SAND--89-8009

DE90 000917

SAND89-8009
Unlimited Release
Printed September 1989

CHEMKIN-II: A FORTRAN CHEMICAL KINETICS PACKAGE FOR THE ANALYSIS OF GAS-PHASE CHEMICAL KINETICS

Robert J. Kee and Fran M. Rupley
Computational Mechanics Division

and

James A. Miller
Combustion Chemistry Division

Sandia National Laboratories
Livermore, CA 94551

ABSTRACT

This document is the user's manual for the second-generation Chemkin package. Chemkin is a software package whose purpose is to facilitate the formation, solution, and interpretation of problems involving elementary gas-phase chemical kinetics. It provides an especially flexible and powerful tool for incorporating complex chemical kinetics into simulations of fluid dynamics. The package consists of two major software components: an Interpreter and a Gas-Phase Subroutine Library. The Interpreter is a program that reads a symbolic description of an elementary, user-specified chemical reaction mechanism. One output from the Interpreter is a data file that forms a link to the Gas-Phase Subroutine Library. This library is a collection of about 100 highly modular Fortran subroutines that may be called to return information on equation of state, thermodynamic properties, and chemical production rates.

ACKNOWLEDGMENTS

This new version of Chemkin has benefited greatly from the many researchers who have applied it, reported their experiences, and suggested improvements. We also appreciate our interactions with those who have developed and shared new applications for the software. Although it is impractical to acknowledge each of those who have either directly or indirectly influenced the evolution of Chemkin, we believe it is important to single out our colleague Michael Coltrin, who has been an active contributor throughout the Chemkin-II project. Also, many discussions with Juergen Warnatz have influenced several aspects of the software, including the design of the data structures to promote vectorization.

CONTENTS

NOMENCLATURE

		CGS Units
a_{nk}	Coefficients to fits of thermodynamic data	depends on n
a_k^o	Standard state specific Helmholtz free energy for the k^{th} species	ergs/g
$\overline{a}$	Mean Helmholtz free energy of a mixture	ergs/g
A_k^o	Standard state Helmholtz free energy for the k^{th} species	ergs/mole
$\overline{A}$	Mean Helmholtz free energy for a mixture	ergs/mole
A_i	Pre-exponential factor in the rate constant of the i^{th} reaction	depends on reaction
c_{p_k}	Specific heat at constant pressure of the k^{th} species	ergs/(g K)
$\overline{c}_p$	Mean specific heat at constant pressure	ergs/(g K)
$C_{p_k}^o$	Standard state specific heat at constant pressure of the k^{th} species	ergs/(mole K)
C_{p_k}	Specific heat at constant pressure of the k^{th} species	ergs/(mole K)
$\overline{C}_p$	Mean specific heat at constant pressure	ergs/(mole K)
c_{v_k}	specific heat at constant volume of the k^{th} species	ergs/(g K)
$\overline{c}_v$	Mean specific heat at constant volume	ergs/(g K)
C_{v_k}	Specific heat at constant volume of the k^{th} species	ergs/(mole K)
$\overline{C}_v$	Mean specific heat at constant volume	ergs/(mole K)
$\dot{C}_k$	Chemical creation rate of the k^{th} species	moles/(cm^3 sec)
$\dot{D}_k$	Chemical destruction rate of the k^{th} species	moles/(cm^3 sec)
E_i	Activation energy in the rate constant of the i^{th} reaction	[cal/mole]*
g_k^o	Standard state specific Gibbs free energy for the k^{th} species	ergs/g
$\overline{g}$	Mean Gibbs free energy of a mixture	ergs/g
G_k^o	Standard state Gibbs free energy for the k^{th} species	ergs/mole

*By default, Chemkin uses activation energies in calories instead of ergs.

		CGS Units
$\overline{G}$	Mean Gibbs free energy of a mixture	ergs/mole
h_k	Specific enthalpy of the k^{th} species	ergs/g
$\overline{h}$	Mean specific enthalpy of a mixture	ergs/g
H_k^o	Standard state enthalpy of the k^{th} species	ergs/mole
H_k	Enthalpy of the k^{th} species	ergs/mole
$\overline{H}$	Mean enthalpy of a mixture	ergs/mole
i	Reaction index	
I	Total number of reactions	
k	Species index	
k_{f_i}	Forward rate constant of the i^{th} reaction	depends on reaction
k_{r_i}	Reverse rate constant of the i^{th} reaction	depends on reaction
K	Total number of species	
K_{c_i}	Equilibrium constant in concentration units for the i^{th} reaction	depends on reaction
K_{p_i}	Equilibrium constant in pressure units for the i^{th} reaction	depends on reaction
$[M]$	Total concentration of a mixture	moles/cm^3
N	Number of coefficients in polynomial fits to C_p^o/R	
P	Pressure	dynes/cm^2
P_{atm}	Pressure of one standard atmosphere	dynes/cm^2
q_i	Rate of progress of the i^{th} reaction	moles/(cm^{3}sec)
R	Universal gas constant	ergs/(mole K)
R_c	Universal gas constant, in same units as activation energy E_i	[cal/(mole K)]
s_k^o	Standard state specific entropy of the k^{th} species	ergs/(g K)
$\overline{s}$	Mean specific entropy of a mixture	ergs/(g K)

		CGS Units
S_k^o	Standard state entropy of the k^{th} species	ergs/(mole K)
S_k	Entropy of the k^{th} species	ergs/(mole K)
$\overline{S}$	Mean entropy of a mixture	ergs/(mole K)
T	Temperature	K
u_k	Specific internal energy of the k^{th} species	ergs/g
$\overline{u}$	Mean specific internal energy of a mixture	ergs/g
U_k	Internal energy of the k^{th} species	ergs/mole
$\overline{U}$	Mean internal energy of a mixture	ergs/mole
Y_k	Mass fraction of the k^{th} species	
X_k	Mole fraction of the k^{th} species	
$[X_k]$	Molar concentration of k^{th} species	moles/cm^3
W_k	Molecular weight of k^{th} species	g/mole
$\overline{W}$	Mean molecular weight of a mixture	g/mole

GREEK

α_{ki}	Enhanced third body efficiencies of the k^{th} species in in the i^{th} reaction.	
β_i	Temperature exponent in the rate constant of the i^{th} reaction.	
ρ	Mass density.	g/cm^3
τ_k	Characteristic chemical destruction time of the k^{th} species.	sec
ν_{ki}	Stoichiometric coefficients of the k^{th} reaction, $\nu_{ki} = \nu''_{ki} - \nu'_{ki}$.	
ν'_{ki}	Stoichiometric coefficients of the k^{th} reactant species in the i^{th} reaction.	
ν''_{ki}	Stoichiometric coefficients of the k^{th} product species in the i^{th} reaction.	
$\dot{\omega}_k$	Chemical production rate of the k^{th} species.	mole/(cm^3 sec)

CHEMKIN–II: A FORTRAN CHEMICAL KINETICS PACKAGE FOR THE ANALYSIS OF GAS-PHASE CHEMICAL KINETICS

I. INTRODUCTION

The Chemkin package is one of three basic elements in a large and growing body of software designed to facilitate simulations of elementary chemical reactions in flowing systems. The other major elements are the transport property package[1,2] and the surface chemistry package.[3] These packages should not be considered "programs" in the ordinary sense. That is, they are not designed to accept input, solve a particular problem, and report the answer. Instead, they are software tools intended to help a user work efficiently with large systems of chemical reactions and develop Fortran representations of systems of equations that define a particular problem. It is up to the user to solve the problem and interpret the answer. A general discussion of this structured approach for simulating chemically reacting flow can be found in Kee and Miller.[4]

An important advantage of the general-purpose and problem-independent structure of Chemkin is that it allows the analyst to work with the same chemical input regardless of the particular problem. Thus there is no need to remember a different input protocol for different problems, and consequently, the time required to switch between problems or to develop a new application is minimized. Additionally, by making Chemkin easily transportable between computers, we hope to facilitate the exchange of applications codes between different sites. Often such exchanges are hampered by machine-dependent or problem-specific coding.

Background

Chemkin–II is a revised, improved version of Chemkin. The original Chemkin[5] was published in 1980 and has remained essentially unchanged until recently. Over the past year, however, we have completely rewritten the package to expand its capabilities. The most important new capability is an accurate and efficient means of describing pressure-dependent reactions. The rate laws for reactions of this type do not follow the modified Arrhenius form that is required in the original Chemkin. Other new capabilities include a Landau-Teller form of the rate expression for vibrational energy transfer processes, a capability for specifying more than one rate expression for a reaction, and a capability for explicitly specifying an Arrhenius expression for the reverse rate of a reversible reaction. We have also restructured the internal data storage and rewritten many of the computational algorithms to facilitate vectorization on computers like the Crays.

Although new features have been added, Chemkin–II omits some capabilities that were included in the original Chemkin. The most important of these is the elimination of the many partial-derivative subroutines. These subroutines were intended to help form the Jacobian matrices that are needed for the computational solution of stiff differential equations. In ten years of using Chemkin, however, we found that we never used the partial-derivative capability. This is because we develop and apply computational algorithms that rely on approximate finite-difference Jacobians rather than exact analytic Jacobians. Furthermore, the inclusion of the pressure-dependent reaction capability makes deriving and implementing the partial derivative capabilities much more complex. Therefore, we decided that the effort to provide this little-used capability was not warranted.

The two packages are nearly compatible, although not entirely so. The original Chemkin handled all character-string manipulations through the Hollerith data type. Under the Fortran–66 standard that was predominant in 1980, Hollerith was the only standard way to deal with string information. However, the Fortran–77 standard is now universally accepted, and it does not recognize Hollerith data type, but replaces it with the much more powerful character data type. Therefore, Chemkin–II has eliminated Hollerith data type and is based entirely on character data.

We have included several new utility subroutines for manipulating character strings. Such capabilities are useful in writing the input and output sections of a new Chemkin application program. For example, in setting initial conditions for a species, it is useful to have a function that can read a character string containing a species name and a floating-point number. Subroutine CKSNUM will parse such a string into a species index number and a floating-point number. Section 15 of Chapter V describes several such utility routines.

Structure and Use of Chemkin

The Chemkin package is composed of two blocks of Fortran code and two files:

- the Interpreter (code)
- the Gas-Phase Subroutine Library (code)
- the Thermodynamic Database (file)
- the Linking File (file).

To apply Chemkin to a problem, the user first writes a Fortran program that describes his particular set of governing equations. This programming is minimal since the user need only call Chemkin subroutines which define the terms in his equations that relate to equation of state, chemical production, and thermodynamics, and combine the result to define his problem.

Next, the user runs the Interpreter, which first reads the user's symbolic description of the reaction mechanism and then extracts the appropriate thermodynamic information for the species involved from the Thermodynamic Database.[6] The database has exactly the same format as that used by the NASA complex chemical equilibrium code by Gordon and McBride.[7] The output of the Interpreter is the Linking File, which contains all the pertinent information on the elements, species, and reactions in the mechanism.

The Linking File is read by an initialization subroutine that is called from the user's code. The purpose of the initialization is to create three data arrays (one integer, one floating point, and one character data type) for use internally by the other subroutines in the Gas-Phase Subroutine Library.

The Gas-Phase Subroutine Library has over 100 subroutines that return information on elements, species, reactions, equations of state, thermodynamic properties, and chemical production rates. Generally, the input to these routines will be the state of gas—pressure or density, temperature, and species composition.

Example

We illustrate a simple application of the Chemkin package using a hydrogen oxidation process. The input file to the Chemkin Interpreter is shown in Fig. 1. It first specifies the elements and species that appear in the mechanism, and then describes the reaction mechanism itself. The input is essentially format free. The elements and species names need only be separated by blank spaces. The character string that describes the reaction appears on the left and is followed by the three Arrhenius coefficients (pre-exponential factor, temperature exponent, and activation energy). Enhanced third body efficiencies

```
ELEMENTS  H O N END
SPECIES   H2 H O2 O OH HO2 H2O2 H2O N N2 NO END

REACTIONS
   H2 + O2 = 2OH                    0.170E + 14   0.00   47780
   OH + H2 = H2O + H                0.117E + 10   1.30    3626   ! D-L&W
   O + OH = O2 + H                  0.400E + 15  -0.50       0   ! JAM 1986
   O + H2 = OH + H                  0.506E + 05   2.67    6290   ! KLEMM ET AL., 1986
   H + O2 + M = HO2 + M             0.361E + 18  -0.72       0   ! DIXON-LEWIS
         H2O/18.6/ H2/2.86/ N2/1.26/
   OH + HO2 = H2O + O2              0.750E + 13   0.00       0   ! D-L
   H + HO2 = 2OH                    0.140E + 15   0.00    1073   ! D-L
   O + HO2 = O2 + OH                0.140E + 14   0.00    1073   ! D-L
   2OH = O + H2O                    0.600E + 09   1.30       0   ! COHEN-WEST
   H + H + M = H2 + M               0.100E + 19  -1.00       0   ! D-L
         H2O/0.0/ H2/0.0/
   H + H + H2 = H2 + H2             0.920E + 17  -0.60       0
   H + H + H2O = H2 + H2O           0.600E + 20  -1.25       0
   H + OH + M = H2O + M             0.160E + 23  -2.00       0   ! D-L
         H2O/5/
   H + O + M = OH + M               0.620E + 17  -0.60       0   ! D-L
         H2O/5/
   O + O + M = O2 + M               0.189E + 14   0.00   -1788   ! NBS
   H + HO2 = H2 + O2                0.125E + 14   0.00       0   ! D-L
   HO2 + HO2 = H2O2 + O2            0.200E + 13   0.00       0
   H2O2 + M = OH + OH + M           0.130E + 18   0.00   45500
   H2O2 + H = HO2 + H2              0.160E + 13   0.00    3800
   H2O2 + OH = H2O + HO2            0.100E + 14   0.00    1800
   O + N2 = NO + N                  0.140E + 15   0.00   75800
   N + O2 = NO + O                  0.640E + 10   1.00    6280
   OH + N = NO + H                  0.400E + 14   0.00       0
END
```

Figure 1. Sample Reaction Mechanism as Read by the Chemkin Interpreter.

for selected species are specified in the line following that for a reaction which contains an arbitrary third body, M.

Assume the governing equation we wish to study is the energy conservation equation for a constant-pressure environment:

$$\frac{\partial T}{\partial t} = -\frac{1}{\rho c_p} \sum_{k=1}^{K} h_k \dot{\omega}_k,$$

where T is the temperature, ρ the mass density, c_p the mean specific heat, h_k the species enthalpies, and $\dot{\omega}_k$ the species molar production rates. The representation of this equation begins with Chemkin subroutine calls (the output variables are underlined to help distinguish them):

```
CALL CKINIT(LENIWK, LENRWK, LENCWK, LINKCK, LOUT, ICKWRK, RCKWRK, CCKWRK)
CALL CKINDX(ICKWRK, RCKWRK, MM, KK, II, NFIT)
CALL CKRHOY(P, T, Y, ICKWRK, RCKWRK, RHO)
CALL CKCPBS(T, Y, ICKWRK, RCKWRK, CPB)
CALL CKHML(T, ICKWRK, RCKWRK, HML)
CALL CKWYP(P, T, Y, ICKWRK, RCKWRK, WDOT)
```

The complete details for these calls are explained in later sections of this document, the object here being to illustrate the relative simplicity of a Chemkin application. Briefly, the first call is to the initialization subroutine CKINIT, which reads the Linking File created by the Interpreter and creates the three work arrays. LENIWK, LENRWK, and LENCWK are dimensions provided by the user for the data arrays ICKWRK, RCKWRK, and CCKWRK. LINKCK is the logical file number of the Linking File, and LOUT is the logical file number for printed diagnostic and error messages. In the remaining calls, P, T, and Y are the pressure, temperature, and vector of species mass fractions, respectively. The output variables correspond to the various terms for describing the equation, i.e., RHO = ρ, CPB = $\bar{c}_p$, HML = h_k, and WDOT = $\dot{\omega}_k$. The total number of species is denoted by KK.

The Fortran representation of the governing equation, given by combining the results of the above subroutine calls, is simply

```
      SUM=0.0
      DO 100 K=1,KK
          SUM = SUM + HML(K)*WDOT(K)
  100 CONTINUE
      DTDT = -SUM/(RHO*CPB)
```

One can see from this example that relatively little programming effort is required to form an arbitrary governing equation from an arbitrary reaction mechanism.

Transportability

The Chemkin package was developed on VAX/VMS and Cray/CTSS computers. However, we have not taken advantage of any special machine-dependent features. Written entirely in ANSI standard Fortran-77, the code is easily transportable to other computer systems. Since double-precision code is often required on small-word-length (i.e., 32-bit word) computers, we provide both single- and double-precision versions of the source code.

Organization of this Report

Chapter II is a compendium of important equations in gas-phase chemical kinetics. Many of the equations are simply definitions; but, in any case, derivations are either sketchy or not given. Although most readers will find all of the equations quite familiar, we find it useful to have these equations stated concisely in one document. For most of the equations, the package contains a subroutine that, when given the variables on the right-hand side, returns the variable on the left. Below the equation number is stated (in brackets) the name of the subroutine that provides information about that equation. For example, Eq. (3) in Chapter II gives mean molecular weight in terms of the mass fractions. Subroutine CKMMWY would therefore be called to return this information.

Chapter III explains the mechanics of using Chemkin and describes the job control logic for running a problem. Chapter IV explains the Chemkin Interpreter and how to set up the required symbolic input to define a reaction mechanism. Chapters V and VI describe the Gas-Phase Subroutine Library, Chapter V being composed of short descriptions for quick reference and Chapter VI (an alphabetical listing) explaining the input and output in the call sequence as well as cross referencing each subroutine to equation numbers in Chapter II. To demonstrate Chemkin explicitly, Chapter VII goes through a sample problem in detail.

Appendix A defines the allocation of three work arrays that are created from the Linking File. With this information, a user can create new subroutines for the library to suit a specialized need that was not anticipated in the current library.

II. THERMODYNAMICS AND CHEMICAL RATE EXPRESSIONS

The purpose of this chapter is to list expressions and equations that are potentially useful in formulating a chemically reacting flow problem. For each expression/equation, the subroutine that evaluates it is named.

