

Control Function (CF) Package Users' Guide

The Control Function (CF) package allows the user to define functions of variables in the MELCOR database, and makes the values of these functions available to other packages in MELCOR. As a result, the user can utilize the time-dependent values of variables that will be calculated during execution of MELCOR. The CF package has a number of uses, some of which may not be immediately obvious. For example, the pressures in appropriate control volumes may be used to control the opening of a valve or the failure of a blow-out panel, the temperature in a volume may define the enthalpy associated with a mass source/sink, or the particle loading on a filter may modify the flow resistance in the corresponding flow path. Complicated control logic may be simulated, involving the values of a number of variables in the system. New variables, not ordinarily part of MELCOR output, may be defined for printed or plotted output. The occurrence of specified conditions may be tested and used to initiate generation of additional output (a single line message, a complete list edit, or an additional plot or restart dump) from MELCOR.

This document includes an introduction to control functions, the concepts involved, and how they may be defined and used. It contains a list of the available types of Control Functions, with the necessary descriptions of each, and describes the relevant MELGEN and MELCOR input. It also presents a number of detailed examples of control function input. This is intended both to aid the new user in getting started, and to illustrate a number of "tricks" which can simplify the task of writing control function definitions.

CF Package Users' Guide

Contents

1.	Introduction	5
1.1	Concept of Control Functions	5
1.2	Applications of Control Functions.....	6
1.3	Definition of Control Functions	7
2.	List of Control Functions	8
2.1	Real-Valued Control Functions	8
2.1.1	Elementary and FORTRAN Functions.....	9
2.1.2	Derivatives and Integrals	10
2.1.3	Proportional-Integral-Differential Control Function	11
2.1.4	Hysteresis Function	12
2.1.5	Trips.....	13
2.1.6	User-Defined Functions	15
2.2	Logical-Valued Control Functions	16
3.	User Input Requirements.....	17
3.1	MELGEN User Input	18
	CFn...n00 – Control Function Definition Record	19
	CFn...n01 – Initial Value of Control Function	19
	CFn...n02 – Upper and Lower Bounds.....	20
	CFn...n0k – Miscellaneous Numbers	20
	CFn...n05 – Logical Control Function Classification.....	21
	CFn...n06 – Logical Control Function Switching Message.....	21
	CFn...nkk – Control Function Arguments.....	21
3.2	MELCOR User Input	22
	CFn...n00 – Control Function Definition Record	22
	CFn...n01 – Initial Value of Control Function	22
	CFn...n02 – Upper and Lower Bounds.....	23
4.	Sensitivity Coefficients.....	23
5.	Plot Variables and Control Function Arguments.....	23
6.	Example Input.....	23
6.1	Constants.....	23
6.2	Pressure and Head Differences.....	24
6.3	Valve Control	25
6.4	Messages and Other Output.....	26
6.5	New Output Variables	26
6.6	Voting Logic	28
6.7	Homologous Pump Model.....	29

List of Figures

Figure 2.1 Hysteresis Function 12

1. Introduction

The following subsections give an introduction to control functions as implemented in MELCOR: what they are, how they are used, and how they are defined by user input. Control functions provide a powerful tool which is intended to alleviate the need to reprogram MELCOR for special cases, permitting the user to effectively modify coding through normal user input.

1.1 Concept of Control Functions

The Control Function (CF) package allows the user to define functions of variables in the MELCOR database, and makes the values of these functions available to the various physics packages. The values of the control functions are themselves variables in the database, each calculated from the start-of-step values of the variables which define its arguments, and are therefore also available for printing and/or plotting as desired. Both real-valued and logical-valued control functions are available.

Note that each control function has a unique value at any time, because all of its arguments (other variables in the MELCOR database) are explicit functions of time. It is therefore itself an implicit function of time. This is in contrast to tabular functions, which are functions in the pure mathematical sense of a rule for accepting arguments and returning function values. In order to obtain a value from the tabular function package, a physics package must specify the argument of the function. (This is the reason that tabular functions cannot be plotted directly, although a control function may be defined as a tabular function of time—or any other available argument—and plotted as desired.)

The Control Function package serves primarily as a utility. It is extremely powerful, but its application is limited to cases where the use has been foreseen and appropriate coding has been provided.

- (1) In principle, the Control Function package has access to the entire MELCOR database for use as arguments. This access requires that specific coding be included in the packages which calculate the various data. Therefore, only a limited number of variables in the database are actually available for use as control function arguments; these are listed in the Users' Guides for the various packages to which the data belong.
- (2) The executive or any physics package may refer to a control function (meaning the value of that control function) by number and use it in any way desired. Here again, the actual use of control functions is limited to cases where specific coding is included in the package that will use the data; this is indicated in the input descriptions in the various package Users' Guides. Whenever a control function is specified as part of MELGEN input, it is the user's responsibility to see that the

CF Package Users' Guide

function he defines is appropriate. The Control Function package cannot anticipate how a function value will be used, and can only check for consistency of its input. Most packages do only limited error checking—primarily for existence of the function—in MELGEN, and trap only the most obvious errors in function values in MELCOR.

1.2 Applications of Control Functions

A real-valued control function may be used to define an input quantity for a physics package, allowing the input to be a function of current conditions in the system being modeled. The possibilities include:

- (1) rate of mass or enthalpy addition, or temperature of a mass source or sink (negative source) in the CVH package;
- (2) value of an independent thermodynamic variable in a time-specified volume in the CVH package;
- (3) velocity in a “time-dependent” flow path in the FL package;
- (4) fraction of a flow path which is open (to represent valves, breaches, blow-out panels, etc.) in the FL package;
- (5) laminar friction coefficient in a flow path segment in the FL package (this is used to model the effects of filter loading);
- (6) spray flow rate in the SPR package;
- (7) inlet temperature for the dT/dz model in the COR package;
- (8) fission power in the COR package; and
- (9) partition of decay heat between metallic and oxidic phases in the CAV package.

Logical- or real-valued Control Functions may be used to initiate or control the operation of components or models as conditions change in a calculation (and in some cases provide the only way to do so). Examples include:

- (1) opening and closing of valves in the FL package;
- (2) operation of pumps in the FL package;
- (3) actuation of spray sources in the SPR package;
- (4) failure of the lower head in the COR package;

- (5) operation of igniters in the BUR package; and
- (6) operation of the PAR.

Logical-valued control functions may be used to initiate additional MELCOR output as an additional list, plot, restart dump, or a short message. This may allow close examination of "interesting" parts of a calculation or capture peak pressures or temperatures in the list or plot files. It may also permit restarts from potential branch points in an accident sequence, or record the time of occurrence of specified events. Control Functions may also be used simply to evaluate desired quantities that would not otherwise appear in the list or plot files. Note, however, that MELCOR 1.8.5 allows values of specified control function arguments to be added directly to the plot file, *without* defining a control function "EQUAL" to that argument. See the EXEC Users' Guide for details.

