg_pr_index

Use the `chg_pr_index` utility to change the index of a `POLY_RANGE` calibration table.

Refer to cyflex.com usage help for `chg_pr_index`.

Appendices

Refer to the following appendices for reference examples.

Appendix A. Example INTERPOLATION Table Type - @AUG07 Format

```
#revision code
@AUG07

# user comment
none

#   cal last adjusted on: Mon Mar 23 14:41:06 2015
#next calibration due on: 09/21/2015

#last_cal(time_t)        cal_interval
 1427136066               26[week]

#Working Standard
0712005625

#Technician
Aaron Martin/ Dave Ammerman

#Description
none

#calibration units
#eng_units          raw_units
  IN_HG                mv

# calibration limits
#  min_cal_range        max_cal_range      tolerance   tolerance_units
    0.00[IN_HG]      200.00[IN_HG]        1.00       %

#MUA Instance Name
none

#Sensor Model
Viatran   118

#Sensor Serial Number
287643

#Sensor Range Units
PSI_G

#Sensor range limits
#  lower_limit         upper_limit
    0.00[PSI_G]       100.00[PSI_G]

#type_code
INTERPOLATION

#number of points
3
#          mv        IN_HG
      0.02189      0.00000
      8.22014     100.00000
     16.28055     200.00000
```

Appendix B. Example POLYNOMIAL Table Type

```
#revision date
@941105

# user comment - 8/5/99 Copied from TC114 - S. Packer
Calibration coefficients for NOX analyzer in cart 51

#     last calibrated on: 9/12/1998
#next calibration due on: 9/12/1999

#last_cal(time_t)        cal_interval
 857742116               52[week]

#Calibration Method
none

#Technician
A B HAMMOND

#Serial Number
NOX analyzer concentration in PPM

#calibration_units
  PPM

#  min_range        max_range       range_index
    0.00[PPM]        2453.0[PPM]          0

#type_code
POLYNOMIAL

#number of points
2

#millivolts      eng_units
-6.2879E+00
 2.8300E-01
```