Choice of Variables

The formulation of any problem requires that a set of dependent variables be chosen. Unfortunately there is no clear choice that is generally superior for all problems. In the Chemkin package we have decided to allow the user to select either pressure or density, temperature, and either mass fraction, mole fraction, or molar concentration. In other words, to define the state of a gas, one variable must be selected from each column of the array below.

$$\begin{pmatrix} P & T & Y_k \\ \rho & & X_k \\ & & [X_k] \end{pmatrix}$$

In making these options available from among the many possible, we have attempted to provide combinations of variables that are natural ones for a wide class of problems. For example, pressure is a natural choice in situations where pressure is fixed, and density is a natural variable where volume is fixed. Moreover, density is a natural variable in many problems involving fluid mechanics because it is determined directly from the mass continuity equation. Temperature is always taken as a natural variable because the thermodynamic properties and the chemical rate constants both depend directly on temperature. Mass fraction and mole fraction are convenient variables for describing the composition of a gas. Molar concentration is usually less convenient, but it is often a natural variable because the rate of progress of chemical reactions depends directly on the molar concentration of the reactants and products.

Equation of State

The equation of state used is that of a perfect gas:

$$P = \frac{\rho RT}{\overline{W}} \qquad (1)$$

[CKPY, CKPX, CKPC]

$$\rho = \frac{P\overline{W}}{RT} \qquad (2)$$

[CKRHOY, CKRHOX, CKRHOC]

The mean molecular weight $\overline{W}$ may be defined variously as

$$\overline{W} = \frac{1}{\sum_{k=1}^{K} Y_k/W_k}, \qquad (3)$$ [CKMMWY]

$$\overline{W} = \sum_{k=1}^{K} X_k W_k, \qquad (4)$$ [CKMMWX]

or

$$\overline{W} = \frac{\sum_{k=1}^{K} [X_k] W_k}{\sum_{k=1}^{K} [X_k]}. \qquad (5)$$ [CKMMWC]

Mole-Mass Conversion

It is often convenient to represent a gas-mixture species composition variously as either mass fraction, mole fraction, or molar concentration. In this section we state the conversion formulas between these ways to describe the mixture composition.

Mass fraction to mole fraction–

$$X_k = \frac{Y_k}{W_k \sum_{j=1}^{K} Y_j/W_j} = \frac{Y_k \overline{W}}{W_k} \qquad (6)$$ [CKYTX]

Mass fraction to molar concentration–

$$[X_k] = \frac{P\,(Y_k/W_k)}{RT \sum_{j=1}^{K} Y_j/W_j} = \left(\frac{P\overline{W}}{RT}\right) \frac{Y_k}{W_k} \qquad (7)$$ [CKYTCP]

$$[X_k] = \rho \frac{Y_k}{W_k} \qquad (8)$$ [CKYTCR]

Mole fraction to mass fraction–

$$Y_k = \frac{X_k W_k}{\sum_{j=1}^{K} X_j W_j} = \frac{X_k W_k}{\overline{W}} \qquad (9)$$ [CKXTY]

Mole fraction to molar concentration–

$$[X_k] = X_k \frac{P}{RT} \qquad (10)$$ [CKXTCP]

$$[X_k] = X_k \frac{\rho}{\overline{W}} \qquad (11)$$ [CKXTCR]

Molar concentration to mass fraction–

$$Y_k = \frac{[X_k] W_k}{\sum_{j=1}^{K} [X_j] W_j} \qquad (12)$$ [CKCTY]

Molar concentration to mole fraction–

$$X_k = \frac{[X_k]}{\sum_{j=1}^{K} [X_j]} \qquad (13)$$ [CKCTX]

Standard-State Thermodynamic Properties

Chemkin presumes that the standard-state thermodynamic properties are given in terms of polynomial fits to the specific heats at constant pressure:

$$\frac{C_{pk}^o}{R} = \sum_{n=1}^{N} a_{nk} T^{(n-1)} \tag{14}$$

The superscript o refers to the standard-state 1 atmosphere. For perfect gases, however, the heat capacities are independent of pressure; the standard-state values are the actual values. Other thermodynamic properties are given in terms of integrals of the specific heats. First, the standard-state enthalpy is given by

$$H_k^o = \int_0^T C_{pk}^o dT \tag{15}$$

so that

$$\frac{H_k^o}{RT} = \sum_{n=1}^{N} \frac{a_{nk} T^{(n-1)}}{n} + \frac{a_{N+1,k}}{T} \tag{16}$$

where the constant of integration $a_{N+1,k}R$ is the standard heat of formation at 0 K. Normally, however, this constant is evaluated from knowledge of the standard heat of formation at 298 K since the polynomial representations are usually not valid down to 0 K.

The standard-state entropy is written as

$$S_k^o = \int_0^T \frac{C_{pk}^o}{T} dT \tag{17}$$

so that

$$\frac{S_k^o}{R} = a_{1k} \ln T + \sum_{n=2}^{N} \frac{a_{nk} T^{(n-1)}}{(n-1)} + a_{N+2,k} \tag{18}$$

where the constant of integration $a_{N+2,k}R$ is evaluated from knowledge of the standard-state entropy at 298 K.

The above equations are stated for an arbitrary-order polynomial, but the Chemkin package is designed to work with thermodynamic data in the form used in the NASA chemical equilibrium code.[7] In this case, seven coefficients are needed for each of two temperature ranges.* These fits take the following form:

$$\frac{C_{p_k}^o}{R} = a_{1k} + a_{2k} T + a_{3k} T^2 + a_{4k} T^3 + a_{5k} T^4 \tag{19}$$

[CKCPOR]

* The Chemkin Interpreter can be modified for additional temperature ranges, which would then require format changes to the thermodynamic data.

$$\frac{H_k^o}{RT} = a_{1k} + \frac{a_{2k}}{2}T + \frac{a_{3k}}{3}T^2 + \frac{a_{4k}}{4}T^3 + \frac{a_{5k}}{5}T^4 + \frac{a_{6k}}{T} \qquad (20)$$
[CKHORT]

$$\frac{S_k^o}{R} = a_{1k}\ln T + a_{2k}T + \frac{a_{3k}}{2}T^2 + \frac{a_{4k}}{3}T^3 + \frac{a_{5k}}{4}T^4 + a_{7k} \qquad (21)$$
[CKSOR]

Other thermodynamic properties are easily given in terms of C_p^o, H^o, and S^o. The specific heat at constant volume C_v^o is stated as

$$C_{v_k}^o = C_{p_k}^o - R; \qquad (22)$$
[CKCVML]

the internal energy U is given as

$$U_k^o = H_k^o - RT, \qquad (23)$$
[CKUML]

the standard-state Gibbs free energy G^o is written as

$$G_k^o = H_k^o - TS_k^o, \qquad (24)$$
[CKGML]

and the standard-state Helmholtz free energy A_o is defined to be

$$A_k^o = U_k^o - TS_k^o. \qquad (25)$$
[CKAML]

For a perfect gas, the standard-state specific heats, enthalpies, and internal energies are also the actual values. Therefore, we drop the superscript o on those quantities.

Often, specific thermodynamic properties are needed in mass units (per gram) rather than in molar units (per mole). The conversion is made by dividing the property in molar units by the molecular weight. The specific properties are thus given as

$$c_{p_k} = \frac{C_{p_k}}{W_k} \qquad (26)$$
[CKCPMS]

$$h_k = \frac{H_k}{W_k} \qquad (27)$$
[CKHMS]

$$s_k^o = \frac{S_k^o}{W_k} \qquad (28)$$
[CKSMS]

$$c_{v_k} = \frac{C_{v_k}}{W_k} \qquad (29)$$
[CKCVMS]

$$u_k = \frac{U_k}{W_k} \qquad (30)$$
[CKUMS]

$$g_k^o = \frac{G_k^o}{W_k} \qquad (31)$$
[CKGMS]

$$a_k^o = \frac{A_k^o}{W_k} \tag{32}$$
[CKAMS]

One also often needs mixture-averaged thermodynamic properties. As with the pure-species properties, the Chemkin thermodynamics subroutines return properties in either mass or molar units. The mixture-averaged specific heats are given by

$$\overline{C}_p = \sum_{k=1}^{K} C_{p_k} X_k \tag{33}$$
[CKCPBL]

$$\bar{c}_p = \sum_{k=1}^{K} c_{p_k} Y_k = \overline{C}_p / \overline{W} \tag{34}$$
[CKCPBS]

$$\overline{C}_v = \sum_{k=1}^{K} C_{v_k} X_k \tag{35}$$
[CKCVBL]

$$\bar{c}_v = \sum_{k=1}^{K} c_{v_k} Y_k = \overline{C}_v / \overline{W}, \tag{36}$$
[CKCVBS]

the enthalpies by

$$\overline{H} = \sum_{k=1}^{K} H_k X_k \tag{37}$$
[CKHBML]

$$\overline{h} = \sum_{k=1}^{K} h_k Y_k = \overline{H} / \overline{W}, \tag{38}$$
[CKHBMS]

and the internal energies by

$$\overline{U} = \sum_{k=1}^{K} U_k X_k \tag{39}$$
[CKUBML]

$$\overline{u} = \sum_{k=1}^{K} u_k Y_k = \overline{U} / \overline{W}. \tag{40}$$
[CKUBMS]

The mixture properties are more complex for the entropies and the Gibbs and Helmholtz free energies. Here the actual values are not the same as the standard-state values and we must account for the appropriate pressure and entropy-of-mixing terms, i.e.,

$$S_k = S_k^o - R \ln X_k - R \ln(P/P_{\rm atm}), \tag{41}$$

where $P_{\rm atm}$ is the standard-state pressure of 1 atmosphere. Thus the mixture-averaged entropy is given by

$$\overline{S} = \sum_{k=1}^{K} \left(S_k^o - R \ln X_k - R \ln(P/P_{\rm atm}) \right) X_k \tag{42}$$
[CKSBML]

$$\bar{s} = \overline{S} / \overline{W}. \tag{43}$$
[CKSBMS]

Similarly, the mixture-averaged Gibbs free energy is given as

$$\overline{G} = \sum_{k=1}^{K} \left[H_k - T\left(S_k^o - R\ln X_k - R\ln(P/P_{\rm atm})\right)\right] X_k \qquad (44)$$

[CKGBML]

$$\bar{g} = \overline{G}/\overline{W}, \qquad (45)$$

[CKGBMS]

and the mixture-averaged Helmholtz free energy is given by

$$\overline{A} = \sum_{k=1}^{K} \left[U_k - T\left(S_k^o - R\ln X_k - R\ln(P/P_{\rm atm})\right)\right] X_k \qquad (46)$$

[CKABML]

$$\bar{a} = \overline{A}/\overline{W}. \qquad (47)$$

[CKABMS]

Chemical Reaction Rate Expressions

Consider I elementary reversible (or irreversible) reactions involving K chemical species that can be represented in the general form

$$\sum_{k=1}^{K} \nu'_{ki}\chi_k \;\rightleftarrows\; \sum_{k=1}^{K} \nu''_{ki}\chi_k \qquad (i = 1, ..., I) \qquad (48)$$

The stoichiometric coefficients ν_{ki} are integers* and χ_k is the chemical symbol for the kth species. Normally, an elementary reaction involves only three or four species; hence the ν_{ki} matrix is quite sparse for a large set of reactions.

The production rate $\dot{\omega}_k$ of the kth species can be written as a summation of the rate-of-progress variables for all reactions involving the kth species:

$$\dot{\omega}_k = \sum_{i=1}^{I} \nu_{ki} q_i \qquad (k = 1, ..., K) \qquad (49)$$

[CKWYP, CKWYR, CKWXP, CKWXR, CKWC, CKCONT]

where

$$\nu_{ki} = (\nu''_{ki} - \nu'_{ki}). \qquad (50)$$

[CKNU]

* Global reactions are sometimes stated with non-integer stoichiometric coefficients. However, because we have designed Chemkin to work exclusively with elementary reaction steps, we only consider integer stoichiometric coefficients.

The rate-of-progress variable q_i for the ith reaction is given by the difference of the forward rates and the reverse rates as

$$q_i = k_{f_i} \prod_{k=1}^{K} [X_k]^{\nu'_{ki}} - k_{r_i} \prod_{k=1}^{K} [X_k]^{\nu''_{ki}} \tag{51}$$

[CKQYP,CKQYR, CKQXP, CKQXR, CKQC, CKCONT]

where $[X_k]$ is the molar concentration of the kth species and k_{f_i} and k_{r_i} are the forward and reverse rate constants of the ith reaction. The forward rate constants for the I reactions are generally assumed to have the following Arrhenius temperature dependence:

$$k_{f_i} = A_i T^{\beta_i} \exp\left(\frac{-E_i}{R_c T}\right) \tag{52}$$

[CKABE]

where the pre-exponential factor A_i, the temperature exponent β_i, and the activation energy E_i are specified.* These three parameters are required input to the Chemkin package for each reaction.

The reverse rate constants k_{r_i} are related to the forward rate constants through the equilibrium constants as

$$k_{r_i} = \frac{k_{f_i}}{K_{c_i}} \tag{53}$$

Although K_{c_i} is given in concentration units, the equilibrium constants are more easily determined from the thermodynamic properties in pressure units; they are related by

$$K_{c_i} = K_{p_i} \left(\frac{P_{\text{atm}}}{RT}\right)^{\sum_{k=1}^{K} \nu_{ki}} \tag{54}$$

[CKEQYP, CKEQYR, CKEQXP, CKEQXR, CKEQC]

where P_{atm} denotes a pressure of 1 atm. The equilibrium constants K_{p_i} are obtained with the relationship

$$K_{p_i} = \exp\left(\frac{\Delta S_i^o}{R} - \frac{\Delta H_i^o}{RT}\right) \tag{55}$$

The Δ refers to the change that occurs in passing completely from reactants to products in the ith reaction. More specifically,

$$\frac{\Delta S_i^o}{R} = \sum_{k=1}^{K} \nu_{ki} \frac{S_k^o}{R} \tag{56}$$

$$\frac{\Delta H_i^o}{RT} = \sum_{k=1}^{K} \nu_{ki} \frac{H_k^o}{RT} \tag{57}$$

---

* Two gas constants, R and R_c, are used throughout this report and the Chemkin code. R_c is used only in conjunction with the activation energy E_i and has compatible units. The reason for the duality is because we find that many users would rather use different units (say calories/mole) for the activation energies even though other units (say cgs or SI) are used otherwise.

Three-Body Reactions

In some reactions a "third body" is required for the reaction to proceed; this is often the case in dissociation or recombination reactions, e.g.,

$$H+O_2+M \rightleftarrows HO_2 +M.$$

When a third body is needed, the concentration of the effective third body must appear in the expression for the rate-of-progress variable. Accordingly, the rate-of-progress variable is different from Eq. (51) by the first factor in the equation below:

$$q_i = \left(\sum_{k=1}^{K}(\alpha_{ki})[X_k]\right)\left(k_{f_i}\prod_{k=1}^{K}[X_k]^{\nu'_{ki}} - k_{r_i}\prod_{k=1}^{K}[X_k]^{\nu''_{ki}}\right) \quad (58)$$

[CKQYP, CKQYR, CKQXP, CKQXR, CKQC, CKTHB]

If all species in the mixture contribute equally as third bodies, then all the $\alpha_{ki} = 1$, and the first factor is the total concentration of the mixture,

$$[M] = \sum_{k=1}^{K}[X_k] = \frac{P}{RT} \quad (59)$$

However, it is often the case that some species act more efficiently as third bodies than do others. The α_{ki} coefficients are then used to specify the increased efficiency of the kth species in the ith reaction. Also, if a species is to be excluded from acting as a third body in a particular reaction, then $\alpha_{ki} = 0$ for that species. Any α_{ki} that differ from 1 must be specified by input to the Chemkin Interpreter.

Pressure-Dependent Fall-off Reactions

Under certain conditions, some reactions can fall in a regime that is between the high- and low-pressure limiting forms of the rate expressions. As an example consider methyl (CH_3) recombination. In the high-pressure limit, the appropriate description of the reaction is $CH_3 + CH_3 \rightleftarrows C_2H_6$. In the low-pressure limit, the appropriate description is $CH_3 + CH_3 + M \rightleftarrows C_2H_6 + M$. When such a reaction is at either limit, the rate expressions discussed in the preceeding paragraphs are applicable. However, when the pressure and temperature are such that the reaction is between the limits, the rate expressions are more complicated. To denote a reaction that is in this "fall-off" region, we write the reaction with the M enclosed in parentheses,

$$CH_3 + CH_3 (+ M) \rightleftarrows C_2H_6 (+ M).$$

There are several methods of representing the rate expressions in this fall-off region. The simplest one is due to Lindemann.[8] There are also now two other (and related) methods that provide a more accurate description of the fall-off region than does the simple Lindemann form. The Chemkin package handles all three of these forms as options.

We begin first with the Lindemann approach. Arrhenius rate parameters are required for both the high- and low-pressure limiting cases, and the Lindemann form for the rate coefficient blends them to produce a pressure-dependent rate expression. In Arrhenius form, the parameters are given for the high-pressure limit (k_∞) and the low-pressure limit (k_0) as follows:

$$k_0 = A_0 T^{\beta_0} \exp(-E_0/R_c T), \tag{60}$$

$$k_\infty = A_\infty T^{\beta_\infty} \exp(-E_\infty/R_c T). \tag{61}$$

The rate constant at any pressure is then taken to be

$$k = k_\infty \left(\frac{P_r}{1+P_r}\right) F, \tag{62}$$

where the reduced pressure P_r is given by

$$P_r = \frac{k_0[M]}{k_\infty} \tag{63}$$

and $[M]$ is the concentration of the mixture (possibly including enhanced third-body efficiencies).† If the F in Eq. (62) is unity, then this is the Lindemann form. The other descriptions involve more complex forms for the function F.

In the Troe form[9] F is given by

$$\log F = \left[1 + \left[\frac{\log P_r + c}{n - d(\log P_r + c)}\right]^2\right]^{-1} \log F_{\text{cent}} \ . \tag{64}$$

The constants in Eq. (64) are

$$c = -0.4 - 0.67 \log F_{\text{cent}} \tag{65}$$

$$n = 0.75 - 1.27 \log F_{\text{cent}} \tag{66}$$

$$d = 0.14 \tag{67}$$

and

$$F_{\text{cent}} = (1-a)\exp(-T/T^{***}) + a\exp(-T/T^{*}) + \exp(-T^{**}/T). \tag{68}$$

The four parameters a, T^{***}, T^{*}, and T^{**} must be specified as input to the Chemkin Interpreter. (It is often the case that the parameter T^{**} is not used. Thus Chemkin provides for the use of either three or four parameters.)

† It is also possible that the third body in the fall-off region could be a specific species rather than the mixture as a whole. In such a case, the reaction could be written, for example, as $CH_3 + CH_3 \ (+ \ N_2) \rightleftarrows C_2H_6 \ (+ \ N_2)$. In this case, the concentration of nitrogen $[N_2]$ would replace the total concentration of the mixture [M] in these equations.