In all of these cases, complex functions may be built up as desired. For example, a relief valve may be controlled as a function of the pressure difference between two control volumes, including appropriate static head terms. The different setpoints in a multi-bank relief valve system may also be modeled, and a block or bypass valve included if desired.

1.3 Definition of Control Functions

A control function is referred to by a user-assigned number. It also has a mnemonic name for edit purposes. Further input defines what type of a function it is; for example, CF 102 may be called 'TEST 1' and be evaluated as a square root; CF 232 may be called 'REACTOR SCRAM' and be evaluated using one of several varieties of trips.

Each control function has one or more arguments, which may be real-valued or logical valued; the number and type of the arguments depend on the specific function involved. The specification of each real-valued argument includes a user-input multiplier and an optional additive constant to be applied to the element of the database to which it refers. That is, if "X" is a real-valued element of the database which is listed as an available control function argument, the actual argument used in evaluation of a control function has the form $A X + B$. A value must be specified for A; B has the default value of 0.0. For example, the control function argument CVH-P.102 refers to the pressure in control volume 102 (assuming that this volume exists); the actual argument to be used by a control function could be specified as $a_1 = 2.0 * CVH-P.102 - 1.7E6$.

A real-valued control function may be modified by an overall multiplier and additive constant after the basic function has been evaluated. Thus, using the basic SIN function, the final value of the control function could be specified as $5.3 * SIN(a_1) + 1.8$. Optional upper and/or lower bounds on the value may also be specified.

A logical-valued control function may be forced to "latch," so that its value cannot change more than once. It may also be converted to a "one-shot," so that it can be true no more

CF Package Users' Guide

than once. Further, it may be made to “alarm” when it changes state, producing a user-specified message in MELCOR output files.

Optional initial values may be specified for each control function. These values may be required to completely define the value of a function with “history” (dependence on past values) such as an integral or a hysteresis function. They may also be used for initialization purposes by other packages during execution of MELGEN. (Note that, in some cases, the definition might otherwise be circular. For example, the value of a control function defining a heat transfer coefficient might depend on a temperature that in turn depended on that control function.)

The definition of a control function may also include the input of one or more *miscellaneous numbers* such as the setpoint for a trip, or the user number of a tabular function. This will become clear when particular cases are examined. (When additional data—such as the values of arguments on the previous timestep—must be saved in the CF database, they are simply added to the list of miscellaneous numbers during input processing. Therefore, the number of miscellaneous numbers shown in edits may be greater than the number input.)

The value of a control function is itself available for use as a control function argument, so that complicated functions may be built if desired. The user should be aware that control functions are evaluated in numerical order. Thus, if one control function refers to another with a higher (user) number, the value used for the latter will always be out-of-date by one timestep.

2. List of Control Functions

The control functions currently available in MELCOR are listed and described below. In each case, the listing and description correspond to the basic function, and do not include the possible scaling or bounding of real-valued functions, or conversion of logical-valued functions to “latches” or “one-shots,” which was discussed in Section 1.3. The formal arguments referred to may include scaling from the actual elements of the MELCOR database.

2.1 Real-Valued Control Functions

In the following, it is to be understood that, in general, the value of the N-th control function, with n arguments, a_i , and m miscellaneous numbers, c_j , is

$$FUNCT_N = SC_N f(a_1, a_2, \dots, a_n; c_1, c_2, \dots, c_m) + AC_N$$

where f is the basic function described below, and SC_N and AC_N are input scaling and offset parameters for the control function. Further, the i -th argument has the general form,

$$a_i = SA_i X + AA_i$$

where X is a *control function argument* available in some package of MELCOR, and SA_i and AA_i are scaling and offset parameters input to the Control Function package.

2.1.1 Elementary and FORTRAN Functions

The first group of real-valued functions are elementary and FORTRAN functions. Their definitions should be clear from the following table, which contains a descriptive name, a short name used as the input type, and a mathematical or FORTRAN definition. In some cases, an additional input parameter, referred to as a "miscellaneous number," is required to complete the definition of the function.

In the list that follows, the order is not strictly alphabetical; similar functions have been grouped together. Except as noted (for L-A-IFTE), all arguments are real. Miscellaneous numbers may be real or integer, depending on their usage.

Function Name	Input Type	# arg	Mathematical or FORTRAN Definition	Miscellaneous Numbers
Equals	EQUALS	1	$f = a_1$	none
Absolute Value	ABS	1	$f = a_1 $	none
Add	ADD	≥ 2	$f = a_1 + \dots + a_n$	none
Positive Diff	DIM	2	$f = \text{DIM}(a_1, a_2)^{[a]}$	none
Multiply	MULTIPLY	≥ 2	$f = a_1 * \dots * a_n$	none
Divide	DIVIDE	$2^{[b]}$ 1	$f = a_2/a_1$ $f = 1.0/a_1$	none none
Power-Integer	POWER-I	1	$f = (a_1)^J$	1:J = integer exponent
Power-Real	POWER-R	1	$f = (a_1)^R$	1:R = real exponent
Power-Variable	POWER-V	2	$f = (a_1)^{a_2}$	none
Exponential	EXP	1	$f = \exp(a_1)$	none
Natural Log	LN	1	$f = \ln(a_1)$	none
Base 10 Log	LOG	1	$f = \log_{10}(a_1)$	none
Square Root	SQRT	1	$f = \sqrt{a_1}$	none
Cosine	COS	1	$f = \cos(a_1)$	none
Sine	SIN	1	$f = \sin(a_1)$	none
Tangent	TAN	1	$f = \tan(a_1)$	none

CF Package Users' Guide

Function Name	Input Type	# arg	Mathematical or FORTRAN Definition	Miscellaneous Numbers
Arccosine	ARCCOS	1	$f = \cos^{-1}(a_1)$	none
Arcsine	ARCSIN	1	$f = \sin^{-1}(a_1)$	none
Arctangent	ARCTAN	1	$f = \tan^{-1}(a_1)$	none
Hyperbolic Cos	COSH	1	$f = \cosh(a_1)$	none
Hyperbolic Sin	SINH	1	$f = \sinh(a_1)$	none
Hyperbolic Tan	TANH	1	$f = \tanh(a_1)$	none
Maximum	MAX	≥ 2	$f = \max(a_1, \dots, a_n)$	none
Minimum	MIN	≥ 2	$f = \min(a_1, \dots, a_n)$	none
Type one Sign	SIGN	2	$f = \text{SIGN}(a_2, a_1)^{[c]}$	none
Type two Sign	SIGNI	1	$f = \text{SIGN}(1.0, a_1)^{[c]}$	none
Unit Normalize	UNIT-NRM ^[d]	1	IF (a ₁ .EQ.0.0) THEN f = 0.0 ELSE f = SIGN(1.0, a ₁) ^[c] END IF	none
Tabular Func	TAB-FUN	1	$f = \text{TF}_N(a_1)^{[e]}$	1:N = Tab. Fcn number
If Then Else	L-A-IFTE	3 ^[f]	IF (a ₁) THEN f = a ₂ ELSE f = a ₃ END IF	None

[a] $\text{DIM}(x,y) \equiv \max(x-y, 0.0)$ is a FORTRAN intrinsic function.