Appendix C. Example POLYNOMIAL_RANGE Table Type with @POLY3 Format

```
#########################################################################
# This is the new format for polynomial tables which support multiple
# instrument ranges.
#
# The "table_type" is POLYNOMIAL_RANGE
#
# The revision code must be "@POLY3"
#
# The table name is not contained within the file, but is
# derived from the filename.
#
# The information about a particular range must all be entered on one
# logical line, but the line continuation character '\' can be used.  The
# keywords used to specify the data are listed below.  There are default
# values for each keyword except "RANGE=" and the "RANGE=" keyword
# must be the first entry on each line.  There must not be a space
# between the '=' character and the value following it.
#  example:    RANGE=0     not    RANGE= 0
#
# Although the CALZ_X and CALS_X keywords are optional, they are required
# for virtual zero and virtual span operations.  They represent the
# actual millivolt values at hardware zero and span and at which the
# bias is 0.0 and the gain is 1.0.
#
#  NOTE: the range index will be selected through the virtual_zero process
#
#              keyword list
#
#  RANGE=
#     A number between 0-7 which specifies the instrument range index.
#        example:   RANGE=0
#
#     For a single range instrument, range 0 should always be used.
#
#  C0=, C1=, C2=, C3=, C4=, C5=, C6=, C7=
#     The polynomial coefficients for a particular range which will be
#     use to convert millivolts to engineering units. Default values
#     are 0.0
#
#  BIAS=
#     A bias value that will be computed as a result of a virtual
#     zeroing operation.  The default value is 0.0, but after a zeroing
#     operation, this value may be written back into the calibration file.
#
#  GAIN=
#     A multiplier value that will be computed as a result of a virtual
#     zeroing/spanning operation.  The default values is 1.0, but after
#     a zero/span operation, this value may be written back into the
#     calibration file.
#
#  MIN=
#     The minimum value at which the calibration is valid.  The default
```

```
#     values is 0.0
#
#  MAX=
#     The maximum value at which the calibration is valid.  The default
#     value is 1,000,000.
#
#  VZ_X=
#     The millivolt value at virtual zero.  The default value is 0.0
#
#  VZ_Y=
#     The engineering units at virtual zero.  The default value is 0.0
#
#  VS_X=
#     The millivolt value at virtual span.  The default value is 0.0
#
#  VS_Y=
#     The engineering units at virtual span.  The default value is 0.0
#
#  CALZ_X=
#     The millivolt value at initial calibration zero.  (hardware zero)
#     The default value is 0.0
#
#  CALS_X=
#     The millivolt value at initial calibration span.  (hardware span)
#     The default value is 0.0

###############################################################################
#revision code
@POLY3

# user comment
no comment

#     last calibrated on: Sat Oct 12 11:55:15 2000
#next calibration due on: Sat Apr 12, 2001

#last_cal(time_t)        cal_interval
 982258578               26[week]

#Calibration Method
QIE 0711100066

#Technician
DEW/LINE

#Serial Number
CAI 0000000001


#calibration units
#eng_units          raw_units
  psi                 mv

# calibration limits
#  min_cal_range        max_cal_range      tolerance   tolerance_units
    0.00[psi]        0.00[psi]        0.00
```

```
#MUA Instance Name
none

#Sensor Model
none

#Sensor Serial Number
none

#Sensor Range Units
psi

#Sensor range limits
#  lower_limit         upper_limit
    0.00[psi]         0.00[psi]


#type_code
POLYNOMIAL_RANGE


# defaults:
#   bias - 0
#        BIAS=
#   gain - 1.0
#        GAIN=
#   virtual settings - 0
#        VZ_X=    VZ_Y=    VS_X=    VS_Y=    CALZ_X=     CALS_X=
#   range_min - 0
#        MIN=
#   range_max - 1.0e6
#        MAX=
#   polynomial coefficients - 0
#        C0=   C1=   C2=   C3=   C4=   C5=   C6=   C7=
#
# NOTE: The RANGE= specification must be the first entry on a line

RANGE=0, C0=15.9834, C1=8.71945, C2=-4.71032e-03, \
         C3=2.22203e-06, C4=-4.09139e-10, C5=3.e-14  MIN=0 MAX=40281
RANGE=1, C0=.123, C1=215.21, C2=12345, C3=.000004, MAX=5000
RANGE=2, C0=.123, C1=430.21, C2=12345, C3=.000004, MAX=10000
```

Appendix D. Example SCALED Table Type

```
#revision code
@941105

# user comment
none

#     last calibrated on: Wed Mar 19 10:21:07 2003
#next calibration due on: Wed Sep 17, 2003

#last_cal(time_t)        cal_interval
 1048087267               26[week]

#Calibration Method
none

#Technician
none

#Serial Number
Total fuel Calibration

#calibration_units
  gal

#  min_range        max_range
    0.00[gal]        0.00[gal]

#type_code
SCALED

#conversion factor
.1
```

Appendix E. Example HYPERBOLIC Table Type

```
#revision date
@941105

# user comment
this is a calibration of speed for a period measurement system

#     last calibrated on: Fri Dec  5 17:53:59 1997
#next calibration due on: Fri Jun 05, 1998

#last_cal(time_t)        cal_interval
 881362439               26[week]

#Calibration Method
computed for 1 Mhz reference frequency (x is counts per tooth)

#Technician
Logterman

#Serial Number
60 tooth dyno gear
#103 tooth flywheel gear
##isx113 tooth flywheel gear
#160 tooth flywheel gear
#calibration_units
  rpm

#  min_range        max_range       range_index
    0.00[rpm]      2500.00[rpm]          0

#type_code
HYPERBOLIC

#number of points
2

# the 2nd coefficient is computed as follows
# (counter frequency in hz) * 60[sec/min] * 16[none]
# / (number of teeth/revolution)

#coefficients
0.000

#################
#  for 60 tooth gear
16000000.0
#
#################

#################
#  for 113 tooth gear
#8495575.221
```