The approach taken at SRI International by Stewart et al.[10] is in many ways similar to that taken by Troe, but the blending function F is approximated differently. Here, F is given by

$$F = \left[a \exp\left(\frac{-b}{T}\right) + \exp\left(\frac{-T}{c}\right)\right]^{X} dT^{e} \tag{69}$$

where

$$X = \frac{1}{1 + \log^2 P_r}. \tag{70}$$

In addition to the six Arrhenius parameters—three each for the low-pressure limit (k_0) and high-pressure limit (k_∞) expressions—the user must supply the parameters a, b, and c in the F expression. Note that a and c here are not the same as the a and c in the Troe formulation. The parameters d and e were not discussed by Stewart et al., but we have included them as additional optional parameters to increase flexibility. If one wishes, d and e can be considered parameters that define the weak-collision efficiency factor (β_c) dependence of F, in the event that one wants to compute strong-collision rate parameters and correct them with various values of β_c.

Landau-Teller Formulation of the Rate Expressions

For reactions such as vibrational energy transfer processes, the Arrhenius form of the rate expression (Eq. 52) is often not used. Instead, it is common to use the following Landau-Teller expression,

$$k_{f_i} = A_i \exp\left(\frac{B_i}{T^{\frac{1}{3}}} + \frac{C_i}{T^{\frac{2}{3}}}\right). \tag{71}$$

In Chemkin, we have provided the possibility to blend the Arrhenius expression with the Landau-Teller expression in the general expression below

$$k_{f_i} = A_i T^{\beta_i} \exp\left(\frac{-E_i}{R_c T} + \frac{B_i}{T^{\frac{1}{3}}} + \frac{C_i}{T^{\frac{2}{3}}}\right). \tag{72}$$

Clearly, by setting B_i and C_i to zero, the Arrhenius expression is recovered, and by setting β_i and E_i to zero, the standard Landau-Teller expression is recovered. If appropriate, however, all the parameters can be used together to provide more flexibility in the reaction-rate expression than could be afforded by one of the forms alone.

Special Forms of the Rate Expressions

It is often convenient to separate the species chemical production rates into creation and destruction rates. Furthermore, some numerical approaches take advantage of this separation. Therefore, we provide subroutines that return the chemical rates in the following form:

$$\dot{\omega}_k = \dot{C}_k - \dot{D}_k, \tag{73}$$

[CKCDYP, CKCDYR, CKCDXP, CKCDXR, CKCDC]

where, for non-three-body reactions,

$$\dot{C}_k = \sum_{i=1}^{I} \nu'_{ki} k_{r_i} \prod_{j=1}^{K} [X_j]^{\nu''_{ji}} + \sum_{i=1}^{I} \nu''_{ki} k_{f_i} \prod_{j=1}^{K} [X_j]^{\nu'_{ji}} \tag{74}$$

and

$$\dot{D}_k = \sum_{i=1}^{I} \nu'_{ki} k_{f_i} \prod_{j=1}^{K} [X_j]^{\nu'_{ji}} + \sum_{i=1}^{I} \nu''_{ki} k_{r_i} \prod_{j=1}^{K} [X_j]^{\nu''_{ji}}. \tag{75}$$

When third body reactions are involved, each sum in the above equations is multiplied by the third-body concentration

$$[M] = \sum_{k=1}^{K} \alpha_{ki} [X_k].$$

Another useful form for the chemical production rates is found by defining a creation rate and a characteristic time for the destruction rate, i.e.,

$$\dot{\omega}_k = \dot{C}_k - \frac{[X_k]}{\tau_k}. \tag{76}$$

[CKCTYP, CKCTYR, CKCTXP, CKCTXR, CKCTC]

Here the characteristic time is given simply in terms of $\dot{D}_k$ as

$$\tau_k = \frac{[X_k]}{\dot{D}_k}. \tag{77}$$

As a precaution against $[X_k]$ and $\dot{D}_k$ simultaneously approaching zero, the Chemkin implementation of the destruction time is written as

$$\tau_k = \frac{[X_k]}{\dot{D}_k + \epsilon}, \tag{78}$$

[CKCTYP, CKCTYR, CKCTXP, CKCTXR, CKCTC]

where ϵ is an arbitrary small number,* say 10^{-50}.

* This computer-dependent number is set in the Gas-Phase Subroutine Library at the time the library is created.

III. THE MECHANICS OF USING CHEMKIN

Chemkin is a highly structured and modular package that requires the manipulation of a number of programs, subroutines, and data files. This chapter describes the structure of the package and the job-control logic that is required to use it.

Structure of Chemkin

The general structure of the Chemkin package is shown in Fig. 2. The Interpreter is a program that reads a symbolic description of a reaction mechanism and then extracts the needed thermodynamic data for each species involved from the Thermodynamic Database. The primary output from the Interpreter is a binary file called the Linking File. This file contains information that contains all required information about the elements, species, and reactions in the user's mechanism.

The Linking File is written on LINKCK (defaulted as Fortran unit 25). The logical file number for LINKCK must be declared both in the Interpreter (so it can be written) and in the user's code (so that it can be read by the initialization subroutine).

In addition to the Linking File, three other files are needed by the Interpreter: an input file, an output file, and a Thermodynamic Database file.[6] The input to the Interpreter is read from file LIN (defaulted as Fortran unit 15) and printed output is directed to LOUT (defaulted as Fortran unit 16). The printed output contains a listing of the elements, species, and the reaction mechanism, and it provides diagnostic error messages if they should be needed.

The Thermodynamic Database is assigned to file LTHRM (defaulted as Fortran unit 17). LTHRM can be a large file with information on many species, most of which are not needed for any given problem. Thermodynamic data can also be read from input; these data can replace or add to that in the Thermodynamic Database.

Once the Interpreter has been executed and the Linking File created, the user is ready to use the Gas-Phase Subroutine Library. Subroutines from this library are called from the user's Fortran code. The user's first step must be to dimension three work arrays (one integer, one floating point, and one character data type)* and then call the initialization subroutine CKINIT to create the work arrays from the Linking File.† One or more of these arrays is required input to every other subroutine in the Chemkin package.

* The minimum length for the arrays can be found in Interpreter output.

† If there is an error in the input to the Interpreter, CKINIT will print a diagnostic message and execution will stop.

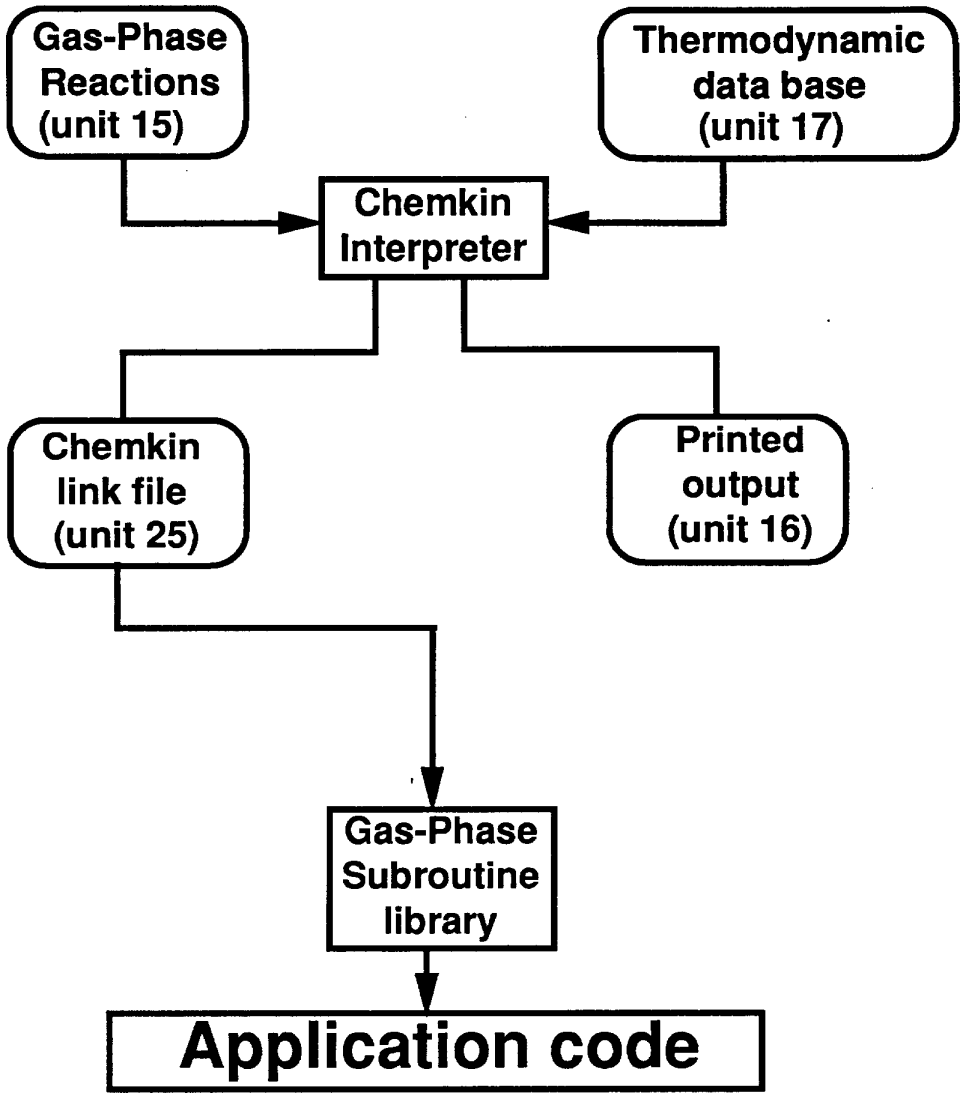

Figure 2. Schematic diagram showing the structure of the Chemkin package and its relationship to an application code.

Selection of Chemkin subroutines for any given problem begins by finding the appropriate equations in Chapter II. Most equations give a reference to a subroutine name, for which the input and output lists are described in Chapters V and VI. Normally only a few of the subroutines in the package would be called for any one problem. Therefore, the subroutine package should be implemented in an object library format* so that only those routines that are actually called by the user's code are loaded at the time of execution.

* An object library is a collection of compiled subroutines that are stored in a special way so that the computer only links those subroutines that are referenced in the user's program. All computer operating systems have such a facility. In VAX/VMS, libraries are made with the LIBRARY/CREATE command.

Job Control

By example we show here how to run a simple application program using Chemkin. Figure 3 is an annotated VAX/VMS command procedure that outlines the important steps. Even though the example is specific to VAX/VMS systems, the same functionality must be invoked on any computer system. For the example, we assume that the Interpreter has already been compiled and is in the form of an executable image. Furthermore, we assume that the Gas-Phase Subroutine Library has been compiled and an object library has been created.

VAX/VMS Commands			Meaning
$assign	MECHANISM.DAT	FOR015	Assign the user's reaction mechanism to Fortran unit 15. This is the input file for the Chemkin Interpreter.
$assign	INTERP.OUT	FOR016	Assign the output file for printed output from the Chemkin Interpreter. The Interpreter writes to unit 16.
$assign	THERMO.DAT	FOR017	Assign the Thermodynamic Database to Fortran unit 17.
$assign	LINK.BIN	FOR025	Assign the Linking file to Fortran unit 25.
$run	INTERP.EXE		Execute the Interpreter.
$for	SAMPLE.FOR		Compile the user's Fortran program.
$assign	SAMPLE.INP	FOR005	Assign a file containing any input required by the user's program to Fortran unit 5.
$assign	SAMPLE.OUT	FOR006	Assign a file to accept any printed output from the user's program to Fortran unit 6.
$link	SAMPLE.OBJ, CKLIB/LIB		Link the user's program with the Chemkin Gas-Phase Subroutine Library.
$run	SAMPLE		Execute the user's program.

Figure 3. A sample VAX/VMS command procedure showing the steps required to run an application code using the Chemkin package.

IV. USING THE INTERPRETER

The Interpreter is used to read (from file LIN) a symbolic description of an elementary chemical reaction mechanism and create a Linking File (LINKCK) of pertinent information about that mechanism. The information in the Linking File is subsequently accessed by various subroutines to provide information on equation of state, thermodynamic properties, and chemical production rates.

The Interpreter input includes information on elements, species, thermodynamic data, and the reaction mechanism. Input information on file LIN is given in 80-column card image format. Element data are read first; species data are second, followed by optional thermodynamic data, with reactions specified last. The thermodynamic data for the species may come from input (file LIN) and/or from a Thermodynamic Database (file LTHRM). The syntax for the four types of input is described below.

With the exception of the thermodynamic data, all input is format free. For the thermodynamic data, we have chosen to use the same format as used in the NASA Chemical Equilibrium code by Gordon and McBride.[7]

Element Data

All chemical species in the reaction mechanism must be composed of chemical elements or isotopes of chemical elements. Each element and isotope must be declared as a one- or two-character symbol. The purpose of the element data is to associate atomic weights of the elements with their character symbol representations and to identify the order in which arrays of element information in the Gas-Phase Subroutine Library are referenced. For example, a Fortran array of atomic weights for the elements is in exactly the same order in which the elements were declared in the element data. In other words, if the atomic weights are stored in an array AWT, then AWT(3) is the atomic weight of the third element declared in the element data.

For the elements appearing on the periodic chart, the Interpreter has the atomic weight (in grams per mole) stored internally. For isotopes, a one- or two- character symbol must be input to the Interpreter to identify each isotope, and a symbol and an atomic weight (in grams per mole) for each must be defined. The same symbol must be used in the thermodynamic data to identify the elemental composition of species involving the isotope. Once an isotope has been so defined, it is treated exactly as a new element. If an ionic species is used in the mechanism (i.e., OH+), an electron must be declared as the element E.

Element data must start with the word ELEMENTS (or ELEM), followed by any number of element symbols on any number of lines. Element symbols may appear anywhere on a line, but those on the same line must be separated by blanks. Any line or portion of a line starting with an exclamation mark (!) is considered a comment and will be ignored. Blank lines are ignored.

If an element is on the periodic chart,* then only the symbol identifying the element need appear in the element data. For an isotope, the atomic weight must follow the identifying symbol and be delimited by slashes (/). The atomic weight may be in integer, floating point, or E format (e.g., 2, 2.0, 0.2E1), but internally it will be converted to a floating point number. For example, the isotope deuterium may be defined as D/2.014/. If desired, the atomic weight of an element in the periodic chart may be altered by including the atomic weight as input just as though the element were an isotope.

Figure 4 shows several equivalent ways to describe element information. In this example the elements are hydrogen, oxygen, nitrogen, and the isotope deuterium. Table I summarizes the rules for element data.

* The elements that Chemkin recognizes are as follows:

H	HE	LI	BE	B	C	N	O	F	NE
NA	MG	AL	SI	P	S	CL	AR	K	CA
SC	TI	V	CR	MN	FE	CO	NI	CU	ZN
GA	GE	AS	SE	BR	KR	RB	SR	Y	ZR
NB	MO	TC	RU	RH	PD	AG	CD	IN	SN
SB	TE	I	XE	CS	BA	LA	CE	PR	ND
PM	SM	EU	GD	TB	DY	HO	ER	TM	YB
LU	HF	TA	W	RE	OS	IR	PT	AU	HG
TL	PB	BI	PO	AT	RN	FR	RA	AC	TH
PA	U	NP	PU	AM	CM	BK	CF	ES	FM
D	E								

```
ELEMENTS      H   D /2.014/   O   N   END

ELEM                                        ! ELEM is equivalent to ELEMENTS
  H
  D / 2.014 /
  O
  N
END                                         ! an END line is optional

ELEM H
ELEM D/2.014/
ELEM O
ELEM N
```

Figure 4. Equivalent Ways to Describe Element Information.

TABLE I. SUMMARY OF THE RULES FOR ELEMENT DATA

1. The first element line must start with the word ELEMENTS (or ELEM).
2. Element or isotope names are either one- or two-character symbols.
3. An isotope name (i.e., a name not on the periodic chart) must be followed by its atomic weight (in grams per mole) delimited by slashes.
4. Each element or isotope should be declared only once; however, duplicated element symbols will be ignored.
5. An element or isotope name may appear anywhere on the line.
6. Any number of element or isotope names may appear on a line, and more than one line may be used.
7. Element or isotope names that appear on the same line must be separated by at least one blank space.
8. An element or isotope name that begins on one line may not continue to the next line.
9. Any blank spaces between an element or isotope name and the first slash are ignored and any blank spaces between slashes and an atomic weight are also ignored. However, no blank spaces are allowed within an element name or an atomic weight.
10. There may be more than one ELEMENT statement.
11. All characters following an exclamation mark are comments.

Species Data

Each chemical species in a problem must be identified on a species line(s). Any set of up to 16 upper or lower case characters* can be used as a species name. In addition, each species must be composed of elements that have been identified in the element data. As for the element data, one of the primary purposes of the species data is to identify the order in which Fortran arrays of species information are referenced in the Gas-Phase Subroutine Library.

Species data must start with the word SPECIES (or SPEC), followed by any number of species symbols on any number of lines. Species symbols may appear anywhere on a line, but those on the same line must be separated by blank spaces. Any line or portion of a line starting with an exclamation mark (!) is considered a comment and will be ignored. Blank lines are ignored. Figure 5 shows several equivalent ways to describe species information. The rules for species data are summarized in Table II.

```
SPECIES     H2  O2  H  O  OH  HO2  N2  N  NO  END

SPEC                                          ! SPEC is equivalent to SPECIES
  H2  O2
  H  O  OH  HO2  N2  N  NO
END                                           ! an END statement is optional

SPEC H2
spec O2
  etc.
```

Figure 5. Equivalent Ways to Describe Species Information.

* Species names may not begin with a number, a +, or an =, or have imbedded blanks; an ionic species may end with any number of +'s or -'s; an imbedded plus sign (+) must be enclosed in parentheses.

TABLE II. SUMMARY OF THE RULES FOR SPECIES DATA

1. Species data must start with the word SPECIES (or SPEC).
2. Species names are composed of up to 16-character upper- or lower- case symbols. The names cannot begin with the characters +, =, or a number; an ionic species name may end with one or more +'s or −'s.
3. Each species should be declared only once; however, duplicated species symbols will be ignored.
4. Each species that subsequently appears in a reaction must be declared.
5. A species name may appear anywhere on the line.
6. Any number of species names may appear on a line, and more than one line may be used.
7. Species named on the same line must be separated by at least one blank space.
8. A species name that begins on one line may not continue to the next line.
9. There may be more than one SPECIES statement.
10. All characters following an exclamation mark are comments.