[b] DIVIDE may be invoked with one argument or with two.

[c] $\text{SIGN}(x,y) \begin{cases} |x|, & y \geq 0 \\ -|x|, & y < 0 \end{cases}$ is a FORTRAN intrinsic function.

[d] UNIT-NRM differs from SIGNI only in the value for $a_1 = 0$.

[e] TF_N is the MELCOR Tabular Function with user number N.

[f] The first argument of L-A-IFTE is logical, the second and third are real.

2.1.2 Derivatives and Integrals

Two finite-difference approximations to derivatives and one finite-difference approximation to an integral are available as control functions. Each requires two arguments and no miscellaneous numbers, with the first argument being the dependent variable and the

second the independent variable. Note that the independent variable is general, and need *not* be time.

These functions employ values from the MELCOR database at more than one time level. The old values needed are preserved by the Control Function package as part of its own database. In what follows, the time level is indicated by a superscript cycle number n , $n-1$, or $n-2$. The Control Function package is called at the end of a MELCOR advancement, to calculate values for use on the next step. Therefore, “ n ” refers to the end of the just-completed advancement, “ $n-1$ ” to the start of that advancement, and “ $n-2$ ” to the start of the immediately preceding advancement. Note that because of the dependence on past history, the definition of appropriate initial values may be important in initializing these functions. The three functions are:

Derivative-Centered Difference DER-C

This function evaluates a derivative using information at three time levels in the form

$$f^n = \frac{1}{2} \left(\frac{a_1^n - a_1^{n-1}}{a_2^n - a_2^{n-1}} + \frac{a_1^{n-1} - a_1^{n-2}}{a_2^{n-1} - a_2^{n-2}} \right) \quad (2.1)$$

Derivative-Forward Difference DER-F

This function evaluates a derivative using information at only two time levels in the form

$$f^n = \left(\frac{a_1^n - a_1^{n-1}}{a_2^n - a_2^{n-1}} \right) \quad (2.2)$$

Integral INTEG

This function evaluates an integral using information at two levels. It does not use the additive constant (AC) in converting f to FUNCT; if a value for AC is input, it must be 0.0. The integral is calculated as

$$f^n = f^{n-1} + \frac{1}{2} (a_1^n + a_1^{n-1}) (a_2^n - a_2^{n-1}) \quad (2.3)$$

2.1.3 Proportional-Integral-Differential Control Function

This control function has the short input name PID, and requires one argument. It is used in control systems, and is defined by the equation

$$f^n = R_1 a_1^n + R_2 \int_{t^n} a_1(t) dt + R_3 \left. \frac{da_1}{dt} \right|_{t^n} \quad (2.4)$$

where R_1 , R_2 , and R_3 are input as miscellaneous REAL numbers. The PID function is evaluated as being equivalent to the composite function

$$\begin{aligned} \text{PID}(a_1) = & R_1 \text{EQUALS}(a_1) + R_2 \text{INTEG}(a_1, \text{TIME}) \\ & + R_3 \text{DER-F}(a_1, \text{TIME}) \end{aligned} \quad (2.5)$$

using equations equivalent to Equations (2.2) and (2.3).

2.1.4 Hysteresis Function

This control function has the short input name HYST, and requires one argument. It is used to model the type of hysteresis behavior exhibited by components such as relief valves which are opened as the controlling variable (e.g., differential pressure) is increasing and closed as it is decreasing, with a dead band between. A typical hysteresis function is shown in Figure 2.1.

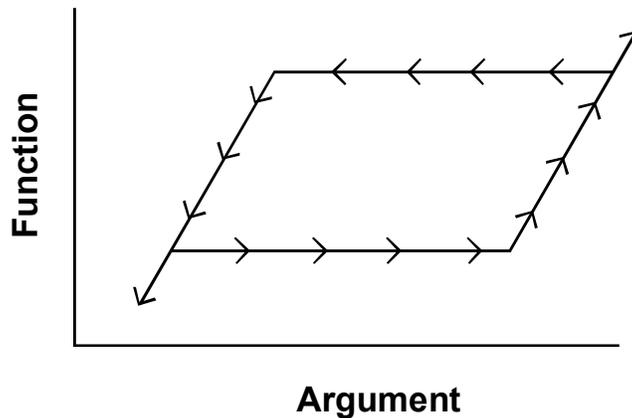


Figure 2.1 Hysteresis Function

The control function value lies between the loading curve, describing the limiting behavior for an increasing argument, and the unloading curve, describing the limiting behavior for a decreasing argument. The value changes only when it is “pushed up” by the loading curve or “pushed down” by the unloading curve. Thus, when argument and function value lie within the open loop of Figure 2.1, the function value remains unchanged until the

argument has either increased to encounter the loading curve or decreased to encounter the unloading curve. In either case, the function value will then follow the appropriate bounding curve.

This may be stated somewhat differently as: When the argument is increasing, the value of the hysteresis function is the maximum of its old value and that defined by the loading curve; when the argument is decreasing, the function takes the minimum of its old value and that given by the unloading curve. That is,

$$f^n = \begin{cases} \max [f^{n-1}, f_{\text{LOAD}}(a_1^n)], & a_1^n \geq a_1^{n-1} \\ \min [f^{n-1}, f_{\text{UNLOAD}}(a_1^n)], & a_1^n < a_1^{n-1} \end{cases} \quad (2.6)$$

The two functions, f_{LOAD} and f_{UNLOAD} , may be specified by tabular functions or other control functions. Two miscellaneous numbers are required; the first identifies the function defining the loading curve and the second identifies the function defining the unloading curve. A number ≥ 0 is interpreted as the number of a control function, while a number < 0 is interpreted as the negative of the number of a tabular function.

Note that if the loading and unloading curves are specified by control functions, it is the user's responsibility to ensure that either

- (1) these functions have the same argument as the hysteresis function, and lower function numbers (so that they are evaluated first), consistent with Equation (2.6)
- or
- (2) the user understands how the function will be evaluated (Equation (2.6) with the argument of the loading and unloading function *not* equal to a_1), and truly desires that result.

Necessary old values are maintained within the Control Function package database; proper specification of initial function values may be important to achieve the desired results.

2.1.5 Trips

The MELCOR Control Function package includes a variety of TRIP functions. Each requires a single argument. In one case, this argument is logical. In all others, the argument is real and either two or four miscellaneous numbers are required to define setpoints. The latter class of function is used to model switching phenomena involving a dead band (such as a heater which turns on when the temperature falls below T_1 and

CF Package Users' Guide

doesn't turn off until the temperature has risen above T_2 , after which it *remains* off until the temperature is again below T_1).