Appendix F. Example POLYNOMIAL_RANGE Table Type with @010301 Format

```
#########################################################################
# This is the new format for polynomial tables which support multiple
# instrument ranges.
#
# The "table_type" is POLYNOMIAL_RANGE
#
# The revision code must be "@010301"
#
# The table name is not contained within the file, but is
# derived from the filename.
#
# The information about a particular range must all be entered on one
# logical line, but the line continuation character '\' can be used.  The
# keywords used to specify the data are listed below.  There are default
# values for each keyword except "RANGE=" and the "RANGE=" keyword
# must be the first entry on each line.  There must not be a space
# between the '=' character and the value following it.
#  example:    RANGE=0     not    RANGE= 0
#
# Although the CALZ_X and CALS_X keywords are optional, they are required
# for virtual zero and virtual span operations.  They represent the
# actual millivolt values at hardware zero and span and at which the
# bias is 0.0 and the gain is 1.0.
#
#  NOTE: the range index will be selected through the virtual_zero process
#
#              keyword list
#
#  RANGE=
#     A number between 0-7 which specifies the instrument range index.
#        example:   RANGE=0
#
#     For a single range instrument, range 0 should always be used.
#
#  C0=, C1=, C2=, C3=, C4=, C5=, C6=, C7=
#     The polynomial coefficients for a particular range which will be
#     use to convert millivolts to engineering units. Default values
#     are 0.0
#
#  BIAS=
#     A bias value that will be computed as a result of a virtual
#     zeroing operation.  The default value is 0.0, but after a zeroing
#     operation, this value may be written back into the calibration file.
#
#  GAIN=
#     A multiplier value that will be computed as a result of a virtual
#     zeroing/spanning operation.  The default values is 1.0, but after
#     a zero/span operation, this value may be written back into the
#     calibration file.
#
#  MIN=
#     The minimum value at which the calibration is valid.  The default
```

```
#     values is 0.0
#
#  MAX=
#     The maximum value at which the calibration is valid.  The default
#     value is 1,000,000.
#
#  VZ_X=
#     The millivolt value at virtual zero.  The default value is 0.0
#
#  VZ_Y=
#     The engineering units at virtual zero.  The default value is 0.0
#
#  VS_X=
#     The millivolt value at virtual span.  The default value is 0.0
#
#  VS_Y=
#     The engineering units at virtual span.  The default value is 0.0
#
#  CALZ_X=
#     The millivolt value at initial calibration zero.  (hardware zero)
#     The default value is 0.0
#
#  CALS_X=
#     The millivolt value at initial calibration span.  (hardware span)
#     The default value is 0.0

################################################################################
#revision code
@010301

# user comment
no comment

#     last calibrated on: Sat Oct 12 11:55:15 2000
#next calibration due on: Sat Apr 12, 2001

#last_cal(time_t)        cal_interval
 982258578               26[week]

#Calibration Method
QIE 0711100066

#Technician
DEW/LINE

#Serial Number
CAI 0000000001

#calibration_units
  ppm

#type_code
POLYNOMIAL_RANGE


# defaults:
```

```
#   bias - 0
#        BIAS=
#   gain - 1.0
#        GAIN=
#   virtual settings - 0
#        VZ_X=    VZ_Y=    VS_X=    VS_Y=    CALZ_X=     CALS_X=
#   range_min - 0
#        MIN=
#   range_max - 1.0e6
#        MAX=
#   polynomial coefficients - 0
#        C0=   C1=   C2=   C3=   C4=   C5=   C6=   C7=
#
# NOTE: The RANGE= specification must be the first entry on a line

RANGE=0, C0=15.9834, C1=8.71945, C2=-4.71032e-03, \
         C3=2.22203e-06, C4=-4.09139e-10, C5=3.e-14  MIN=0 MAX=40281
RANGE=1, C0=.123, C1=215.21, C2=12345, C3=.000004, MAX=5000
RANGE=2, C0=.123, C1=430.21, C2=12345, C3=.000004, MAX=10000
```