Thermodynamic Data

Any chemical species that appears in a problem must have thermodynamic data associated with it. The data may be extracted from a database (file LTHRM) and/or read from input (file LIN). If all the thermodynamic data are to be extracted from the database, then no thermodynamic data input is required. However, if the user wishes to override information in the database or to provide data on species not in the database, then Interpreter input is needed. In any case the format for the information is the same.

The format (see Table III) is a minor modification of that used by Gordon and McBride[3] for the Thermodynamic Database in the NASA Chemical Equilibrium code. Our modification allows for a different midpoint temperature for the fits to the properties of each chemical species. We also allow a species to be composed of a maximum of five elements, not four. However, the formatting is such that the Chemkin Interpreter can use the NASA database directly without any modification.

As indicated in Table III, the pertinent information includes the species name, the elemental composition of the species, and the temperature ranges over which the polynomial fits to thermodynamic data are valid. The fits to C_p^o/R, H^o/RT, and S^o/R

TABLE III. SUMMARY OF THE RULES FOR THERMO DATA

Line Number	Contents	Format	Column
1	THERMO (or THERMO ALL[a])	Free	Any
2[b]	Temperature ranges for 2 sets of coefficients: lowest T, common T, and highest T	3F10.0	1 to 30
3	Species name (must start in Column 1)	18A1	1 to18
	Date (not used in the code)	6A1	19 to 24
	Atomic symbols and formula	4(2A1,I3)	25 to 44
	Phase of species (S, L, or G for solid, liquid, or gas, respectively)	A1	45
	Low temperature	E10.0	46 to 55
	High temperature	E10.0	56 to 65
	Common temperature (if needed) (blank for default)	E8.0	66 to 73
	Atomic symbols and formula (if needed) (blank for default)	2A1,I3	74 to 78
	The integer 1	I1	80
4	Coefficients $a_1 - a_5$ in Eqs. (19) - (21), for upper temperature interval	5(E15.0)	1 to 75
	The integer 2	I1	80
5	Coefficients a_6, a_7 for upper temperature interval, and a_1, a_2, and a_3 for lower	5(E15.0)	1 to 75
	The integer 3	I1	80
6	Coefficients a_4, a_5, a_6, a_7 for lower temperature interval	4(E15.0)	1 to 60
	The integer 4	I1	80
...	Repeat lines 3 - 6 for each species.		
last	END (Optional, end of thermodynamic data.)	Free	Any

[a]Use only when all the thermodynamic data are to be taken from Interpreter input.
[b]Include line 2 *only* with THERMO ALL (it is already in the Thermodynamic Database).

consist of seven coefficients for each of two temperature ranges [see Eqs. (19) - (21)].* Further information about the fitting procedure and data for many species can be found in a report on the Chemkin Thermodynamic Database.[5]

When thermodynamic data input is required, it must immediately follow species data.† The first thermodynamic data line must start with the word THERMO (or THER). If all the thermodynamic data are input directly to the Interpreter, then the first line must read THERMO ALL and the code will not expect a Thermodynamic Database from file LTHRM; for this option the next line must be line 2 of Table III. For either option, the subsequent thermodynamic data lines must be in the format of lines 3 – 6 of Table III. (For the THERMO option the midpoint temperature is taken from the line 2 information already in the Thermodynamic Database.) As many species as needed can be included as THERMO input.

Figure 6 shows some examples of thermodynamic property input. In these three examples for OH, OH+, and OH–, it is seen from columns 25 – 34 that the elemental composition of each molecule is one O atom and one H atom. In addition, columns 35 – 39 indicate that two of the species, OH+ and OH–, are ionic since they contain –1 and +1 electrons (E), respectively. The G in column 45 indicates that all three species are gaseous. The 1000.00 in columns 66 – 73 for OH+ indicates that the common temperature between the high- and low-temperature fits is 1000.00 K. If columns 66 – 73 are left blank, as they are for OH+ and OH–, then the common temperature is that given in columns 21 – 30 of line 2 in Table III, which in this example is in the Thermodynamic Database.

```
THERMO
OH                      O   1H   1          G  0300.00   5000.00  1000.00
 0.02882730E+02 0.10139743E-02-0.02276877E-05 0.02174683E-09-0.05126305E-14
 0.03886888E+05 0.05595712E+02 0.03637266E+02 0.01850910E-02-0.16761646E-05
 0.02387202E-07-0.08431442E-11 0.03606781E+05 0.13588605E+01
OH+                     O   1H   1E  -1     G  0300.00   5000.00
 0.02719058E+02 0.15085714E-02-0.05029369E-05 0.08261951E-09-0.04947452E-13
 0.15763414E+06 0.06234536E+02 0.03326978E+02 0.13457859E-02-0.03777167E-04
 0.04687749E-07-0.01780982E-10 0.15740294E+06 0.02744042E+02
OH-                121286O   1H   1E   1     G  0300.00   5000.00
 0.02846204E+02 0.10418347E-02-0.02416850E-05 0.02483215E-09-0.07775605E-14
-0.01807280E+06 0.04422712E+02 0.03390037E+02 0.07922381E-02-0.01943429E-04
 0.02001769E-07-0.05702087E-11-0.01830493E+06 0.12498923E+01
END
```

Figure 6. Examples of Thermodynamic Data Input.

* Additional temperature ranges and their fit coefficients may be accommodated by minor changes to the Interpreter and the Thermodynamic Database.

† In the original Chemkin, the thermodynamic data preceeded the species data.

The following is a summary of the possibilities for specifying thermodynamic data.

Case 1: All thermodynamic data from database

1. Assign the database as file LTHRM (default Fortran unit 17)
2. No THERMO data required as input

Case 2: Thermodynamic data from database and input

1. Assign the database as file LTHRM (default Fortran unit 17)
2. Include the following lines:
   THERMO
   Data in Table III format (lines 3 – 6 repeated) for species not in the database or to override species in database
   END

Case 3: All thermodynamic data from input

1. Do not attach a database
2. Include the following lines:
   THERMO ALL
   Line 2 of Table III format.
   Data in Table III format (lines 3 – 6 repeated) for at least all species named in the species data.
   END

Reaction Mechanism Description

The reaction mechanism may consist of any number of chemical reactions involving the species named in the species data. A reaction may be reversible or irreversible; it may be a three-body reaction with an arbitrary third body and/or enhanced third body efficiencies; it may have a Lindemann,[8] Troe,[9] or SRI* fall-off formulation†; and it may involve a photon.

Reaction data must start with the word REACTIONS (or REAC). On the same line, the user may specify units of the Arrhenius rate coefficients [Eq. (52)] to follow by including the word CAL/MOLE, KCAL/MOLE, JOULES/MOLE, or KELVINS for E_i, and/or MOLES or MOLECULES for A_i. If MOLECULES is specified, then the units for A_i are cm-molecules-sec-K. If units are not specified, A_i and E_i must be in cm-mole-sec-K and cal/mole, respectively. The lines following the REACTION line contain reaction descriptions together with their Arrhenius rate coefficients. The reaction description is composed of reaction data and perhaps auxiliary information data.

---

* SRI refers to the formulation of Stewart et al.[10], who are at SRI International, Menlo Park, CA.

† See Section III for a discussion of the different formulations

Reaction Data

Each reaction line is divided into two fields. The first contains the symbolic description of the reaction while the second contains the Arrhenius rate coefficients. Both fields are format free and blank spaces are ignored. Any line or portion of a line starting with an exclamation mark (!) is considered a comment and will be ignored. Blank lines are ignored.

The reaction description, given in the first field, must be composed of the species symbols, coefficients, delimiters, and special symbols as summarized below.

Species Symbols: Each species in a reaction is described with the unique sequence of characters as they appear in the species data and the thermodynamic data.

Coefficients: A species symbol may be preceded by an integer coefficient. The coefficient has the meaning that there are that many moles of the particular species present as either reactants or products; e.g., 2OH is equivalent to OH + OH (a non-integer coefficient is not allowed).

Delimiters:

\+ A plus sign is the delimiter between each reactant species and each product species

= An equality sign is the delimiter between the last reactant and the first product in a reversible reaction

<=> An equality sign enclosed by angle brackets can also used as the delimiter between the last reactant and the first product in a reversible reaction

=> An equality sign with an angle bracket on the right is the delimiter between the last reactant and first product in an irreversible reaction

Special Symbols:

+M An M as a reactant and/or product stands for an arbitrary third body. Normally it would appear as both a reactant and a product. However, it has the identical meaning even if it appears only as a reactant or a product. In other words, an M anywhere in the reaction description indicates that a third body is participating in the reaction. In a reaction containing an M, species can be specified to have enhanced third body efficiencies, in which case auxiliary information data (described below) must follow the reaction line. If no enhanced third

body efficiencies are specified, then all species act equally as third bodies and the effective concentration of the third body is the total concentration of the mixture.

(+M) An M as a reactant and/or product surrounded by parentheses indicates that the reaction is a pressure-dependent reaction, in which case auxiliary information line(s) (described below) must follow the reaction to identify the fall-off formulation and parameters. A species may also be enclosed in parenthesis. Here, for example, ($+H_2O$) indicates that water is acting as the third body in the fall-off region, not the total concentration M.

HV The symbol HV as a reactant and/or product indicates that photon radiation ($h\nu$) is present. If HV appears in a reaction description, the wavelength of the radiation may be specified on an auxiliary information line (described below).

E The symbol E as a reactant and/or product is used to represent an electron. An electron is treated just like any other species, and is composed of the element E, which must be declared as element data. If an E appears in any reaction, then it must also be declared as a species in the species data and thermodynamic data must be supplied for it.

! An exclamation mark means that any and all following characters are comments on the reaction. For example the comment may be used to give a reference to the source of the reaction and rate data.

The second field of the reaction line is used to define the Arrhenius rate coefficients A_i, β_i, and E_i, in that order, as given by Eq. (52). At least one blank space must separate the first number and the last symbol in the reaction. The three numbers must be separated by at least one blank space, be stated in either integer, floating point, or E format (e.g., 123 or 123.0 or 12.3E1), and have units associated with them. Unless modified by the REACTION line, the default units for A_i are cgs (cm, sec, K, mole), the exact units depending on the reaction. The factor β_i is dimensionless. The default units for the activation energies are cal/mole.

Examples of some reaction data are shown in Figure 7, and Table IV is a summary of the reaction data rules.

```
REACTIONS                                CAL/MOLE

H2+O2 = 2OH                                    1.7E13  0  47780.  ! Ref. 21
! H2 + O2 = OH + OH                      1.7E13  0  47780.  ! same as previous reaction,
                                                  ! commented to prevent a duplication error
H+O2 + M = HO2 + M                             2.0E15  0.000  -870.
! H + O2 + M = HO2                            2.0E15  0.000  -870.
! H + O2 = HO2 + M                            2.0E15  0.000  -870.
OH+ +H+E = H2O                                   1.E19  0  0.0
O+HV = O(*)                                      1.E15  0.  0.
END                                            ! END statement is optional;
                                               ! <eof> condition is equivalent
```

Figure 7. Examples of Reaction Data.

TABLE IV. SUMMARY OF THE RULES FOR REACTION DATA

1. The first reaction line must start with the word REACTIONS (or REAC), and may include units definition(s).
2. The reaction description can begin anywhere on the line. All blank spaces, except those within Arrhenius coefficients, are ignored.
3. Each reaction description must have =, <=> or => between the last reactant and the first product.
4. Each reaction description must be contained on one line.
5. Three Arrhenius coefficients must appear in order (A_i, β_i, and E_i) on each Reaction line, separated from each other and from the reaction description by at least one blank space; no blanks are allowed within the numbers themselves.
6. There cannot be more than three reactants or three products in a reaction.
7. Comments are any and all characters following an exclamation mark.

Auxiliary Information Data

The format of an auxiliary information line is a character-string keyword followed by a slash-delimited (/) field containing an appropriate number of parameters (either integer, floating point, or E format).

If a reaction contains M as a reactant and/or product, auxiliary information lines may follow the reaction line to specify enhanced third body efficiencies of certain species [i.e., α_{ki}, Eq. (58)]. To define an enhanced third body efficiency, the keyword is the species name of the third body, and its one parameter is its enhanced efficiency factor. A species that acts as an enhanced third body must be declared as a species.

If a pressure-dependent reaction is indicated by a (+M) or by a species contained in parenthesis, say ($+H_2O$), then one or more auxiliary information lines must follow to define the fall-off parameters. The Arrhenius coefficients A_∞, β_∞, and E_∞ on the reaction line are for the high-pressure limit. For all fall-off reactions an auxiliary information line must follow to specify the low-pressure limit Arrhenius parameters. On this line the keyword LOW must appear, with three rate parameters A_0, β_0, and E_0 [Eq. (60)]. There are then three possible interpretations of the fall-off reaction:

> To define the Lindemann[8] formulation of a fall-off reaction, no additional fall-off parameters are defined.
>
> To define a Troe[9] fall-off reaction, in addition to the LOW parameters, the keyword TROE followed by three or four parameters must be included in the following order: a, T^{***}, T^{*}, and T^{**} [Eq.(68)]. The fourth parameter is optional and if omitted, the last term in Eq. (68) is not used.
>
> To define an SRI fall-off reaction,† in addition to the LOW parameters, the keyword SRI followed by three or five parameters must be included in the following order: a, b, c, d, and e (Eq. [69)]. The fourth and fifth parameters are optional. If only the first three are stated, then by default $d = 1$ and $e = 0$.

If a reaction contains HV as a reactant and/or product, an auxiliary information line may follow the reaction to specify radiation wavelength. For the wavelength specification, the keyword is HV and its one parameter is the wavelength in angstroms. This information is not used in the Gas-Phase Subroutine Library, but it is available to the user through a subroutine call.

For a reversible reaction, auxiliary information data may follow the reaction to specify Arrhenius parameters for the reverse-rate expression. Here, the three Arrhenius parameters (A_i, β_i, and E_i) for the reverse rate must follow the keyword REV. Using

† SRI refers to the formulation of Stewart et al.[10], who are at SRI International, Menlo Park, CA.

this option overrides the reverse rates that would be normally computed through the equilibrium constant, Eq. (53).

It sometimes happens that two or more reactions can involve the same set of reactants and products, but proceed through distinctly different processes. In these cases, it may be appropriate to state a reaction mechanism that has two or more reactions that are the same, but have different rate parameters. However, duplicate reactions are normally considered errors by the Interpreter; if the user requires duplication (e.g., the same reactants and products with different Arrhenius parameters), an auxiliary information statement containing the keyword DUP (with no parameters) must follow the reaction line of each duplicate reaction (including the first occurrence of the reaction that is duplicated). For example, if the user wishes to specify different rate expressions for each of three identical reactions, there must be three occurrences of the DUP keyword, one following each of the reactions.

To specify Landau-Teller parameters, the keyword LT must be followed by two parameters—the coefficients B_i and C_i from Eq. (72). The Arrhenius parameters A_i, β_i, and E_i are taken from the numbers specified on the reaction line itself. If reverse parameters are specified in a Landau-Teller reaction by a REV, the reverse Landau-Teller parameters must also be defined, with the keyword RLT and two coefficients B_i and C_i for the reverse rate.

Any number of auxiliary information lines may follow a reaction line, in any order, and any number of keywords or enhanced third bodies* may appear on an auxiliary information line; however, a keyword and its parameter(s) must appear on the same line.

Examples of equivalent ways to state auxiliary information are shown in Figure 8. The above rules are summarized in Table V.

Problems Having No Reactions

In some problems only information about the elements and species is needed (e.g., chemical equilibrium computations). For these it is not necessary to include reaction data. The Interpreter will create the LINKCK file, but it will not contain any reaction information. Therefore, no subroutines in the Gas-Phase Subroutine Library that deal with chemical reactions (e.g., chemical production rates) may be used.

---

* If more than ten species have enhanced third body efficiencies in any one reaction, some dimensioning needs to be changed in the Interpreter.

```
REACTIONS                                   CAL/MOLE

HCO+M=H+CO+M                                0.250E+15  0.000  16802.000 ! Warnatz
    CO/1.87/  H2/1.87/  CH4/2.81/  CO2/3./  H2O/5./

H+C2H4(+M)=C2H5(+M)                         0.221E+14  0.000  2066.000 ! Michael
    LOW / 6.369E27  -2.76  -54.0 /                  ! Lindemann fall-off reaction
    H2/2/  CO/2/  CO2/3/  H2O/5/            ! enhanced third-body efficiencies

CH3+CH3(+M)=C2H6(+M)                                    9.03E16  -1.18  654.
    LOW  / 3.18E41  -7.03  2762. /
    TROE / 0.6041  6927.  132. /        ! TROE fall-off reaction, using 3 parameters
    H2/2/  CO/2/  CO2/3/  H2O/5/            ! enhanced third body efficiencies

CH3+H(+M)=CH4(+M)                                       6.0E16  -1.0  0.0
    LOW / 8.0E26  -3.0  0.0/
    SRI  / 0.45  797.  979. /                   ! SRI fall-off reaction
    H2/2/  CO/2/  CO2/3/  H2O/5/

CH4+H=CH3+H2                                1.25E14  0  1.190E4  ! Westbrook
                                          REV/4.80E12  0  1.143E4/

! The following two reactions are acceptable duplicates:
    H2+O2 = 2OH                                         1.7E13  0  47780
        DUPLICATE
    H2+O2 = 2OH                                         1.0E13  0  47000
        DUPLICATE

H2(1)+H2O(000)=H2(0)+H2O(001)                                   2.89E15  0  0
                                        LT/-67 62.1/        ! Landau-Teller reaction
END                                                         !END line is optional
```

Figure 8. Examples of Auxiliary Information Definitions.

TABLE V. SUMMARY OF THE RULES FOR AUXILIARY INFORMATION DATA.

1. Auxiliary information lines may follow reaction lines that contain an M to specify enhanced third-body efficiencies, a reaction that contains an HV to specify the radiation wavelength, a reversible reaction to specify the reverse rate parameters explicitly, or any reaction that specifies Landau-Teller parameters. Auxiliary information *must* follow any duplicate reactions as well as all reactions that indicate pressure-dependent behavior by (+M) (i.e., provide fall-off parameters).
2. A species may have only one enhanced third body efficiency associated with it in any one reaction.
3. Only one radiation wavelength may be declared in a reaction.
4. The order in which the enhanced third body declarations are given is the order in which arrays of enhanced third body information are referenced in the subroutine package.
5. There cannot be more than ten enhanced third bodies in a reaction.
6. Keyword declarations may appear anywhere on the line, in any order.
7. Any number of keywords may appear on a line and more than one line may be used; however, a keyword and its parameter(s) must appear on the same line.
8. Keyword declarations that appear on the same line must be separated by at least one blank space.
9. Any blank spaces between a keyword and the first slash are ignored and any blanks between the slashes and parameter(s) are also ignored. However, no blank spaces are allowed within a keyword or a parameter.
10. All characters following an exclamation mark are comments.