In simplest form, such a control has only two states, *off* and *on*, and could be represented as a logical function. MELCOR, however, employs a more general implementation where trips have been implemented as real-valued functions. The function value is zero if the trip is off and non-zero if it is on. There are two possible *on* states: *on-forward*, indicated by a positive function value, and *on-reverse*, indicated by a negative value. If a trip is on-forward or on-reverse, the magnitude of the function value is equal to the time (in seconds) since it was last in some different state, and can therefore be used for timing or delay purposes. The terms "forward" and "reverse" refer to the direction of counting time when a trip is on; in practice, either type of trip could be used with appropriate (positive or negative) scaling.

The simplest trip, with the short input name of TRIP, is a function of a single logical variable and requires no miscellaneous numbers. It is initially off (with value zero), and remains off until its argument changes state; thereafter, it is on-forward (with positive value) if its argument is true, or on-reverse (with negative value) if its argument is false, and can never again be off. When the trip is on, the magnitude of the function value is the time elapsed since the last change of state occurred. (For the timestep on which the change took place, this time is taken as roundoff compared to the timestep.)

Of the trips which are functions of a real variable, one of the simplest is TRIP-OFF-FORWARD, abbreviated as T-O-F for input purposes. (The name is intended to indicate that it is off for small argument values and on-forward for large ones.) Two setpoints, S_1 and S_2 , $S_1 < S_2$, divide the argument domain into a region where the trip is off, one where it is on-forward, and a dead band in between:

$$\begin{array}{ll} a_1 \leq S_1 & \text{trip is off} \\ S_1 < a_1 < S_2 & \text{state = previous state, off or on-forward} \\ S_2 \leq a_1 & \text{trip is on-forward.} \end{array}$$

The setpoints are input as miscellaneous numbers, in the order S_1, S_2 . If the trip is off, the function value is 0.0; if it is on-forward, the value is the time elapsed since it was last off. (For the timestep on which the change took place, the time at which the argument crossed the setpoint is estimated by linear interpolation.) The behavior may be compactly summarized as

T-O-F	off	S_1	dead-band	S_2	on-fwd
-------	-----	-------	-----------	-------	--------

Another simple case involving a real argument is TRIP-REVERSE-OFF, abbreviated as T-R-O. Here, two setpoints, S_1 and S_2 , $S_1 < S_2$, divide the argument domain into a region where the trip is on-reverse, one where it is off, and a dead-band in between:

$$a_1 \leq S_1 \quad \text{trip is on-reverse}$$

$S_1 < a_1 < S_2$ state = previous state, off or on-forward
 $S_2 \leq a_1$ trip is off.

If the trip is off, the function value is 0.0; if it is on-reverse, the value is the negative of the time elapsed since it was last off. (Again, the time at which the argument crossed the setpoint is estimated by linear interpolation.) This may be compactly summarized as

T-R-O	on-rev	S_1	dead-band	S_2	off
-------	--------	-------	-----------	-------	-----

The remaining cases, including several with four setpoints, which define two dead-bands, are constructed in exact analogy to the preceding two, and should be easily understood from the following summaries. (In all cases, the time at which the argument crosses a setpoint is estimated by linear interpolation.)

T-O-R	off	S_1	Dead-band	S_2	on-rev				
T-F-O	on-fwd	S_1	Dead-band	S_2	off				
T-O-F-O	off	S_1	Dead-band	S_2	on-fwd	S_3	dead-band	S_4	off
T-O-R-O	off	S_1	Dead-band	S_2	on-rev	S_3	dead-band	S_4	off
T-F-O-F	on-fwd	S_1	Dead-band	S_2	off	S_3	dead-band	S_4	on-fwd
T-R-O-R	on-rev	S_1	Dead-band	S_2	off	S_3	dead-band	S_4	on-rev
T-F-O-R	on-fwd	S_1	Dead-band	S_2	off	S_3	dead-band	S_4	on-rev
T-R-O-F	on-rev	S_1	Dead-band	S_2	off	S_3	dead-band	S_4	on-fwd

For each of these trips, the setpoints must be input as miscellaneous numbers in the order S_1 , S_2 , or S_1 , S_2 , S_3 , S_4 as appropriate. Necessary old data are maintained within the Control Function database. Because of the dead-band, the state of a trip may depend on past history. Therefore, proper definition of initial function values in MELGEN may be important to achieve the desired results.

2.1.6 User-Defined Functions

In order to simplify the addition of special-purpose functions, interfaces are provided for ten user-defined functions with names FUN1, FUN2, ..., FUN10. Each is coded as a function of five real arguments and includes an error flag in its calling sequence.

In standard versions of MELCOR, these functions are coded as dummies that do nothing but set their error flag if called. This will prevent the writing of a restart file by MELGEN if a user-defined function is referenced in its input.

In order to make use of the interfaces, the user must generate his own FORTRAN FUNCTION in the form

REAL FUNCTION FUNn (A1, A2, A3, A4, A5, IERROR)

The arguments A1 through A5, defined as for any other type of control function, may be used as desired in evaluating FUNn, but must not be changed by the FUNCTION routine. Five arguments *must* be specified. The value of FUNn is used as the control function value (before any user-specified scaling is performed). IERROR is returned as 1 if an error was encountered and as 0 otherwise. If the error flag is set to 1 during MELCOR execution, the CF package will halt execution and generate a restart dump. MELGEN and MELCOR must be linked with the new routine used to *replace* the dummy. The exact procedure for linking will, of course, be different at different computing sites.

2.2 Logical-Valued Control Functions

Logical-valued control functions are not, of course, amenable to scaling. They may, however, be modified to latch or to function as a one-shot. If a control function is classified as NORMAL, its value will always change in response to change(s) of its argument(s). In contrast, a control function with a LATCH or ONE-SHOT classification becomes uncoupled from its argument(s) after the first change from the initial (MELGEN) value of the function. For the LATCH, the new value will be retained forever after, independent of changes in the function argument(s); for the ONE-SHOT, the changed value is retained for one timestep only, after which the function reverts to its initial value on the next timestep and retains that value for the duration of the calculation.

Some logical-valued control function types take logical arguments; others take real arguments: in the latter case, the argument scaling discussed previously should be understood to apply.

In the list below, the order is not strictly alphabetic; similar functions have been grouped together. FORTRAN equivalents are shown in the belief that they will be more familiar than Boolean expressions.

Except as noted (for the arithmetic comparison functions, L-EQ, L-GT, L-GE, and L-NE) all arguments are logical.