Appendix G. Example of a Local Three-Dimensional Table File

```
# Description/Serial Number
Intake Manifold Temp Mapping
# Table Type Table Name
5 loc_two_d_0
# x units y units z units
RPM LB_FT deg_F
# x label
RPM
# x coordinate values
800 900 1000 1100 1200 1300 1400 1500 1600 1700 1800 1900 2000 2100
# y/z coordinate pairs
# x values:
#800      900       1000     1100     1200     1300     1400     1500     1600     1700     1800     1900     2000     2100
# spd tmp| spd tmp| spd tmp| spd tmp| spd tmp| spd tmp| spd tmp| spd tmp| spd tmp| spd tmp| spd tmp| spd tmp| spd tmp| spd tmp
    0  77    0  77    0  77    0  77    0  77    0  77    0  77    0  77    0  77    0  77    0  77    0  77    0  77    0  77
  100  78  100  78  100  78  100  79  100  79  100  79  100  79  100  80  100  80  100  80  100  80  100  80  100  80  100  80
  200  79  200  80  200  80  200  81  200  81  200  81  200  82  200  82  200  84  200  83  200  83  200  82  200  82  200  82
  300  79  300  80  300  81  300  82  300  83  300  83  300  83  300  85  300  86  300  86  300  86  300  85  300  85  300  84
  400  81  400  82  400  83  400  84  400  85  400  85  400  86  400  87  400  89  400  89  400  89  400  88  400  87  400  86
  500  82  500  83  500  84  500  86  500  88  500  88  500  90  500  91  500  92  500  92  500  92  500  91  500  91  500  90
  600  83  600  84  600  85  600  87  600  88  600  89  600  90  600  92  600  94  600  95  600  94  600  94  600  93  600  92
  700  84  700  85  700  87  700  88  700  89  700  91  700  92  700  95  700  97  700  98  700  97  700  97  700  97  700  96
  800  85  800  86  800  88  800  90  800  91  800  93  800  94  800  97  800 100  800 101  800 101  800 100  800 100  800  99
  900  85  900  87  900  89  900  91  900  93  900  94  900  96  900  99  900 102  900 104  900 104  900 103  900 103  900 102
 1000  86 1000  88 1000  90 1000  93 1000  94 1000  97 1000  99 1000 103 1000 106 1000 107 1000 107 1000 108 1000 107 1000 106
 1100  86 1100  88 1100  91 1100  94 1100  96 1100  99 1100 101 1100 105 1100 110 1100 110 1100 110 1100 110 1100 110 1100 108
 1200  86 1200  89 1200  92 1200  95 1200  98 1200 101 1200 103 1200 107 1200 115 1200 112 1200 112 1200 112 1200 111 1200 111
 1300  87 1300  90 1300  93 1300  96 1300  99 1300 103 1300 105 1300 108 1300 115 1300 114 1300 115 1300 114 1300 112 1300 111
 1400  87 1400  90 1400  94 1400  98 1400 101 1400 104 1400 106 1400 110 1400 120 1400 120 1400 118 1400 114 1400 112 1400 111
 1500  87 1500  91 1500  95 1500  98 1500 102 1500 105 1500 108 1500 112 1500 120 1500 125 1500 118 1500 114 1500 112 1500 111
 1600  87 1600  92 1600  95 1600 100 1600 103 1600 107 1600 110 1600 114 1600 120 1600 125 1600 118 1600 114 1600 112 1600 111
 1700  88 1700  92 1700  96 1700 100 1700 106 1700 109 1700 111 1700 116 1700 120 1700 125 1700 118 1700 114 1700 112 1700 111
 1800  88 1800  93 1800  97 1800 101 1800 106 1800 110 1800 113 1800 116 1800 120 1800 125 1800 118 1800 114 1800 112 1800 111
```