Error Checks

The Interpreter checks each input line for proper syntax and writes self-explanatory diagnostic messages on logical file LOUT if errors are encountered. If an error condition occurs, the Interpreter continues to read and diagnose the input, but an error flag is written to the Linking file and Chemkin subroutine CKINIT will not initialize the work arrays. Therefore, the input must be error free before any of the Chemkin subroutines can be called.

The possibilities for an error condition are as follows:

Element Data

Atomic weight for an element or isotope is not declared, and the element is not found in the Interpreter's database.

Atomic weight has been declared, but not enclosed by two slashes (/).

If an element is declared twice, a diagnostic message is printed, but the duplicate is simply eliminated from consideration and is not considered a fatal error.

There are more elements than the Interpreter is dimensioned for (10).

Species Data

If a species is declared twice, a diagnostic message is printed, the duplicate is eliminated from consideration and is not considered a fatal error.

No thermodynamic data have been found for a declared species.

There are more species than the Interpreter is dimensioned for (100).

Thermodynamic Data

Thermodynamic Data are format sensitive and therefore provide possibilities for error if not formatted exactly as described by Table III.

An element in the thermodynamic data for a declared species has not been included in the element data.

With the THERMO ALL option, line 2 (Table III) is not found.

Reaction Data

A delimiter =>, <=>, or = between the reactants and the products is not found.

Three Arrhenius parameters are not found.

Reactants and/or products have not been properly delineated by a plus sign (+).

A species as a reactant or product has not been declared in the species data.

The reaction does not balance.

The charge of the reaction does not balance.

A reaction is a duplicate not declared by the auxiliary data keyword DUP.

A third-body species enclosed in parentheses in a fall-off reaction appears as reactant or product, but not both.

The third-body reactant is not the same as the third-body product in a fall-off reaction.

A species is a third-body in a fall-off reaction, and +M also appears in the reaction.

More than one +M or third-body as reactants and/or products.

HV declared as a reactant and as a product.

There are more reactions than the Interpreter is dimensioned for (500).

There are more then three reactants or three products.

Auxiliary Data

An unknown or misspelled keyword or enhanced third-body species name.

Parameters for a keyword not enclosed in slashes.

Wrong number of parameters for a keyword.

Duplicate keywords.

LOW, TROE, or SRI found after a reaction that did not have a species or M enclosed in parentheses.

LOW not found after a fall-off reaction.

TROE and SRI both found.

LT and REV found for a Landau-Teller reaction, but RLT not found.

LT or REV given for a fall-off reaction.

There are more than ten enhanced third bodies.

V. QUICK REFERENCE GUIDE TO THE GAS-PHASE SUBROUTINE LIBRARY

This chapter is arranged by topical area to provide a quick reference to each of the Gas-Phase Library Subroutines. In addition to the subroutine call list itself, the purpose of the subroutine is briefly described. Where appropriate, the description refers to an equation number in Chapter II. The page number given for each subroutine refers a detailed description of the subroutine call in Chapter VI.

Mnemonics

There are some good rules of thumb for explaining the subroutine naming conventions. All subroutines names begin with the letters CK so that Chemkin subroutines are easily recognized and so that they are likely different from any user subroutine names. The four remaining letters identify the purpose of the subroutine: The first one or two usually refer to the variable that is being computed; the last letters refer to either the input variables or the units.

State variables are denoted by P (pressure), T (temperature), Y (mass fraction), X (mole fraction), and C (molar concentration). Thermodynamic properties are referred to by CP and CV (specific heats), H (enthalpy), S (entropy), U (internal energy), G (Gibbs free energy), and A (Helmholtz free energy). The thermodynamic property subroutines may be called to return properties in mass units, denoted by MS or S as the last letter(s), or in molar units, denoted by ML or L as the last letter(s). The letter B (for the bar as in $\overline{C}_p$) in a thermodynamic property subroutine name indicates that it returns mean properties.

Subroutines that return net chemical production rates have a W (for $\dot{\omega}_k$) following the CK, and routines that return creation and destruction rates or creation rates and destruction times have a CD or a CT, respectively, following the CK. Rate-of-progress variables are denoted by Q and equilibrium constants by EQ.

The mnemonics for the variable names in the subroutine calls are roughly the same as for the subroutine names. However, because six letters can be used (only four are available in the subroutine names because CK occupies two), the mnemonics can be more explicit.

In most cases the subroutines are functionally identical with the corresponding routines in the original Chemkin. However, there are some cases where either the functionality is different or the call list is changed, but we have still used the same subroutine name. These routines are identified by an asterisk.

8. THERMODYNAMIC PROPERTIES (NONDIMENSIONAL)

9. THERMODYNAMIC PROPERTIES (MASS UNITS)

Page

14. EQUILIBRIUM CONSTANTS AND RATE OF PROGRESS VARIABLES.

15. UTILITIES

This subroutine is called to parse a character string, LINE, that is composed of several blank-delimited substrings. Each substring in LINE is compared with an ordered reference array of character strings, KRAY(*). For each substring in LINE that is also an entry in KRAY(*), the index position in KRAY(*) is returned in the integer array, NRAY(*). It is expected that each substring in LINE will be found in KRAY(*). If a substring is not found in KRAY(*), an error flag is returned. For example, after reading a line of species names, the subroutine might be called to assign Chemkin species index numbers to the list of species names, as is in the following example:

input: LINE = "OH N2 NO"
KRAY(*) = "H2" "O2" "N2" "H" "O" "N" "OH" "H2O" "NO"
NN = 9, the number of entries in KRAY(*)
LOUT = 6, a logical unit number for diagnostic messages.

output: NRAY(*) = 7, 3, 9, the index numbers of the entries in KRAY(*) corresponding to the substrings in LINE.
NF = 3, the number of correspondences found.
KERR = .FALSE.

Returns the character string representation of an integer, and the effective length of the string.

This subroutine is called to parse a character string, LINE, that is composed of several blank-delimited substrings. The final segment of LINE containing NPAR substrings is found, beginning in the ISTART column; this segment is then copied into the character string IPAR. This allows format-free input of combined alpha-numeric data. For example, after reading a line containing alpha-numeric information ending with several numbers, the subroutine might be called to find the segment of a line containing specific numbers:

input: LINE = "t1 t2 dt 300.0 3.0E3 50"
NPAR = 3, the number of substrings requested
output: IPAR = "300.0 3.0E3 50"
ISTART = 11, the starting column in LINE of the NPAR substrings

Returns the character string representation of a real number, and the effective length of the string.

SUBROUTINE CKSNUM (LINE,NEXP,LOUT,KRAY,NN,KNUM,NVAL,RVAL,KERR) 94

This subroutine is called to parse a character string, LINE, that is composed of several blank-delimited substrings. It is expected that the first substring in LINE is also an entry in a reference array of character strings, KRAY(*), in which case the index position in KRAY(*) is returned as KNUM, otherwise an error flag is returned. The substrings following the first are expected to represent numbers, and are converted to elements of the array RVAL(*). If NEXP substrings are not found, an error flag will be returned. This allows format-free input of combined alpha-numeric data. For example, after reading a line containing a species name followed by several numerical values, the subroutine might be called to find a Chemkin species index and convert the other substrings to real values:

input: LINE = "N2 1.2"
NEXP = 1, the number of values expected
LOUT = 6, a logical unit number on which to write diagnostic messages.
KRAY(*) = "H2" "O2" "N2" "H" "O" "N" "OH" "H2O" "NO"
NN = 9, the number of entries in KRAY(*)

output: KNUM = 3, the index number of the element in KRAY(*) which corresponds to the first substring in LINE
NVAL = 1, the number of values found in LINE following the first substring
RVAL(*) = 1.200E+00, the substring converted to a number
KERR = .FALSE.

SUBROUTINE CKSUBS (LINE, LOUT, NDIM, SUB, NFOUND, KERR) 95

Returns an array of the blank-delimited substrings in a character string, and the number of substrings found.

SUBROUTINE CKXNUM (LINE, NEXP, LOUT, NVAL, RVAL, KERR) 103

This subroutine is called to parse a character string, LINE, that is composed of several blank-delimited substrings. Each substring is expected to represent a number, which is converted to entries in the array of real numbers, RVAL(*). NEXP is the number of values expected, and NVAL is the number of values found. If NEXP values are required, the user can compare NVAL against NEXP and decide how to proceed. This allows format-free input of numerical data. For example:

input: LINE = " 0.170E+14 0 47780.0"
NEXP = 3, the number of values requested
LOUT = 6, a logical unit number for diagnostic messages.

output: NVAL = 3, the number of values found
RVAL(*) = 1.700E+13, 0.000E+00, 4.778E+04

VI. ALPHABETICAL LISTING OF THE GAS-PHASE SUBROUTINE LIBRARY WITH DETAILED DESCRIPTIONS OF THE CALL LISTS

Each subroutine in the Gas-Phase Subroutine Library is described in this chapter, together with a detailed description of the variables in the call lists. For all arrays, information is given on the required dimensioning in the calling program. For all variables having units, the cgs units are stated. In many cases a reference to the most applicable equation in Chapter II is also given.

In most cases the subroutines are functionally identical with the corresponding routines in the original Chemkin. However, there are some cases where either the functionality is different or the call list is changed, but we have still used the same subroutine name. These routines are identified by an asterisk.

```
CKABE      CKABE      CKABE      CKABE      CKABE      CKABE      CKABE
           ****************************************
                      *************************
                                 ******
```

SUBROUTINE CKABE (ICKWRK, RCKWRK, RA, RB, RE)
Returns the Arrhenius coefficients of the reactions; see Eq. (52).

INPUT
- ICKWRK - Array of integer workspace
  - Data type - integer array
  - Dimension ICKWRK(*) at least LENIWK.
- RCKWRK - Array of real work space.
  - Data type - real array
  - Dimension RCKWRK(*) at least LENRWK.

OUTPUT
- RA - Pre-exponential constants for the reactions.
  - cgs units - mole-cm-sec-K
  - Data type - real array
  - Dimension RA(*) at least II, the total number of reactions.
- RB - Temperature dependence exponents for the reactions
  - cgs units - none
  - Data type - real array
  - Dimension RB(*) at least II, the total number of reactions.
- RE - Activation energies for the reactions.
  - cgs units - kelvins
  - Data type - real array
  - Dimension RE(*) at least II, the total number of reactions.

```
CKABML     CKABML     CKABML     CKABML     CKABML     CKABML     CKABML
           ****************************************
                      *************************
                                 ******
```

SUBROUTINE CKABML (P, T, X, ICKWRK, RCKWRK, ABML)*
Returns the Helmholtz free energy of the mixture in molar units, given the pressure, temperature, and mole fractions; see Eq. (46).

INPUT
- P - Pressure.
  - cgs units - dynes/cm**2
  - Data type - real scalar
- T - Temperature.
  - cgs units - kelvins
  - Data type - real scalar
- X - Mole fractions of the species.
  - cgs units - none
  - Data type - real array
  - Dimension X(*) at least KK, the total number of species.
- ICKWRK - Array of integer workspace
  - Data type - integer array
  - Dimension ICKWRK(*) at least LENIWK.
- RCKWRK - Array of real work space.
  - Data type - real array
  - Dimension RCKWRK(*) at least LENRWK.

OUTPUT
- ABML - Mean Helmholtz free energy in molar units
  - cgs units - ergs/mole
  - Data type - real scalar

```
CKABMS     CKABMS     CKABMS     CKABMS     CKABMS     CKABMS     CKABMS
           ******************************************
                      ************************
                                 ******

SUBROUTINE CKABMS (P, T, Y, ICKWRK, RCKWRK, ABMS)*
   Returns the mean Helmholtz free energy of the mixture in mass units,
   given the pressure, temperature and mass fractions; see Eq. (47).

INPUT
   P       - Pressure.
                  cgs units - dynes/cm**2
                  Data type - real scalar
   T       - Temperature.
                  cgs units - kelvins
                  Data type - real scalar
   Y       - Mass fractions of the species.
                  cgs units - none
                  Data type - real array
                  Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                  Data type - integer array
                  Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                  Data type - real array
                  Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   ABMS    - Mean Helmholtz free energy in mass units.
                  cgs units - ergs/gm
                  Data type - real scalar


CKAML      CKAML      CKAML      CKAML      CKAML      CKAML      CKAML
           ******************************************
                      ************************
                                 ******

SUBROUTINE CKAML (T, ICKWRK, RCKWRK, AML)
   Returns the standard state Helmholtz free energies in molar units;
   see Eq. (25).

INPUT
   T       - Temperature.
                  cgs units - kelvins
                  Data type - real scalar
   ICKWRK  - Array of integer workspace
                  Data type - integer array
                  Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                  Data type - real array
                  Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   AML     - Standard state Helmholtz free energies in molar units
             for the species.
                  cgs units - ergs/mole
                  Data type - real array
                  Dimension AML(*) at least KK, the total number of species.
```

```
CKAMS      CKAMS      CKAMS      CKAMS      CKAMS      CKAMS      CKAMS
           ***********************************************
                      **************************
                                 ******

SUBROUTINE CKAMS (T, ICKWRK, RCKWRK, AMS)
   Returns the standard state Helmholtz free energies in mass
   units; see Eq. (32).

INPUT
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
 OUTPUT
   AMS     - Standard state Helmholtz free energies in mass units
             for the species.
                   cgs units - ergs/gm
                   Data type - real array
                   Dimension AMS(*) at least KK, the total number of species.


CKATHM     CKATHM     CKATHM     CKATHM     CKATHM     CKATHM     CKATHM
           ***********************************************
                      **************************
                                 ******

SUBROUTINE CKATHM (NDIM1, NDIM2, ICKWRK, RCKWRK, MAXTP, NT, TMP, A)
   Returns the coefficients of the fits for thermodynamic properties
   of the species:see Eqs. (19) - (21).

INPUT
   NDIM1   - First dimension of the three-dimensional array of thermodynamic
             fit coefficients, A; NDIM1 must be at least NCP2, the total
             number of coefficients for one temperature range.
   NDIM2   - Second dimension of the three-dimensional array of
             thermodynamic fit coefficients, A; NDIM2 must be at
             least MXPT-1, the total number of temperature ranges.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
 OUTPUT
   NT      - Number of temperatures used for fitting coefficients of
             thermodynamic properties for the species.
                   Data type - integer array
                   Dimension NT(*) at least KK, the total number of species.
   TMP     - Common temperatures dividing the thermodynamic fits for
             the species.
                   cgs units - kelvins
                   Data type - real array
                   Dimension TMP(MAXT,*) exactly MAXT for the first
                   dimension (the maximum number of temperatures
                   allowed for a species) , and at least KK for the
                   second dimension (the total number of species)
   A       - Three dimensional array of fit coefficients to the
             thermodynamic data for the species
             The indices in A(N,L,K) mean-
             N = 1,NN are polynomial coefficients in CP/R
                 CP/R(K)=A(1,L,K) + A(2,L,K)*T + A(3,L,K)*T**2 + ...
             N = NN+1 is a6 in Eq. (20).
             N = NN+2 is a7 in Eq. (21).
             L = 1 ... MXTP-1 is for each temperature range.
             K  is  the  species index
                   Data type - real array
                   Dimension A(NPCP2,NDIM2,*) exactly NPCP2 and MXTP-1
                   for the first and second dimensions and at least
                   KK for the third.
```

```
CKAWT      CKAWT      CKAWT      CKAWT      CKAWT      CKAWT      CKAWT
           ****************************************
                      **************************
                                ******

SUBROUTINE CKAWT (ICKWRK, RCKWRK, AWT)
   Returns the atomic weights of the elements

INPUT
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   AWT    - Atomic weights of the elements.
                 cgs units - gm/mole
                 Data type - real array
                 Dimension AWT(*) at least MM, the total number of
                 elements in the problem.


CKCDC      CKCDC      CKCDC      CKCDC      CKCDC      CKCDC      CKCDC
           ****************************************
                      **************************
                                ******

SUBROUTINE CKCDC (T, C, ICKWRK, RCKWRK, CDOT, DDOT)
   Returns the molar creation and destruction rates of the species
   given the temperature and molar concentrations; see Eq. (73).

INPUT
   T      - Temperature.
                 cgs units - kelvins
                 Data type - real scalar
   C      - Molar concentrations of the species.
                 cgs units - mole/cm**3
                 Data type - real array
                 Dimension C(*) at least KK, the total number of species.
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CDOT   - Chemical molar creation rates of the species.
                 cgs units - mole/(cm**3*sec)
                 Data type - real array
                 Dimension CDOT(*) at least KK, the total number of species.
   DDOT   - Chemical molar destruction rates of the species.
                 cgs units - moles/(cm**3*sec)
                 Data type - real array
                 Dimension DDOT(*) at least KK, the total number of species.
```

```
CKCDXP    CKCDXP    CKCDXP    CKCDXP    CKCDXP    CKCDXP    CKCDXP
          ******************************************
                    ************************
                              ******

SUBROUTINE CKCDXP (P, T, X, ICKWRK, RCKWRK, CDOT, DDOT)
   Returns the molar creation and destruction rates of the species
   given pressure, temperature and mole fractions; see Eq. (73).

INPUT
   P       - Pressure.
                   cgs units - dynes/cm**2
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   X       - Mole fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension X(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CDOT    - Chemical molar creation rates of the species.
                   cgs units - mole/(cm**3*sec)
                   Data type - real array
                   Dimension CDOT(*) at least KK, the total number of species.
   DDOT    - Chemical molar destruction rates of the species.
                   cgs units - moles/(cm**3*sec)
                   Data type - real array
                   Dimension DDOT(*) at least KK, the total number of species.


CKCDXR    CKCDXR    CKCDXR    CKCDXR    CKCDXR    CKCDXR    CKCDXR
          ******************************************
                    ************************
                              ******

SUBROUTINE CKCDXR (RHO, T, X, ICKWRK, RCKWRK, CDOT, DDOT)
   Returns the molar creation and destruction rates of the species
   given the mass density, temperature and mole fractions; see Eq. (73).

INPUT
   RHO     - Mass density.
                   cgs units - gm/cm**3
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   X       - Mole fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension X(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CDOT    - Chemical molar creation rates of the species.
                   cgs units - mole/(cm**3*sec)
                   Data type - real array
                   Dimension CDOT(*) at least KK, the total number of species.
   DDOT    - Chemical molar destruction rates of the species.
                   cgs units - moles/(cm**3*sec)
                   Data type - real array
                   Dimension DDOT(*) at least KK, the total number of species.
```

```
CKCDYP    CKCDYP    CKCDYP    CKCDYP    CKCDYP    CKCDYP    CKCDYP
          ******************************************
                    ************************
                              ******

SUBROUTINE CKCDYP (P, T, Y, ICKWRK, RCKWRK, CDOT, DDOT)
   Returns the molar creation and destruction rates of the species
   given mass density, temperature and mass fractions; see Eq. (73).

INPUT
   P       - Pressure.
                   cgs units - dynes/cm**2
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   Y       - Mass fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
 OUTPUT
   CDOT    - Chemical molar creation rates of the species.
                   cgs units - mole/(cm**3*sec)
                   Data type - real array
                   Dimension CDOT(*) at least KK, the total number of species.
   DDOT    - Chemical molar destruction rates of the species.
                   cgs units - moles/(cm**3*sec)
                   Data type - real array
                   Dimension DDOT(*) at least KK, the total number of species.


CKCDYR    CKCDYR    CKCDYR    CKCDYR    CKCDYR    CKCDYR    CKCDYR
          ******************************************
                    ************************
                              ******

SUBROUTINE CKCDYR (RHO, T, Y, ICKWRK, RCKWRK, CDOT, DDOT)
   Returns the molar creation and destruction rates of the species
   given the mass density, temperature and mass fractions: see Eq. (73).

INPUT
   RHO     - Mass density.
                   cgs units - gm/cm**3
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   Y       - Mass fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CDOT    - Chemical molar creation rates of the species.
                   cgs units - mole/(cm**3*sec)
                   Data type - real array
                   Dimension CDOT(*) at least KK, the total number of species.
   DDOT    - Chemical molar destruction rates of the species.
                   cgs units - moles/(cm**3*sec)
                   Data type - real array
                   Dimension DDOT(*) at least KK, the total number of species.
```

```
CKCHRG    CKCHRG    CKCHRG    CKCHRG    CKCHRG    CKCHRG    CKCHRG
          ****************************************
                    ************************
                          ******

SUBROUTINE CKCHRG (ICKWRK, RCKWRK, KCHARG)
   Returns the electronic charges of the species

INPUT
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   KCHARG - Electronic charges of the species; KCHARG(K)=-2
            indicates that the Kth species has two excess electrons.
                 Data type - integer array
                 Dimension KCHARG(*) at least KK, the total number of
                 species.


CKCOMP    CKCOMP    CKCOMP    CKCOMP    CKCOMP    CKCOMP    CKCOMP
          ****************************************
                    ************************
                          ******

SUBROUTINE CKCOMP (IST, IRAY, II, I)*
   Returns the index of an element of a reference character string array
   that corresponds to a character string; leading and trailing blanks are
   ignored.

INPUT
   IST    - A character string
                 Data type - CHARACTER*(*)
   IRAY   - An array of character strings
                 Data type - CHARACTER*(*)
                 Dimension at least II
   II     - The length of IRAY
                 Data type - integer scaler.
OUTPUT
   I      - The first integer location in IRAY in which IST corresponds to
            IRAY(I); if IST is not also an entry in IRAY, then I = 0.


CKCONT    CKCONT    CKCONT    CKCONT    CKCONT    CKCONT    CKCONT
          ****************************************
                    ************************
                          ******

SUBROUTINE CKCONT (K, Q, ICKWRK, RCKWRK, CIK)
   Returns the contributions of the reactions to the molar
   production rate of a species; see Eqs. (49) and (51).

INPUT
   K      - Integer species number.
                 Data type - integer scalar
   Q      - Rates of progress for the reactions.
                 cgs units - moles/(cm**3*sec)
                 Data type - real array
                 Dimension Q(*) at least II, the total number of reactions.
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CIK    - Contributions of the reactions to the molar production
            rate of the Kth species
                 cgs units - mole/(cm**3*sec)
                 Data type - real array
                 Dimension CIK(*) at least II, the total number of
                 reactions.
```

```
CKCPBL     CKCPBL     CKCPBL     CKCPBL     CKCPBL     CKCPBL     CKCPBL
           ******************************************
                      ************************
                                 ******

SUBROUTINE CKCPBL (T, X, ICKWRK, RCKWRK, CPBML)
   Returns the mean specific heat at constant pressure; see Eq. (33).

INPUT
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   X       - Mole fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension X(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
 OUTPUT
   CPBML   - Mean specific heat at constant pressure in molar units.
                   cgs units - ergs/(mole*K)
                   Data type - real scalar


CKCPBS     CKCPBS     CKCPBS     CKCPBS     CKCPBS     CKCPBS     CKCPBS
           ******************************************
                      ************************
                                 ******

SUBROUTINE CKCPBS (T, Y, ICKWRK, RCKWRK, CPBMS)
   Returns the mean specific heat at constant pressure; see Eq. (34).

INPUT
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   Y       - Mass fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CPBMS   - Mean specific heat at constant pressure in mass units.
                   cgs units - ergs/(gm*K)
                   Data type - real scalar


CKCPML     CKCPML     CKCPML     CKCPML     CKCPML     CKCPML     CKCPML
           ******************************************
                      ************************
                                 ******

SUBROUTINE CKCPML (T, ICKWRK, RCKWRK, CPML)
   Returns the specific heats at constant pressure in molar units.

INPUT
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CPML    - Specific heats at constant pressure in molar units
             for the species.
                   cgs units - ergs/(mole*K)
                   Data type - real array
                   Dimension CPML(*) at least KK, the total number of species.
```

```
CKCPMS     CKCPMS     CKCPMS     CKCPMS     CKCPMS     CKCPMS     CKCPMS
           ****************************************
                     ************************
                               ******

SUBROUTINE CKCPMS (T, ICKWRK, RCKWRK, CPMS)
   Returns the specific heats at constant pressure in mass units;
   see Eq. (26).

INPUT
   T      - Temperature.
                 cgs units - kelvins
                 Data type - real scalar
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CPMS   - Specific heats at constant pressure in mass units
            for the species.
                 cgs units - ergs/(gm*K)
                 Data type - real array
                 Dimension CPMS(*) at least KK, the total number of species.


CKCPOR     CKCPOR     CKCPOR     CKCPOR     CKCPOR     CKCPOR     CKCPOR
           ****************************************
                     ************************
                               ******

SUBROUTINE CKCPOR (T, ICKWRK, RCKWRK, CPOR)
   Returns the nondimensional specific heats at constant pressure;
   see Eq. (19).

INPUT
   T      - Temperature.
                 cgs units - kelvins
                 Data type - real scalar
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CPOR   - Nondimensional specific heats at constant pressure
            for the species.
                 cgs units - none
                 Data type - real array
                 Dimension CPOR(*) at least KK, the total number of species.
```