Function Name	Input Type	# arg	Mathematical or FORTRAN definition	Miscellaneous Numbers
L-Equals	L-EQUALS	1	$f = a_1$	none
L-Not	L-NOT	1	$f = \text{.NOT. } a_1$	none
L-Equivalent	L-EQV	2	$f = a_1 \text{.EQV. } a_2$	none
L-Equal	L-EQ	2 ^[a]	$f = a_1 \text{.EQ. } a_2$	none
L-Greater Than	L-GT	2 ^[a]	$f = a_1 \text{.GT. } a_2$	none
L-Greater or Equal	L-GE	2 ^[a]	$f = a_1 \text{.GE. } a_2$	none

Function Name	Input Type	# arg	Mathematical or FORTRAN definition	Miscellaneous Numbers
L-Not Equal	L-NE	2 ^[a]	$f = a_1 .NE. a_2$	none
L-AND	L-AND	≥ 2	$f = a_1 .AND. \dots .AND. a_n$	none
L-OR	L-OR	≥ 2	$f = a_1 .OR. \dots .OR. a_n$	none
L-If Then Else	L-L-IFTE	3	IF (a ₁) THEN f = a ₂ ELSE f = a ₃ END IF	none

[a] Arguments of the arithmetic comparison functions are real.

3. User Input Requirements

A description of all control functions must be provided as part of MELGEN input. Certain elements of the description may be changed on restart through input to MELCOR.

In all versions of MELCOR through 1.8.4, control function numbers were restricted to three digits by the format of the input records. This limited the number of control functions that could be defined in any calculation to 999. The restriction has been eliminated, and control function numbers can now be as long as eight digits. The number of control functions that can be defined in a calculation is thus effectively limited only by the computer memory available and the endurance of the user.

In a few cases, MELCOR packages that use control functions store the function numbers as REAL variables rather than as INTEGERS. In such uses, a control function number is not acceptable unless it is exactly representable as a REAL. For 32-bit IEEE representation, the smallest INTEGER not exactly representable is $2^{24}+1 = 16777217$. Above this value, only even numbers are representable through $2^{25} = 33554432$, only multiples of 4 through $2^{26} = 67108864$, etc. Thus, any 7-digit number will be acceptable, but only some 8-digit numbers. The input variables and records involved are:

- (1) A loading or unloading function number for a HYST control function input as a "miscellaneous number" on a CFn...n03 or CFn...n04 record in the CF Package, if the value is > 0 indicating that it is a control function number;
- (2) The burn completeness parameters, ICCFLG and ICCDCH, input on BURCCXX records, and flame speed parameters IFSFLG and IFSDCH, input on BURFSXX records in the BUR Package, if the values are > 0 indicating that they are control function numbers;
- (3) A vent valve delay time, VNSTP(3), input on an ESFCND0800 input record, if the value is < 0 indicating that it is a control function number;

CF Package Users' Guide

- (4) A structural failure control function number for supporting structure input as ISSLCF input on a CORxxxSS input record (here xxx can be 000, Zjj, Rii, or ijj) in the COR package;
- (5) A laminar flow coefficient, SLAM, input on an FlnnnSk record in the FL Package, if the value is < 0.0 indicating that it is a control function number.

In each of these cases, a test of the actual input is made, and if the function number is unacceptable, an error message such as

“MISC INTEGER CANNOT BE MAPPED INTO REAL”

or

“CONTROL FUNCTION NUMBER NOT EXACTLY REPRESENTABLE AS REAL”

is generated.

In addition, although MELCOR writes control function numbers to the plot file as integers, the plot program HISPLTM compares them internally to desired indices stored as REALs. Therefore, some 8-digit CF numbers can't be plotted by HISPLTM or, presumably, by any other plot program that has copied its data acquisition logic.

3.1 MELGEN User Input

The user input for the Control Function package is described below. One set of records is required for each control function; the actual records required differ for different types of control functions (see Section 2).

Following general MELCOR practice, the number associated with a control function is embedded in the input record identifiers of all records associated with that control function. Throughout this section, the control function number is shown as “n...n”, which may represent a 3- to 8-digit number, and must be greater than 0.

Older versions of MELCOR required that the number in the record identifier contain exactly three digits, with leading zeroes for functions with numbers less than 100. For example, any input record of the form CF005xx refers to control function 5, while one of the form CF873 refers to function 873. In order to maintain compatibility with older input decks, the code currently requires that *at least* three digits be included in the record identifier, with leading zeroes *required* for control functions with numbers less than 100. However leading zeroes are *not permitted* for function numbers greater than 999. (This greatly simplifies the input processing by ensuring that all records pertaining to a given control function have record identifiers of the same length.) Thus, a record identifier of the form CF12345678xx refers to control function 12345678, while one of the form CF0009999 is invalid. The

presence of such an invalid record identifier is fatal; MELGEN will exit after first-pass processing, without generating a restart file.

Unless otherwise stated, if the field variable name in a record starts with I through N, it is an integer while if it begins with A through H or O through Z it is a real number.

CFn...n00 – Control Function Definition Record

n...n is the 3- to 8-digit control function number

Required

This record defines the control function name, type, number of arguments, and the multiplicative and additive scaling constants. The allowed types are described in Section 2. Scaling of logical functions is not possible, but the scale factor and constant (CFSCAL and CFADCN) must still be input; they are not used.

The value of the control function is given by

$$\text{CFSCAL} * \text{Function Value} + \text{CFADCN}$$

- (1) CFNAME - User-defined control function name.
(type = character*16, default = none)
- (2) CFTYPE - Control function type (from Section 2).
(type = character*8, default = none)
- (3) NCFARG - Number of arguments. Must agree with description in Section 2.
(type = integer, default = none)
- (4) CFSCAL - Multiplicative scale factor for control function.
(type = real, default = none, units = dimensionless)
- (5) CFADCN - Additive constant for control function.
(type = real, default = 0.0, units = same as control function)

CFn...n01 – Initial Value of Control Function

n...n is the 3- to 8-digit control function number

Optional

The initial value of a control function may be either user-specified via this record, or calculated from the initial database. In cases where the control function involves "history" (depends on past value), it should be user-specified to assure the desired result. In other cases, identified in the Users' Guide for the package involved, it should be specified to allow initialization of the package's database.

CF Package Users' Guide

The value input, real or logical, must be appropriate to the function type.

- (1) CFVALR - Real initial value if real-valued control function.
(type = real, default = none, units = same as control function)
- or
- (2) LCFVAL - Logical initial value if logical-valued control function.
(type = logical, default = none, units = dimensionless)

CFn...n02 – Upper and Lower Bounds

n...n is the 3- to 8-digit control function number

Optional

Upper and lower bounds may be specified for a real-valued control function. If no bounds are input, the function values are limited only by the internal representation of real numbers on the computer used.

- (1) ICFLIM - Switch specifying bounds input.
(type = integer, default = 0, units = dimensionless)
= 0 no bounds input (default)
= 1 only lower bound input
= 2 only upper bound input (field 2 must be present, but is unused)
= 3 both bounds input
- (2) CFLIML - Lower bound, required for ICFLIM = 1, 2, or 3, but unused for ICFLIM = 2.
(type = real, default = none, units = same as control function)
- (3) CFLIMU - Upper bound, required for ICFLIM = 2 or 3.
(type = real, default = none, units = same as control function)

CFn...n0k – Miscellaneous Numbers

n...n is the 3- to 8-digit control function number

$3 \leq k \leq 4$, k used for ordering

Required for some types of control functions

Some control functions require input of “miscellaneous numbers” as noted in Section 2. More than one may be input on a single record; two records may be used if required.

- (1) FIELDS - Appropriate integers or reals required for control function n...n, as described in Section 2.

(type = real or integer, default = none, units depend on application)

CFn...n05 – Logical Control Function Classification

n...n is the 3- to 8-digit control function number

Optional

A logical-valued control function may be converted to a “latch” or “one-shot” as described in Section 2.2. A “NORMAL” control function has a value simply defined by its argument(s). A control function with a LATCH classification will change state only once, retaining its new value thereafter; one with a ONE-SHOT classification will change state for one timestep only, reverting to its initial value thereafter.

- (1) CLASS - Classification. May be 'NORMAL', 'LATCH', or 'ONE-SHOT'.
(type = character*8, default = 'NORMAL')

CFn...n06 – Logical Control Function Switching Message

n...n is the 3- to 8-digit control function number

Optional

A logical-valued control function may be made to write a message to the output file, the special message file, and/or the log file whenever its state changes.

- (1) MSGFIL - Flag indicating files to which the message is written.
(type = integer, default = 0, units = dimensionless)
= 0 do not write message
= 1 write message to standard output file only
= 2 write message to standard output file, special message file, and log file, but only if timestep is successfully completed.
- (2) SWTMSG - Switching message.
(type = character*64, default = none)

CFn...nkk – Control Function Arguments

n...n is the 3- to 8-digit control function number

$10 \leq kk \leq ZZ$, kk is used for ordering

Required

Each control function requires one or more arguments. Each argument is constructed from an element in the database. The type (REAL or LOGICAL) of each argument is checked in MELGEN, and an error is generated if it does not match that expected for the given control function type (CFTYPE on record CFn...n00). Scaling of real arguments is allowed, and the value of argument is given by

ARSCAL * Database Element + ARADCN

Scaling of logical arguments is not possible, but the scale factor and constant must still be input; they are not used.

- (1) ARSCAL - Multiplicative scale factor.
(type = real, default = none, units = dimensionless)
- (2) ARADCN - Additive constant.
(type = real, default = 0.0, units = same as database element)
- (3) CHARG - Database element identifier. Refer to the Users' Guides for the various packages for permitted values.
(type = character*24, default = none)

3.2 MELCOR User Input

Some of the data associated with an existing control function may be changed when running MELCOR. Note that new control functions may *not* be added, nor may the *type* of an existing control function be changed. The MELCOR input is a subset of the MELGEN input described in Section 3.1; record formats are identical in both cases.

CFn...n00 – Control Function Definition Record

n...n is the 3- to 8-digit control function number

Required

This record defines the control function name, type, number of arguments, and the multiplicative and additive scaling constants. The allowed types are described in Section 2. The control function type and the number of arguments must agree with the original values input to MELGEN on the CFn...n00 record. The user-defined name and scaling parameters, however, may be changed as desired.

CFn...n01 – Initial Value of Control Function

n...n is the 3- to 8-digit control function number

Optional

This record may be used to override the value of a control function in the MELCOR database at the time of restart. At the first execution of MELCOR, it will replace the value initialized in MELGEN; for later restarts, it will replace values calculated by the CF package on the last previous MELCOR timestep.

If this record is not input, the control function will not be reevaluated until *after* the first timestep is completed. Therefore, if record CFn...n00 or CFn...n02 is present

in the MELCOR input, but record CFn...n01 is not, a warning message is generated.

CFn...n02 – Upper and Lower Bounds

n...n is the 3- to 8-digit control function number
Optional

This record may be used to replace one or both of the bounds on the value of a control function which are stored in the MELCOR database at the time of restart, whether the old limits were default or specified by input to MELGEN or a previous MELCOR run.

4. Sensitivity Coefficients

There are no sensitivity coefficients associated with the Control Function package.

5. Plot Variables and Control Function Arguments

The variables in the Control Function package which may be used for plot variables and control function arguments are listed and described below. The control function arguments are denoted by a "c," the plot variables by a "p," within slashes ("/") following the variable name.

CFVALU.n /cp/ Value of control function n. At present, only real-valued control functions can be plotted.

Control functions are evaluated in the order of their user-assigned numbers. Therefore, if the value of control function n is specified as an argument for control function m, the new value of function n will be used if $m > n$ while the old value will be used if $m \leq n$.

6. Example Input

The following examples illustrate input to the Control Function package, and suggest ways in which the package may be used.

6.1 Constants

A constant value may be defined in several ways. Two simple ones, using scale factors and offsets on the function or on its arguments, are:

```
CF00100  PI    EQUALS    1    0.0  3.1416 *Constant value 3.1416
```

CF Package Users' Guide

```
CF00110  1.0  0.0  TIME                                *(Argument irrelevant)
```

or

```
CF00100  PI    EQUALS    1    1.0                    *Constant value 3.1416
CF00110  0.0   3.1416    TIME                                *(Argument irrelevant)
```

In either case, the actual argument specified is irrelevant so long as it is a valid control function argument.

6.2 Pressure and Head Differences

The difference in pressure between two control volumes, 200 and 300, perhaps representing primary and containment, can be evaluated as:

```
CF00500  DELTA-P      ADD    2    1.0                    *Pressure difference
CF00510   1.0  0.0  CVH-P.200                          *Press in volume 200
CF00511  -1.0  0.0  CVH-P.300                          *Press in volume 300
```

Here CVH-P.nnn is a control function argument in the CVH package, and the pressures used will correspond to the pool surface.

If the static head in the pool in volume 200 is important, it could be included by modifying CF005 as follows:

```
CF00500  DELTA-P      ADD    3    1.0                    *Pressure difference
CF00510   1.0  0.0  CVH-P.200                          *Press in volume 200
CF00511   1.0  0.0  CFVALU.4                          *Pressure head in pool
CF00512  -1.0  0.0  CVH-P.300                          *Press in volume 300
*
CF00400  POOL-HEAD    MULTIPLY  2    9.81                *Pressure head in pool
CF00410   1.0  0.0  CVH-RHOP.200                      *Pool density
CF00411   1.0  0.0  CFVALU.3                          *Pool depth over junction
*
CF00300  POOL-DEPTH    DIM    2    1.0                    *Pool depth over junction
CF00310   1.0  0.0  CVH-LIQLEV.200                    *Pool surface elevation
CF00311   0.0  3.57  TIME                              *Junction elevation=3.57m
```

Here CVH-RHOP.nnn and CVH-LIQLEV.nnn are the pool density and the pool-surface elevation, respectively, in volume nnn in the CVH package. The junction elevation is 3.57m (specified as $0.0 \cdot \text{TIME} + 3.57$ in CF004), and the acceleration of gravity (9.81 m/s^2) appears as the scale factor in CF003. The relative ordering of the control function numbers must be as shown so that the value of the pool depth (CF003) is current before it is multiplied by the pool density in CF004, which is done before the head is added to the

thermodynamic pressure in CF005. Note that it is the number associated with a control function that is significant and not the position of its definition in the input deck.