CKCRAY CKCRAY CKCRAY CKCRAY CKCRAY CKCRAY CKCRAY

```
SUBROUTINE CKCRAY (LINE, NN, KRAY, LOUT, NF, NRAY, KERR)
   This subroutine is called to parse a character string, LINE, that is
   composed of several blank-delimited substrings. Each substring in LINE
   is compared with an ordered reference array of character strings,
   KRAY(*). For each substring in LINE that is also an entry in KRAY(*),
   the index position in KRAY(*) is returned in the integer array, NRAY(*).
   It is expected that each substring in LINE will be found in KRAY(*). If
   a substring cannot be found in KRAY(*) an error flag will be returned.
   For example, after reading a line of species names, the subroutine
   might be called to assign Chemkin species index numbers to the list of
   species names. This application is made more concrete in the following
   example:

   input:  LINE     = "OH  N2  NO"
           KRAY(*)  = "H2" "O2" "N2" "H" "O" "N" "OH" "H2O" "NO"
           NN       = 9, the number of entries in KRAY(*)
           LOUT     = 6, a logical unit number on which to write
                      diagnostic messages.
   output: NRAY(*)  = 7, 3, 9, the index numbers of the entries
                      in KRAY(*) corresponding to the substrings
                      in LINE
           NF       = 3, the number of correspondences found.
           KERR     = .FALSE.

INPUT
   LINE - A character string.
   KRAY - An array of character strings.
   NN   - Total number of character strings in KRAY
               Data type - integer scalar
   LOUT - Output unit for error messages
               Data type - integer scalar
OUTPUT
   NRAY - Index numbers of the elements of KRAY which
          correspond to the substrings in LINE
               Data type - integer array
   NF   - Number of correspondences found.
               Data type - integer scalar
   KERR - Error flag.
               Data type - logical
```

```
CKCTC      CKCTC      CKCTC      CKCTC      CKCTC      CKCTC      CKCTC
           ******************************************
                      ************************
                                ******
```

```
SUBROUTINE CKCTC (T, C, ICKWRK, RCKWRK, CDOT, TAU)
   Returns the molar creation rates and characteristic destruction
   times of the species given temperature and molar concentrations;
   see Eqs. (76) and (78).

INPUT
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   C       - Molar concentrations of the species.
                   cgs units - mole/cm**3
                   Data type - real array
                   Dimension C(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CDOT    - Chemical molar creation rates of the species.
                   cgs units - mole/(cm**3*sec)
                   Data type - real array
                   Dimension CDOT(*) at least KK, the total number of species.
   TAU     - Characteristic destruction times of the species.
                   cgs units - sec
                   Data type - real array
                   Dimension TAU(*) at least KK, the total number of species.
```

```
CKCTX      CKCTX      CKCTX      CKCTX      CKCTX      CKCTX      CKCTX
           ******************************************
                      ************************
                                ******
```

```
SUBROUTINE CKCTX (C, ICKWRK, RCKWRK, X)
   Returns the mole fractions given the molar concentrations; see Eq. (13).

INPUT
   C       - Molar concentrations of the species.
                   cgs units - mole/cm**3
                   Data type - real array
                   Dimension C(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   X       - Mole fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension X(*) at least KK, the total number of species.
```

```
CKCTXP    CKCTXP    CKCTXP    CKCTXP    CKCTXP    CKCTXP    CKCTXP
          ****************************************
                    ************************
                              ******

SUBROUTINE CKCTXP (P, T, X, ICKWRK, RCKWRK, CDOT, TAU)
   Returns the molar creation rates and characteristic destruction
   times of the species given the pressure, temperature and mole
   fractions; see Eqs. (76) and (78)

INPUT
   P       - Pressure.
                   cgs units - dynes/cm**2
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   X       - Mole fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension X(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CDOT    - Chemical molar creation rates of the species.
                   cgs units - mole/(cm**3*sec)
                   Data type - real array
                   Dimension CDOT(*) at least KK, the total number of species.
   TAU     - Characteristic destruction times of the species.
                   cgs units - sec
                   Data type - real array
                   Dimension TAU(*) at least KK, the total number of species.


CKCTXR    CKCTXR    CKCTXR    CKCTXR    CKCTXR    CKCTXR    CKCTXR
          ****************************************
                    ************************
                              ******

SUBROUTINE CKCTXR (RHO, T, X, ICKWRK, RCKWRK, CDOT, TAU)
   Returns the molar creation rates and characteristic destruction
   times of the species given the mass density, temperature and
   mole fractions; see Eqs. (76) and (78).

INPUT
   RHO     - Mass density.
                   cgs units - gm/cm**3
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   X       - Mole fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension X(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CDOT    - Chemical molar creation rates of the species.
                   cgs units - mole/(cm**3*sec)
                   Data type - real array
                   Dimension CDOT(*) at least KK, the total number of species.
   TAU     - Characteristic destruction times of the species.
                   cgs units - sec
                   Data type - real array
                   Dimension TAU(*) at least KK, the total number of species.
```

```
CKCTY      CKCTY      CKCTY      CKCTY      CKCTY      CKCTY      CKCTY
           ******************************************
                      ************************
                                 ******

SUBROUTINE CKCTY (C, ICKWRK, RCKWRK, Y)
   Returns the mass fractions given the molar concentrations; see Eq. (12).

INPUT
   C       - Molar concentrations of the species.
                   cgs units - mole/cm**3
                   Data type - real array
                   Dimension C(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   Y       - Mass fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.


CKCTYP     CKCTYP     CKCTYP     CKCTYP     CKCTYP     CKCTYP     CKCTYP
           ******************************************
                      ************************
                                 ******

SUBROUTINE CKCTYP (P, T, Y, ICKWRK, RCKWRK, CDOT, TAU)
   Returns the molar creation rates and characteristic destruction
   times of the species given the mass density, temperature and
   mass fractions; see Eqs. (76) and (78).

INPUT
   P       - Pressure.
                   cgs units - dynes/cm**2
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   Y       - Mass fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CDOT    - Chemical molar creation rates of the species.
                   cgs units - mole/(cm**3*sec)
                   Data type - real array
                   Dimension CDOT(*) at least KK, the total number of species.
   TAU     - Characteristic destruction times of the species.
                   cgs units - sec
                   Data type - real array
                   Dimension TAU(*) at least KK, the total number of species.
```

```
CKCTYR    CKCTYR    CKCTYR    CKCTYR    CKCTYR    CKCTYR    CKCTYR
          ******************************************
                    ***************************
                              ******

SUBROUTINE CKCTYR (RHO, T, Y, ICKWRK, RCKWRK, CDOT, TAU)
   Returns the molar creation rates and characteristic destruction
   times of the species given the mass density, temperature and
   mass fractions; see Eqs. (76) and (78).

INPUT
   RHO    - Mass density.
            cgs units - gm/cm**3
            Data type - real scalar
   T      - Temperature.
            cgs units - kelvins
            Data type - real scalar
   Y      - Mass fractions of the species.
            cgs units - none
            Data type - real array
            Dimension Y(*) at least KK, the total number of species.
   ICKWRK - Array of integer workspace
            Data type - integer array
            Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
            Data type - real array
            Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CDOT   - Chemical molar creation rates of the species.
            cgs units - mole/(cm**3*sec)
            Data type - real array
            Dimension CDOT(*) at least KK, the total number of species.
   TAU    - Characteristic destruction times of the species.
            cgs units - sec
            Data type - real array
            Dimension TAU(*) at least KK, the total number of species.


CKCVBL    CKCVBL    CKCVBL    CKCVBL    CKCVBL    CKCVBL    CKCVBL
          ******************************************
                    ***************************
                              ******

SUBROUTINE CKCVBL (T, X, ICKWRK, RCKWRK, CVBML)
   Returns the mean specific heat at constant volume in molar units;
   see Eq. (35).

INPUT
   T      - Temperature.
            cgs units - kelvins
            Data type - real scalar
   X      - Mole fractions of the species.
            cgs units - none
            Data type - real array
            Dimension X(*) at least KK, the total number of species.
   ICKWRK - Array of integer workspace
            Data type - integer array
            Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space
            Data type - real array
            Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CVBML  - Mean specific heat at constant volume in molar units.
            cgs units - ergs/(mole*K)
            Data type - real scalar
```

```
CKCVBS     CKCVBS     CKCVBS     CKCVBS     CKCVBS     CKCVBS     CKCVBS
           ********************************************
                      ************************
                            ******

SUBROUTINE CKCVBS (T, Y, ICKWRK, RCKWRK, CVBMS)
   Returns the Mean specific heat at constant volume in mass units;
   see Eq. (36).

INPUT
   T       - Temperature.
                  cgs units - kelvins
                  Data type - real scalar
   Y       - Mass fractions of the species
                  cgs units - none
                  Data type - real array
                  Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                  Data type - integer array
                  Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                  Data type - real array
                  Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CVBMS   - Mean specific heat at constant volume in mass units
                  cgs units - ergs/(gm*K)
                  Data type - real scalar


CKCVML     CKCVML     CKCVML     CKCVML     CKCVML     CKCVML     CKCVML
           ********************************************
                      ************************
                            ******

SUBROUTINE CKCVML (T, ICKWRK, RCKWRK, CVML)
   Returns the specific heats in constant volume in molar units;
   see Eq. (22).

INPUT
   T       - Temperature.
                  cgs units - kelvins
                  Data type - real scalar
   ICKWRK  - Array of integer workspace
                  Data type - integer array
                  Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                  Data type - real array
                  Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CVML    - Specific heats at constant volume in molar units
             for the species.
                  cgs units - ergs/(mole*K)
                  Data type - real array
                  Dimension CVML(*) at least KK, the total number of species.
```

```
CKCVMS     CKCVMS     CKCVMS     CKCVMS     CKCVMS     CKCVMS     CKCVMS
           ******************************************
                     ************************
                               ******

SUBROUTINE CKCVMS (T, ICKWRK, RCKWRK, CVMS)
   Returns the specific heats at constant volume in mass units;
   see Eq. (29).

INPUT
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   CVMS    - Specific heats at constant volume in mass units
             for the species.
                   cgs units - ergs/(gm*K)
                   Data type - real array
                   Dimension CVMS(*) at least KK, the total number of species.


CKEQC      CKEQC      CKEQC      CKEQC      CKEQC      CKEQC      CKEQC
           ******************************************
                     ************************
                               ******

SUBROUTINE CKEQC (T, C, ICKWRK, RCKWRK, EQKC)
   Returns the equilibrium constants of the reactions given
   temperature and molar concentrations; see Eq. (54)

INPUT
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   C       - Molar concentrations of the species
                   cgs units - mole/cm**3
                   Data type - real array
                   Dimension C(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   EQKC    - Equilibrium constants in concentration units for the reactions:
                   cgs units - (mole/cm**3)**some power, depending on
                               the reaction.
                   Data type - real array
                   Dimension EQKC(*) at least II, the total number of
                   reactions.
```

```
CKEQXP    CKEQXP    CKEQXP    CKEQXP    CKEQXP    CKEQXP    CKEQXP
          ******************************************
                    ************************
                              ******

SUBROUTINE CKEQXP (P, T, X, ICKWRK, RCKWRK, EQKC)
   Returns the equilibrium constants for the reactions given
   pressure, temperature and mole fractions; see Eq. (54).

INPUT
   P       - Pressure.
                   cgs units - dynes/cm**2
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   X       - Mole fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension X(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   EQKC    - Equilibrium constants in concentration units for the reactions:
                   cgs units - (mole/cm**3)**some power, depending on the
                               reaction.
                   Data type - real array
                   Dimension EQKC(*) at least II, the total number of
                   reactions.


CKEQXR    CKEQXR    CKEQXR    CKEQXR    CKEQXR    CKEQXR    CKEQXR
          ******************************************
                    ************************
                              ******

SUBROUTINE CKEQXR (RHO, T, X, ICKWRK, RCKWRK, EQKC)
   Returns the equilibrium constants of the reactions given mass
   density, temperature and mole fractions; see Eq. (54).

INPUT
   RHO     - Mass density.
                   cgs units - gm/cm**3
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   X       - Mole fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension X(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   EQKC    - Equilibrium constants in concentration units for the reactions:
                   cgs units - (mole/cm**3)**some power, depending on the
                               reaction.
                   Data type - real array
                   Dimension EQKC(*) at least II, the total number of
                   reactions.
```

```
CKEQYP    CKEQYP    CKEQYP    CKEQYP    CKEQYP    CKEQYP    CKEQYP
          ********************************************
                    ************************
                              ******

SUBROUTINE CKEQYP (P, T, Y, ICKWRK, RCKWRK, EQKC)
   Returns the equilibrium constants for the reactions given
   pressure, temperature and mass fractions; see Eq. (54).

INPUT
   P       - Pressure.
                   cgs units - dynes/cm**2
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   Y       - Mass fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   EQKC    - Equilibrium constants in concentration units for the reactions:
                   cgs units - (mole/cm**3)**some power, depending on the
                               reaction.
                   Data type - real array
                   Dimension EQKC(*) at least II, the total number of
                   reactions.


CKEQYR    CKEQYR    CKEQYR    CKEQYR    CKEQYR    CKEQYR    CKEQYR
          ********************************************
                    ************************
                              ******

SUBROUTINE CKEQYR (RHO, T, Y, ICKWRK, RCKWRK, EQKC)
   Returns the equilibrium constants of the reactions given mass
   density, temperature and mass fractions; see Eq. (54).

INPUT
   RHO     - Mass density.
                   cgs units - gm/cm**3
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   Y       - Mass fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   EQKC    - Equilibrium constants in concentration units for the reactions
                   cgs units - (mole/cm**3)**some power, depending on the
                               reaction.
                   Data type - real array
                   Dimension EQKC(*) at least II, the total number of
                   reactions.
```

```
CKGBML    CKGBML    CKGBML    CKGBML    CKGBML    CKGBML    CKGBML
          ****************************************
                    **************************
                              ******
```

SUBROUTINE CKGBML (P, T, X, ICKWRK, RCKWRK, GBML)*
Returns the mean Gibbs free energy of the mixture in molar units, given the pressure, temperature and mole fractions; see Eq. (44).