Clearly, all other head terms, pool and atmosphere, could be similarly included as additional arguments to the ADD function CF005.

6.3 Valve Control

The pressure differential given by CF005 above could be used to initiate a primary system failure by opening a valve in an initially closed flow path between volumes 200 and 300. For example, the differential could be used to actuate a trip by defining CF010 input such as

```
CF01000 RUPTURE      T-O-F      1      1.0      *Trip on pressure diff
CF01001      0.0      *Initial value (off)
CF01003     -2.0E7      *Lower (off) setpoint
CF01004      2.0E7      *Upper (on-fwd) setpoint
CF01010      1.0  0.0      CFVALU.5      *Pressure difference
```

Of course, the use of this trip to actuate the valve must be established in the FL package, i.e., by entering the control function number 10 as variable 1 on record FLnnnVk.

The valve state would not change from its initial value (closed) until the pressure differential rose above 20.0 MPa and the control function tripped "on-forward." Then, the open fraction would be given by the Control Function specified by variable 2 on the FLnnVk record. (If the differential dropped below the lower setpoint, the open fraction would remain constant with its last calculated value.)

After CF010 trips on-forward, its value will be the time (in seconds) since the trip. Thus the open fraction could be made a function of time after failure by simply defining the second control function as a function of CFVALU.10.

Alternatively, the pressure differential could be used to control the operation of a relief valve. In this case, no trip would be used (variable 1 negative on the FLnnnVk record), and the control function whose number was entered as variable 2 on the FLnnnVk record might define the open fraction of the relief valve area through the hysteresis function

```
CF02000 VALVE-FRACT HYST 1      1.0      *Open fract of valve area
CF02001      0.0      *Initial value (closed)
CF02003     -100      *TF100 for loading
CF02004     -200      *TF200 for unloading
CF02010      1.0  0.0  CFVALU.5      *Pressure difference
```

The opening characteristic could be made extremely detailed through appropriate definition of the tabular functions, and multiple valves included so long as their dead-bands do not

CF Package Users' Guide

overlap. (If the dead-bands of a bank of relief valves *do* overlap, this may be accounted for by summing two or more hysteresis functions of the form of CF020, using a higher-numbered ADD control function.)

6.4 Messages and Other Output

In either of the cases in Section 6.2, the times at which the valve opened and closed could be recorded in output files using switching messages from logical functions. For example, assuming that the trip option (using CF010) was used,

```
CF10000 VALVE-OPEN  L-GT 2      1.0      *True if valve open
CF10001 .FALSE.      *Initially false
CF10006  2          'VALVE OPENED OR CLOSED' *To out, log, message
CF10010  1.0  0.0  CFVALU.10 *Valve trip
CF10011  0.0  0.0  TIME      *Zero
```

would produce a message every time the valve opened or closed. Inclusion of the record

```
CF10005          LATCH          *First opening only
```

would terminate messages after the first opening of the valve, in which case the message itself might better be

```
CF10006  2          'FIRST OPENING OF VALVE' *To out, log, message
```

Finally, the control function

```
CF11000 EDIT-FLAG  L-EQUALS  1      1.0    *For record EDITCF
CF11001 .FALSE.      *Initially false
CF11005 ONE-SHOT      *True at most once
CF11010  1.0  0.0  CFVALU.100 *Valve open
```

could be used to trigger a special (complete) edit the first time the valve opened through use of the EDITCF record described in the MELCOR/MELGEN Users' Guide. The ONE-SHOT classification prevents more than one such edit from being produced.

6.5 New Output Variables

The values of control functions are included in printed edits and in plot dumps. Thus, the pressure differences, etc., defined in Section 6.2 would appear directly in the printed output of MELCOR and could be plotted as functions of time if desired. The values printed could, of course, be obtained by hand calculations from other edited quantities, but the plots could not be obtained in any other way (using the current version of HISPLT).

Many variables, particularly in the RadioNuclide (RN) package, are available as control function arguments but not as plot variables. This is because the size and complexity of the database makes it impractical to include all quantities in the plot file. In versions of MELCOR through 1.8.4, these variables could be plotted only by defining a control function "EQUAL" to the argument, and then plotting the value of that control function. This is no longer necessary, because the values of specified control function arguments can be added directly to the plot file using PLOTxxx input records without defining a control function. In addition to simplification of the input, this allows the variables to be plotted using their mnemonic names rather than arbitrary control function numbers. See the Executive (EXEC) Package Users' Guide for details.

There will still be cases where the user will need to define a control function simply to make available the value of a quantity which would otherwise not appear in the output, but only when some arithmetic or logic is involved. As a simple example, involving hydrodynamic variables, the peak pressure in a control volume could be evaluated by a single control function as

```
CF13000 PEAK-PRESS MAX 2 1. *Peak pressure in cv 202
CF13001 0.0 *Initialize to 0.0
CF13010 1.0 0.0 CVH-P.202 *Pressure in cv 202
CF13011 1.0 0.0 CFVALU.130 *Previous peak
```

which evaluates the maximum of the pressure in control volume 202 and the previous value of that maximum.

Another way to do this, and to capture the time at which the peak occurred, is

```
CF14000 NEW-GT-OLD L-GT 2 1. 0. *True if p .gt. old max
CF14001 .TRUE. *Initialize true
CF14010 1.0 0.0 CVH-P.202 *Pressure in cv202
CF14011 1.0 0.0 CFVALU.141 *Previous peak press
*
CF14100 PEAK-PRESS L-A-IFTE 3 1. *Peak pressure in cv202
CF14110 1.0 0.0 CFVALU.140 *New .gt. old peak
CF14111 1.0 0.0 CVH-P.202 *Pressure in cv202
CF14112 1.0 0.0 CFVALU.141 *Previous peak press
*
CF14200 T-AT-PEAK L-A-IFTE 3 1. *Time at peak pressure
CF14210 1.0 0.0 CFVALU.140 *New .gt. old peak
CF14211 1.0 0.0 TIME *Time
CF14212 1.0 0.0 CFVALU.142 *Time at previous peak
```

Since CF140 is given the initial value .TRUE., CF141 and CF142 will be properly initialized to the values of their second arguments, CVH-P.202 and TIME, respectively. The uninitialized third arguments (the previous values of these control functions themselves) will not be used. Otherwise, the user would have to initialize (correctly) both CF141 and

CF Package Users' Guide

CF142, in which case no initial value for CF140 would be needed or used. The values of all three control functions would be available in printed output. The real-valued ones could also be plotted if desired.

6.6 Voting Logic

Control logic may be simulated in many ways, and there are often tricks that employ far fewer control functions than a simple translation of the logic. For example, a two-of-four voting control could be modeled using 4 logical functions for the individual channels, 6 more to test all combinations of two signals true, an 11th to "OR" these 6 possibilities, and (usually) a 12th to generate a real value from this final logical one.

A more compact, but less obvious, implementation is shown below for the case where the individual channels involve detecting a pressure greater than a threshold.

```
* Test each channel. Value =1 if Pn.ge.thresh, else =0
*
CF15100 P1.GT.TH      SIGNI      1   .5  .5  *1 or 0 form p1 test
CF15110   1.0 -1.02E5  CV-P.1                *1.02E5 is thresh
*
CF15200 P2.GT.TH      SIGNI      1   .5  .5  *1 or 0 from p2 test
CF15210   1.0 -1.02E5  CVH-P.2
*
CF15300 P3.GT.TH      SIGNI      1   .5  .5  *Same for p3
CF15310   1.0 -1.02E5  CVH-P.3
*
CF15400 P4.GT.TH      SIGNI      1   .5  .5  *Same for p4
CF15410   1.0 -1.02E5  CVH-P.4
*
* Vote 2 of 4. Result is 0.0 or 1.0
*
CF16000 2-OF-4 ADD   4     2.   -2.5 Value = 2*N - 2.5
CF16002   3     0.0   1.0                *Limit between 0 and 1
CF16011   1.0   0.0   CFVALU.151        *1 or 0 for channel 1
CF16012   1.0   0.0   CFVALU.152        *1 or 0 for channel 2
CF16013   1.0   0.0   CFVALU.153        *1 or 0 for channel 3
CF16014   1.0   0.0   CFVALU.154        *1 or 0 for channel 4
```

Note the usage of multiplicative and additive constants and of limits. For each channel (CF15n) the function value is -1.0 or $+1.0$ before scaling, and 0.0 or 1.0 after.

In the "vote" conducted by CF160, for the case of

0	1	2	3	4	pressures over threshold,
---	---	---	---	---	---------------------------

the function value is:

0.0	1.0	2.0	3.0	4.0	before scaling,
-2.5	-0.5	+1.5	+3.5	+5.5	after scaling, and
0.0	0.0	1.0	1.0	1.0	after limits are imposed.

CF160 could now be used to initiate a trip and start a timer. For example:

```
CF17000 VOTE-TRIP    T-O-F      1      1.0    *Trip on 2-of-4 vote
CF17001      0.0                                *Initially value (off)
CF17003      0.1                                *Lower (off) setpoint
CF17004      0.9                                *Upper (on) setpoint
CF17010      1.0  0.0  CFVALU.160              *2-of-4
```

Note that the scaling and bounding of control functions 151–154 and 160 are not essential if their only use is to actuate the trip; the setpoints of CF170 could be changed to work correctly with the unscaled, unbounded values. However, the scaling costs little and makes it easier to check the control logic.

6.7 Homologous Pump Model

The QUICK-CF Pump model in the Flow Path package uses a control function to calculate the pressure developed by a pump. The following input indicates how a control function can be constructed to yield a homologous pump model similar to that used in TRAC and RELAP.

```
* Homologous pump model for flow path 2
* CF203 = delta-p = density*grav*speed**2 * fcn (flow/speed)
*
CF20000 SPEED    any real CF                *(Normalized) pump speed
CF200..
*
CF20100 FLOW/SPEED  DIVIDE    2      1.    *(Normalized) flow/speed
CF20110      1.0  0.0  CFVALU.200          *Speed
CF20111      0.05 0.0  FL-VELLIQ.2        *1/.05 is v at rated flow
*
CF20200 PUMP-HEAD  TAB-FUN    1      1.    *Fcn (flow/speed) in m
CF20203  202                                *Pump curve
CF20210      1.0  0.0  CFVALU.201          *(Normalized) flow/speed
*
TF20200 PUMP-CURVE  n      1.    0.    *Table of n points
*   Norm flow/speed Head (m)
TF20210      a.a                                x.x
TF202..      b.b                                y.y
*
CF20300 DELTA-P    MULTIPLY  4      9.81  *9.81 is grav
CF20310      1.0  0.0  CVH-RHOP.3         *Upstream volume is 3
```

CF Package Users' Guide

CF20311	1.0	0.0	CFVALU.202	*Head in m
CF20312	1.0	0.0	CFVALU.200	*Speed/nom_speed
CF20313	1.0	0.0	CFVALU.200	*Speed/nom_speed

The input shown is appropriate for the normal region of pump operation (positive rotation, forward flow), when the fluid is liquid only. The calculated pressure boost will be far too great for two-phase or vapor flow; because the control function defines the pressure directly rather than as a head, it must be reduced in these cases by the density ratio (in addition to any two-phase degradation factor). To do so, the density used as the first argument of CF203 should be generalized to represent the actual flow density multiplied, if desired, by a two-phase degradation term. As a first approximation, it could simply be set to zero if the flow path void fraction—available as control function argument FL-VOID.nnn—is non-zero. It might be necessary to incorporate some time-averaging through integration if oscillations are observed.

In RELAP, the head curve in the normal pump region is input in two parts, using

$$\begin{aligned} \text{HEAD} &= \text{SPEED}^{**2} * \text{FCN1}(\text{FLOW}/\text{SPEED}) && \text{for } \text{FLOW}/\text{SPEED} < 1 \text{ and} \\ \text{HEAD} &= \text{FLOW}^{**2} * \text{FCN2}(\text{SPEED}/\text{FLOW}) && \text{for } \text{SPEED}/\text{FLOW} < 1. \end{aligned}$$

(the former is referred to as “HAN” and the latter as “HVN”), but both may be accommodated in CF202. The curve can be further extended to the energy dissipation region (positive rotation, negative flow), corresponding to “HAD” and “HVD.” In this case, the density argument in CF203 should also reflect the direction of flow.

The input assumes that the rotational speed of the pump is available as a control function value. In general, this will be a tabular function of time, but a truly determined user could calculate it by integration of the angular momentum equation for a pump rotor, using still more control functions and the homologous torque curves. If reversal of the pump is anticipated, it is necessary (at least) to avoid the possibility of division by zero in CF201. This is probably best done by explicitly using the alternative representation of the head curve in those regions where $\text{SPEED}/\text{FLOW} < 1$ (“HVN,” etc.).