INPUT
- P - Pressure.
  cgs units - dynes/cm**2
  Data type - real scalar
- T - Temperature.
  cgs units - kelvins
  Data type - real scalar
- X - Mole fractions of the species.
  cgs units - none
  Data type - real array
  Dimension X(*) at least KK, the total number of species.
- ICKWRK - Array of integer workspace
  Data type - integer array
  Dimension ICKWRK(*) at least LENIWK.
- RCKWRK - Array of real work space.
  Data type - real array
  Dimension RCKWRK(*) at least LENRWK.

OUTPUT
- GBML - Mean Gibbs free energy in molar units.
  cgs units - ergs/mole
  Data type - real scalar

```
CKGBMS    CKGBMS    CKGBMS    CKGBMS    CKGBMS    CKGBMS    CKGBMS
          ****************************************
                    **************************
                              ******
```

SUBROUTINE CKGBMS (P, T, Y, ICKWRK, RCKWRK, GBMS)*
Returns the mean Gibbs free energy of the mixture in mass units, given the pressure, temperature, and mass fractions; see Eq. (45).

INPUT
- P - Pressure.
  cgs units - dynes/cm**2
  Data type - real scalar
- T - Temperature.
  cgs units - kelvins
  Data type - real scalar
- Y - Mass fractions of the species.
  cgs units - none
  Data type - real array
  Dimension Y(*) at least KK, the total number of species.
- ICKWRK - Array of integer workspace
  Data type - integer array
  Dimension ICKWRK(*) at least LENIWK.
- RCKWRK - Array of real work space.
  Data type - real array
  Dimension RCKWRK(*) at least LENRWK.

OUTPUT
- GBMS - Mean Gibbs free energy in mass units.
  cgs units - ergs/gm
  Data type - real scalar

```
CKGML      CKGML      CKGML      CKGML      CKGML      CKGML      CKGML
          ********************************************
                     **************************
                          ******

SUBROUTINE CKGML (T, ICKWRK, RCKWRK, GML)
   Returns the standard state Gibbs free energies in molar units;
   see Eq. (24).

INPUT
   T       - Temperature.
                  cgs units - kelvins
                  Data type - real scalar
   ICKWRK  - Array of integer workspace
                  Data type - integer array
                  Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                  Data type - real array
                  Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   GML     - Standard state gibbs free energies in molar units
             for the species.
                  cgs units - ergs/mole
                  Data type - real array
                  Dimension GML(*) at least KK, the total number of species.


CKGMS      CKGMS      CKGMS      CKGMS      CKGMS      CKGMS      CKGMS
          ********************************************
                     **************************
                          ******

SUBROUTINE CKGMS (T, ICKWRK, RCKWRK, GMS)
   Returns the standard state Gibbs free energies in mass units;
   see Eq. (31).

INPUT
   T       - Temperature.
                  cgs units - kelvins
                  Data type - real scalar
   ICKWRK  - Array of integer workspace
                  Data type - integer array
                  Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                  Data type - real array
                  Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   GMS     - Standard state Gibbs free energies in mass units
             for the species:
                  cgs units - ergs/gm
                  Data type - real array
                  Dimension GMS(*) at least KK, the total number of species.
```

```
CKHBML     CKHBML     CKHBML     CKHBML     CKHBML     CKHBML     CKHBML
          ******************************************
                    ************************
                              ******

SUBROUTINE CKHBML (T, X, ICKWRK, RCKWRK, HBML)
   Returns the mean enthalpy of the mixture in molar units; see Eq. (37).

INPUT
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   X       - Mole fractions of the species
                   cgs units - none
                   Data type - real array
                   Dimension X(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   HBML    - Mean enthalpy in molar units:
                   cgs units - ergs/mole
                   Data type - real scalar.


CKHBMS     CKHBMS     CKHBMS     CKHBMS     CKHBMS     CKHBMS     CKHBMS
          ******************************************
                    ************************
                              ******

SUBROUTINE CKHBMS (T, Y, ICKWRK, RCKWRK, HBMS)
   Returns the mean enthalpy of the mixture in mass units; see Eq. (38).

INPUT
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   Y       - Mass fractions of the species
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   HBMS    - Mean enthalpy in mass units:
                   cgs units - ergs/gm
                   Data type - real scalar.
```

```
CKHML      CKHML      CKHML      CKHML      CKHML      CKHML      CKHML
           ****************************************************
                      ***************************
                                ******

SUBROUTINE CKHML (T, ICKWRK, RCKWRK, HML)
   Returns the enthalpies in molar units

INPUT
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   HML     - Enthalpies in molar units for the species
                   cgs units - ergs/mole
                   Data type - real array
                   Dimension HML(*) at least KK, the total number of species.


CKHMS      CKHMS      CKHMS      CKHMS      CKHMS      CKHMS      CKHMS
           ****************************************************
                      ***************************
                                ******

SUBROUTINE CKHMS (T, ICKWRK, RCKWRK, HMS)
   Returns the enthalpies in mass units; see Eq. (27).

INPUT
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   HMS     - Enthalpies in mass units for the species.
                   cgs units - ergs/gm
                   Data type - real array
                   Dimension HMS(*) at least KK, the total number of species.


CKHORT     CKHORT     CKHORT     CKHORT     CKHORT     CKHORT     CKHORT
           ****************************************************
                      ***************************
                                ******

SUBROUTINE CKHORT (T, ICKWRK, RCKWRK, HORT)
   Returns the nondimensional enthalpies; see Eq. (20).

INPUT
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   HORT    - Nondimensional enthalpies for the species
                   cgs units - none
                   Data type - real array
                   Dimension HORT(*) at least KK, the total number of species.
```

```
CKI2CH    CKI2CH    CKI2CH    CKI2CH    CKI2CH    CKI2CH    CKI2CH
          ********************************************
                    **************************
                              ******

SUBROUTINE CKI2CH (NUM, STR, I, KERR)
   Returns a character string representation of an integer
   and the effective length of the string.

INPUT
   NUM    - A number to be converted to a character string; the maximum
            magnitude of NUM is machine dependent:
                 Data type - integer scalar.
OUTPUT
   STR    - A left-justified character string representing NUM:
                 Data type - integer scalar.
   I      - The effective length of the character string:
                 Data type - integer scalar.
  KERR    - Error flag; character length errors will result in KERR=.TRUE.
                 Data type - logical.


CKINDX    CKINDX    CKINDX    CKINDX    CKINDX    CKINDX    CKINDX
          ********************************************
                    **************************
                              ******

SUBROUTINE CKINDX (ICKWRK, RCKWRK, MM, KK, II, NFIT)*
   Returns a group of indices defining the size of the particular
   reaction mechanism.

INPUT
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   MM     - Total number of elements in mechanism.
                 Data type - integer scalar
   KK     - Total number of species in mechanism.
                 Data type - integer scalar
   II     - Total number of reactions in mechanism.
                 Data type - integer scalar
   NFIT   - number of coefficients in fits to thermodynamic data
            for one temperature range; NFIT = number of
            coefficients in polynomial fits to CP/R  +  2.
                 Data type - integer scalar
```

```
CKINIT    CKINIT    CKINIT    CKINIT    CKINIT    CKINIT    CKINIT
          ****************************************
                    ************************
                              ******

SUBROUTINE CKINIT (LENIWK, LENRWK, LENCWK, LINC, LOUT, ICKWRK,
                   RCKWRK, CCKWRK)*
   Reads the linking file and creates the internal work arrays ICKWRK,
   CCKWRK, and RCKWRK. CKINIT must be called before any other CHEMKIN
   subroutine is called. The work arrays must then be made available
   as input to the other CHEMKIN subroutines.

INPUT
   LENIWK - Length of the integer work array, ICKWRK:
                 Data type - integer scalar
   LENCWK - Length of the character work array, CCKWRK
            The minimum length of CCKWRK(*) is MM + KK:
                 Data type - integer scalar
   LENRWK - Length of the real work array, WORK:
                 Data type - integer scalar
   LINC   -  Logical file number for the linking file:
                 Data type - integer scalar
   LOUT   -  Output file for printed error messages:
                 Data type - integer scalar
OUTPUT
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
   CCKWRK - Array of character work space.
                 Data type - CHARACTER*16 array
                 Dimension CCKWRK(*) at least LENCWK.


CKITR     CKITR     CKITR     CKITR     CKITR     CKITR     CKITR
          ********************************************
                    ************************
                              ******

SUBROUTINE CKITR (ICKWRK, RCKWRK, ITHB, IREV)
   Returns a set of flags indicating whether the reactions are
   reversible or whether they contain arbitrary third bodies.

INPUT
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   ITHB   - Third-body flags for the reactions;
            ITHB(I)= -1  reaction I is not a third-body reactions
            ITHB(I)=  0  reaction I is is a third-body reaction with
                         no enhanced third body efficiencies
            ITHB(I)=  N  reaction I is a third-body reaction with
                         N species enhanced third-body efficiencies.
                 Data type - integer array
                 Dimension ITHB(*) at least II, the total number of
                 reactions.
   IREV   - Reversibility flags and number of species
           (reactants plus products) for reactions.
            IREV(I)=+N, reversible reaction I has N species
            IREV(I)=-N, irreversible reaction I has N species
                 Data type - integer array
                 Dimension IREV(*) at least II, the total number of
                 reactions.
```

```
CKMMWC     CKMMWC     CKMMWC     CKMMWC     CKMMWC     CKMMWC     CKMMWC
           ******************************************
                      **************************
                                 ******

SUBROUTINE CKMMWC (C, ICKWRK, RCKWRK, WTM)
   Returns the mean molecular weight of the gas mixture given the
   molar concentrations; see Eq. (5).

INPUT
   C       - Molar concentrations of the species.
                   cgs units - mole/cm**3
                   Data type - real array
                   Dimension C(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   WTM     - Mean molecular weight of the species mixture.
                   cgs units - gm/mole
                   Data type - real scalar


CKMMWX     CKMMWX     CKMMWX     CKMMWX     CKMMWX     CKMMWX     CKMMWX
           ******************************************
                      **************************
                                 ******

SUBROUTINE CKMMWX (X, ICKWRK, RCKWRK, WTM)
   Returns the mean molecular weight of the gas mixture given the
   mole fractions; see Eq. (4).

INPUT
   X       - Mole fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension X(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   WTM     - Mean molecular weight of the species mixture.
                   cgs units - gm/mole
                   Data type - real scalar
```

```
CKMMWY     CKMMWY     CKMMWY     CKMMWY     CKMMWY     CKMMWY     CKMMWY
          ******************************************
                    **************************
                              ******

SUBROUTINE CKMMWY (Y, ICKWRK, RCKWRK, WTM)
   Returns the mean molecular weight of the gas mixture given the
   mass fractions; see Eq. (3).

INPUT
   Y       - Mass fractions of the species.
             cgs units - none
             Data type - real array
             Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
             Data type - integer array
             Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
             Data type - real array
             Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   WTM     - Mean molecular weight of the species mixture.
             cgs units - gm/mole
             Data type - real scalar


CKNCF      CKNCF      CKNCF      CKNCF      CKNCF      CKNCF      CKNCF
          ******************************************
                    **************************
                              ******

SUBROUTINE CKNCF (MDIM, ICKWRK, RCKWRK, NCF)
   Returns the elemental composition of the species.

INPUT
   MDIM    - First dimension of the two-dimensional array NCF;
             MDIM must be equal to or greater than the number of
             elements, MM.
             Data type - integer scalar
   ICKWRK  - Array of integer workspace
             Data type - integer array
             Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
             Data type - real array
             Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   NCF     - Matrix of the elemental composition of the species;
             NCF(M,K) is the number of atoms of the Mth element
             in the Kth species
             Data type - integer array
             Dimension NCF(MDIM,*) exactly MDIM (at least MM,
             the total number of elements in the problem) for
             the first dimension and at least KK, the total
             number of species, for the second.
```

```
CKNPAR     CKNPAR     CKNPAR     CKNPAR     CKNPAR     CKNPAR     CKNPAR
           ******************************************
                      ************************
                                ******
```

SUBROUTINE CKNPAR (LINE, NPAR, LOUT, IPAR, ISTART, KERR)

This subroutine is called to parse a character string, LINE, that is composed of several blank-delimited substrings. That final segment of LINE containing NPAR substrings is found, beginning in the ISTART column; this segment is then copied into the character string IPAR. This allows format-free input of combined alpha-numeric data. For example, after reading a line containing alpha-numeric information ending with several numbers, the subroutine might be called to find the segment of the line containing the numbers:

```
input:  LINE   = "t1, t2, dt  300.0  3.0E3  50"
        NPAR   = 3, the number of substrings requested
        LOUT   = 6, a logical unit number on which to write diagnostic
                 messages.
output: IPAR   = "300.0  3.0E3  50"
        ISTART = 13, the starting column in LINE of the
                 NPAR substrings
          KERR = .FALSE.
```

INPUT

```
   LINE   - A character string
                 Data type - CHARACTER*(*)
   NPAR   - Number of substrings expected
                 Data type - integer scalar
   LOUT   - Output unit for printed diagnostics
                 Data type - integer scalar
```

OUTPUT

```
   IPAR   - A character string containing only the NPAR substrings.
   ISTART - The starting location in LINE of the NPAR substrings.
     KERR - Error flag; an error in syntax or character length will result
            in KERR = .TRUE.
                 Data type - logical.
```

```
CKNU       CKNU       CKNU       CKNU       CKNU       CKNU       CKNU
           ******************************************
                      ************************
                                ******
```

SUBROUTINE CKNU (KDIM, ICKWRK, RCKWRK, NUKI)

Returns the stoichiometric coefficients of the reaction mechanism; see Eq. (50).

INPUT

```
   KDIM   - First dimension of the two-dimensional array NUKI; KDIM must
            be greater than or equal to the total number of species, KK.
                 Data type - integer scalar
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
```

OUTPUT

```
   NUKI   - Matrix of stoichiometric coefficients for the species in the
            reactions; NUKI(K,I) is the stoichiometric coefficient of
            species K in reaction I.
                 Data type - integer array
                 Dimension NUKI(KDIM,*) exactly KDIM (at least KK, the
                 total number of species) for the first dimension and at
                 least II for the second, the total number of reactions.
```

```
CKPC      CKPC      CKPC      CKPC      CKPC      CKPC      CKPC
          ****************************************
                    **************************
                              ******

SUBROUTINE CKPC (RHO, T, C, ICKWRK, RCKWRK, P)
   Returns the pressure of the gas mixture given the mass density,
   temperature and molar concentrations; see Eq. (2).

INPUT
   RHO    - Mass density.
            cgs units - gm/cm**3
            Data type - real scalar
   T      - Temperature.
            cgs units - kelvins
            Data type - real scalar
   C      - Molar concentrations of the species
            cgs units - mole/cm**3
            Data type - real array
            Dimension C(*) at least KK, the total number of species.
   ICKWRK - Array of integer workspace
            Data type - integer array
            Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
            Data type - real array
            Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   P      - Pressure.
            cgs units - dynes/cm**2
            Data type - real scalar


CKPHAZ    CKPHAZ    CKPHAZ    CKPHAZ    CKPHAZ    CKPHAZ    CKPHAZ
          ****************************************
                    **************************
                              ******

SUBROUTINE CKPHAZ (ICKWRK, RCKWRK, KPHASE)
   Returns a set of flags indicating phases of the species.

INPUT
   ICKWRK - Array of integer workspace
            Data type - integer array
            Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
            Data type - real array
            Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   KPHASE - Phases of the species;
            KPHASE(K)=-1  the Kth species is solid
            KPHASE(K)= 0  the Kth species is gaseous
            KPHASE(K)=+1  the Kth species is liquid
            Data type - integer array
            Dimension KPHASE(*) at least KK, the total number of
            species.
```

```
CKPX       CKPX       CKPX       CKPX       CKPX       CKPX       CKPX
           ***********************************************
                      ***************************
                                 ******.

SUBROUTINE CKPX (RHO, T, X, ICKWRK, RCKWRK, P)
   Returns the pressure of the gas mixture given the mass density,
   temperature and mole fractions; see Eq. (1).

INPUT
   RHO     - Mass density.
                   cgs units - gm/cm**3
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   X       - Mole fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension X(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   P       - Pressure.
                   cgs units - dynes/cm**2
                   Data type - real scalar


CKPY       CKPY       CKPY       CKPY       CKPY       CKPY       CKPY
           ***********************************************
                      ***************************
                                 ******

SUBROUTINE CKPY (RHO, T, Y, ICKWRK, RCKWRK, P)
   Returns the pressure of the gas mixture given the mass density,
   temperature and mass fractions; see Eq. (1).

INPUT
   RHO     - Mass density.
                   cgs units - gm/cm**3
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   Y       - Mass fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   P       - Pressure.
                   cgs units - dynes/cm**2
                   Data type - real scalar
```

```
CKQC       CKQC       CKQC       CKQC       CKQC       CKQC       CKQC
           ******************************************
                      ***************************
                                 ******

SUBROUTINE CKQC (T, C, ICKWRK, RCKWRK, Q)
   Returns the rates of progress for the reactions given
   temperature and molar concentrations; see Eqs. (51) and (58).

INPUT
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   C       - Molar concentrations of the species.
                   cgs units - mole/cm**3
                   Data type - real array
                   Dimension C(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   Q       - Rates of progress for the reactions.
                   cgs units - moles/(cm**3*sec)
                   Data type - real array
                   Dimension Q(*) at least II, the total number of reactions.


CKQXP      CKQXP      CKQXP      CKQXP      CKQXP      CKQXP      CKQXP
           **********************************************
                      ***************************
                                 ******

SUBROUTINE CKQXP (P, T, X, ICKWRK, RCKWRK, Q)
   Returns the rates of progress for the reactions given pressure,
   temperature and mole fractions; see Eqs. (51) and (58).

INPUT
   P       - Pressure.
                   cgs units - dynes/cm**2
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   X       - Mole fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension X(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   Q       - Rates of progress for the reactions.
                   cgs units - moles/(cm**3*sec)
                   Data type - real array
                   Dimension Q(*) at least II, the total number of reactions.
```

```
CKQXR     CKQXR     CKQXR     CKQXR     CKQXR     CKQXR     CKQXR
          ********************************************
                    **************************
                          ******

SUBROUTINE CKQXR (RHO, T, X, ICKWRK, RCKWRK, Q)
   Returns the rates of progress for the reactions given mass
   density, temperature and mole fractions; see Eqs. (51) and (58).

INPUT
   RHO    - Mass density.
            cgs units - gm/cm**3
            Data type - real scalar
   T      - Temperature.
            cgs units - kelvins
            Data type - real scalar
   X      - Mole fractions of the species.
            cgs units - none
            Data type - real array
            Dimension X(*) at least KK, the total number of species.
   ICKWRK - Array of integer workspace
            Data type - integer array
            Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
            Data type - real array
            Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   Q      - Rates of progress for the reactions.
            cgs units - moles/(cm**3*sec)
            Data type - real array
            Dimension Q(*) at least II, the total number of reactions.


CKQYP     CKQYP     CKQYP     CKQYP     CKQYP     CKQYP     CKQYP
          ********************************************
                    **************************
                          ******

SUBROUTINE CKQYP (P, T, Y, ICKWRK, RCKWRK, Q)
   Returns the rates of progress for the reactions given pressure,
   temperature and mass fractions; see Eqs. (51) and (58).

INPUT
   P      - Pressure.
            cgs units - dynes/cm**2
            Data type - real scalar
   T      - Temperature.
            cgs units - kelvins
            Data type - real scalar
   Y      - Mass fractions of the species.
            cgs units - none
            Data type - real array
            Dimension Y(*) at least KK, the total number of species.
   ICKWRK - Array of integer workspace
            Data type - integer array
            Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
            Data type - real array
            Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   Q      - Rates of progress for the reactions.
            cgs units - moles/(cm**3*sec)
            Data type - real array
            Dimension Q(*) at least II, the total number of reactions.
```

```
CKQYR      CKQYR      CKQYR      CKQYR      CKQYR      CKQYR      CKQYR
           ******************************************
                     ************************
                           ******

SUBROUTINE CKQYR (RHO, T, Y, ICKWRK, RCKWRK, Q)
   Returns the rates of progress for the reactions given mass
   density, temperature and mass fractions; see Eqs. (51) and (58).

INPUT
   RHO     - Mass density.
                 cgs units - gm/cm**3
                 Data type - real scalar
   T       - Temperature.
                 cgs units - kelvins
                 Data type - real scalar
   Y       - Mass fractions of the species.
                 cgs units - none
                 Data type - real array
                 Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   Q       - Rates of progress for the reactions.
                 cgs units - moles/(cm**3*sec)
                 Data type - real array
                 Dimension Q(*) at least II, the total number of reactions.


CKR2CH     CKR2CH     CKR2CH     CKR2CH     CKR2CH     CKR2CH     CKR2CH
           ******************************************
                     ************************
                           ******

SUBROUTINE CKR2CH (RNUM, STR, I, KERR)
   Returns a character string representation of a real number
   and the effective length of the string.

INPUT
   RNUM    - The number to be converted to a string; the maximum magnitude
             is machine dependent:
                 Data type - real scalar.
OUTPUT
   STR     - A left-justified character string representing RNUM, with five
             to ten characters, depending on the input value, e.g.,
             RNUM = 0.0 returns STR = " 0.00"
             RNUM = -10.5 returns STR = "-1.05E+01"
             RNUM = 1.86E-100 returns STR = "  1.86E-100"
                 Data type - CHARACTER*(*)
                 The minimum length of STR is 5.
   I       - The effective length of STR.
                 Data type - integer scalar.
   KERR    - Error flag; a character-length error will result in KERR=.TRUE.
                 Data type - logical.
```

```
CKRAEX    CKRAEX    CKRAEX    CKRAEX    CKRAEX    CKRAEX    CKRAEX
          ****************************************
                    ************************
                              ******

SUBROUTINE CKRAEX (I, RCKWRK, RA)
   Get/put the pre-exponential coefficient of the Ith reaction.

INPUT
   I       - Reaction number; I > 0 gets RA(I) from RCKWRK
                              I < 0 puts RA(I) into RCKWRK
                   Data type - integer scalar
   RCKWRK - Array of real work space
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
   If I < 1, then
   RA      - Pre-exponential coefficient for the Ith reaction
                   cgs units - mole-cm-sec-K
                   Data type - real scalar
OUTPUT
   If I > 1, then
   RA      - Pre-exponential coefficient for Ith reaction
                   cgs units - mole-cm-sec-K
                   Data type - real scalar.


CKRHOC    CKRHOC    CKRHOC    CKRHOC    CKRHOC    CKRHOC    CKRHOC
          ****************************************
                    ************************
                              ******

SUBROUTINE CKRHOC (P, T, C, ICKWRK, RCKWRK, RHO)
   Returns the mass density of the gas mixture given the pressure,
   temperature and molar concentrations; see Eq. (2).

INPUT
   P       - Pressure.
                   cgs units - dynes/cm**2
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   C       - Molar concentrations of the species.
                   cgs units - mole/cm**3
                   Data type - real array
                   Dimension C(*) at least KK, the total number of species.
   ICKWRK - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   RHO     - Mass density.
                   cgs units - gm/cm**3
                   Data type - real scalar
```

```
CKRHOX    CKRHOX    CKRHOX    CKRHOX    CKRHOX    CKRHOX    CKRHOX
          ****************************************
                    **************************
                              ******

SUBROUTINE CKRHOX (P, T, X, ICKWRK, RCKWRK, RHO)
   Returns the mass density of the gas mixture given the pressure,
   temperature and mole fractions; see Eq. (2).

INPUT
   P       - Pressure.
                   cgs units - dynes/cm**2
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   X       - Mole fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension X(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   RHO     - Mass density.
                   cgs units - gm/cm**3
                   Data type - real scalar


CKRHOY    CKRHOY    CKRHOY    CKRHOY    CKRHOY    CKRHOY    CKRHOY
          ****************************************
                    **************************
                              ******

SUBROUTINE CKRHOY (P, T, Y, ICKWRK, RCKWRK, RHO)
   Returns the mass density of the gas mixture given the pressure,
   temperature and mass fractions; see Eq. (2).

INPUT
   P       - Pressure.
                   cgs units - dynes/cm**2
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   Y       - Mass fractions of the species.
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   RHO     - Mass density.
                   cgs units - gm/cm**3
                   Data type - real scalar
```

```
CKRP      CKRP      CKRP      CKRP      CKRP      CKRP      CKRP
          ****************************************
                    ************************
                              ******

SUBROUTINE CKRP (ICKWRK, RCKWRK, RU, RUC, PA)
   Returns universal gas constants and the pressure of one standard
   atmosphere.

INPUT
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   RU     - Universal gas constant.
                 cgs units - 8.314E7 ergs/(mole*K)
                 Data type - real scalar
   RUC    - Universal gas constant used only in conjuction with
            activation energy
                 preferred units - 1.987 cal/(mole*K)
                 Data type - real scalar
   PA     - Pressure of one standard atmosphere.
                 cgs units - 1.01325E6 dynes/cm**2
                 Data type - real scalar


CKSBML    CKSBML    CKSBML    CKSBML    CKSBML    CKSBML    CKSBML
          ****************************************
                    ************************
                              ******

SUBROUTINE CKSBML (P, T, X, ICKWRK, RCKWRK, SBML)*
   Returns the mean entropy of the mixture in molar units, given the
   pressure, temperature and mole fractions; see Eq. (42).

INPUT
   P      - Pressure.
                 cgs units - dynes/cm**2
                 Data type - real scalar
   T      - Temperature.
                 cgs units - kelvins
                 Data type - real scalar
   X      - Mole fractions of the species.
                 cgs units - none
                 Data type - real array
                 Dimension X(*) at least KK, the total number of species.
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   SBML   - Mean entropy in molar units.
                 cgs units - ergs/(mole*K)
                 Data type - real scalar
```

```
CKSBMS     CKSBMS     CKSBMS     CKSBMS     CKSBMS     CKSBMS     CKSBMS
          ******************************************
                    ************************
                              ******

SUBROUTINE CKSBMS (P, T, Y, ICKWRK, RCKWRK, SBMS)*
   Returns the mean entropy of the mixture in mass units, given the
   pressure, temperature and mass fractions; see Eq.(43).

INPUT
   P       - Pressure
                   cgs units - dynes/cm**2
                   Data type - real scalar
   T       - Temperature.
                   cgs units - kelvins
                   Data type - real scalar
   Y       - Mass fractions of the species
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   SBMS    - Mean entropy in mass units
                   cgs units - ergs/(gm*K)
                   Data type - real scalar


   CKSML      CKSML      CKSML      CKSML      CKSML      CKSML      CKSML
          ******************************************
                    ************************
                              ******

SUBROUTINE CKSML (T, ICKWRK, RCKWRK, SML)
   Returns the standard state entropies in molar units.

INPUT
   T       - Temperature
                   cgs units - kelvins
                   Data type - real scalar
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   SML     - Standard state entropies in molar units for the species.
                   cgs units - ergs/(mole*K)
                   Data type - real array
                   Dimension SML(*) at least KK, the total number of species.
```

```
CKSMS      CKSMS      CKSMS      CKSMS      CKSMS      CKSMS      CKSMS
           ****************************************
                      ************************
                            ******
```

```
SUBROUTINE CKSMS (T, ICKWRK, RCKWRK, SMS)
   Returns the standard state entropies in mass units; see Eq. (28).

INPUT
   T      - Temperature
                 cgs units - kelvins
                 Data type - real scalar
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   SMS    - Standard state entropies in mass units for the species.
                 cgs units - ergs/(gm*K)
                 Data type - real array
                 Dimension SMS(*) at least KK, the total number of species.
```

```
CKSNUM     CKSNUM     CKSNUM     CKSNUM     CKSNUM     CKSNUM     CKSNUM
           ****************************************
                      ************************
                            ******
```

```
SUBROUTINE CKSNUM (LINE,NEXP,LOUT,KRAY,NN,KNUM,NVAL,RVAL,KERR)
   This subroutine is called to parse a character string, LINE, that is
   composed of several blank-delimited substrings. It is expected that the
   first substring in LINE is also an entry in a reference array of character
   strings, KRAY(*), in which case the index position in KRAY(*) is returned
   as KNUM; otherwise an error flag is returned. The substrings following the
   first are expected to represent numbers and are converted to elements of
   the array RVAL(*). If NEXP substrings are not found, an error flag is
   returned. This allows format-free input of combined alpha-numeric data.
   For example, after reading a line containing a species name followed by
   several numerical values, the subroutine might be called to find a Chemkin
   species index and convert the other substrings to real values:

   input:  LINE     = "N2  1.2"
           NEXP     = 1, the number of values expected
           LOUT     = 6, a logical unit number on which to write
                      diagnostic messages
           KRAY(*)  = "H2" "O2" "N2" "H" "O" "N" "OH" "H2O" "NO"
           NN       = 9, the number of entries in KRAY(*)
   output: KNUM     = 3, the index number of the substring in KRAY(*),
                      which corresponds to the first substring in LINE
           NVAL     = 1, the number of values found in LINE
                      following the first substring
           RVAL(*)  = 1.200E+00, the substring converted to a number
           KERR     = .FALSE.

INPUT
   LINE   - A character string
                 Data type - CHARACTER*80
   NEXP   - Number of real values to be found in character string
                 Data type - integer scalar
   LOUT   - Output unit for error messages.
                 Data type - integer scalar
   KRAY   - Array of character strings
                 Data type - CHARACTER*(*)
   NN     - Total number of character strings in KRAY
                 Data type - integer scalar
OUTPUT
   KNUM   - Index number of character string in array which
            corresponds to the first substring in LINE
                 Data type - integer scalar
   NVAL   - Number of real values found in LINE
                 Data type - integer scalar
   RVAL   - Array of real values found in LINE
                 Data type - real array
   KERR   - Error flag; KERR=.TRUE. if there is a syntax or dimensioning
            error, the corresponding string is not found, or the total of
            values found is not the number of values expected.
                 Data type - logical.
```

```
CKSOR     CKSOR     CKSOR     CKSOR     CKSOR     CKSOR     CKSOR
          ******************************************
                    ************************
                              ******

SUBROUTINE CKSOR (T, ICKWRK, RCKWRK, SOR)
   Returns the nondimensional entropies; see Eq. (21).

INPUT
   T       - Temperature
                   cgs units - kelvins
                   Data type - real scalar
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   SOR     - Nondimensional entropies for the species.
                   cgs units - none
                   Data type - real array
                   Dimension SOR(*) at least KK, the total number of species.


CKSUBS    CKSUBS    CKSUBS    CKSUBS    CKSUBS    CKSUBS    CKSUBS
          ******************************************
                    ************************
                              ******

SUBROUTINE CKSUBS (LINE, LOUT, NDIM, SUB, NFOUND, KERR)
   Returns an array of substrings in a character string with blanks
   as the delimiter.

INPUT
   LINE   - A character string
                   Data type - CHARACTER*(*)
   LOUT   - Output unit for printed diagnostics.
   NDIM   - Dimension of array SUB(*)*(*)

OUTPUT
   SUB    - An array of the character substrings of LINE
                   Data type - CHARACTER*(*) array
                   Dimension of SUB(*) at least NDIM.
   NFOUND - Number of substrings found in LINE
                   Data type - integer
   KERR   - Error flag; KERR=.TRUE. if there are dimensioning errors
                   Data type - logical.


CKSYME    CKSYME    CKSYME    CKSYME    CKSYME    CKSYME    CKSYME
          ******************************************
                    ************************
                              ******

SUBROUTINE CKSYME (CCKWRK, LOUT, ENAME, KERR)*
   Returns the character strings of element names.

INPUT
   CCKWRK - Array of character work space
                   Data type - character array
                   Dimension CCKWRK(*) at least LENCWK.
   LOUT   - Output unit for printed diagnostics
                   Data type - integer scalar

OUTPUT
   ENAME  - Element names
                   Data type - CHARACTER*(*)* array
                   Dimension ENAME at least MM, the total number of
                   elements in the problem.
   KERR   - Error flag; KERR=.TRUE. if there is a character length error
                   Data type - logical.
```

```
CKSYMR     CKSYMR     CKSYMR     CKSYMR     CKSYMR     CKSYMR     CKSYMR
           ***********************************************
                      ***************************
                                 ******

SUBROUTINE CKSYMR (I, ICKWRK, RCKWRK, CCKWRK, LT, ISTR, KERR)*
   Returns a character string which describes the Ith reaction,
   and the effective length of the character string.

INPUT
   I       - Reaction index.
                   Data type - integer scalar
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
   CCKWRK  - Array of character work space
                   Data type - CHARACTER*16 array
                   Dimension CCKWRK(*) at least LENCWK.
OUTPUT
   ISTR    - Character string describing the Ith reaction
                   Data type - CHARACTER*(*)
   LT      - Number of characters in the reaction description.
                   Data type - integer scalar
   KERR    - Error flag; KERR=.TRUE. if there is a character-length error
                   Data type - logical.


CKSYMS     CKSYMS     CKSYMS     CKSYMS     CKSYMS     CKSYMS     CKSYMS
           ***********************************************
                      ***************************
                                 ******

SUBROUTINE CKSYMS (CCKWRK, LOUT, KNAME, KERR)*
   Returns the character strings of species names.

INPUT
   CCKWRK  - Array of character work space
                   Data type - CHARACTER*16 array
                   Dimension CCKWRK(*) at least LENCWK.
OUTPUT
   KNAME   - Species names
                   Data type - CHARACTER(*) array
                   Dimension KNAME(*) at least KK, the total number of species.
   KERR    - Error flag; KERR=.TRUE. if there is a character-length error
                   Data type - logical.
```

```
CKTHB      CKTHB'     CKTHB      CKTHB      CKTHB      CKTHB      CKTHB
          ******************************************
                    ***************************
                              ******

SUBROUTINE CKTHB (KDIM, ICKWRK, RCKWRK, AKI)
   Returns matrix of enhanced third body coefficients; see Eq. (58).

INPUT
   KDIM   - First dimension of the two dimensional array AKI;
            KDIM must be greater than or equal to the total
            number of species, KK
                 Data type - integer scalar
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   AKI    - Matrix of enhanced third body efficiencies of the
            species in the reactions; AKI(K,I) is the enhanced
            efficiency of the Kth species in the Ith reaction
                 Data type - real array
                 Dimension AKI(KDIM,*) exactly KDIM (at least KK,
                 the total number of species) for the first
                 dimension and at least II for the second, the total
                 number of reactions.


CKUBML     CKUBML     CKUBML     CKUBML     CKUBML     CKUBML     CKUBML
          ******************************************
                    ***************************
                              ******

SUBROUTINE CKUBML (T, X, ICKWRK, RCKWRK, UBML)
   Returns the mean internal energy of the mixture in molar units;
   see Eq. (39).

INPUT
   T      - Temperature
                 cgs units - kelvins
                 Data type - real scalar
   X      - Mole fractions of the species
                 cgs units - none
                 Data type - real array
                 Dimension X(*) at least KK, the total number of species.
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   UBML   - Mean internal energy in molar units:
                 cgs units - ergs/mole
                 Data type - real scalar.
```

```
CKUBMS     CKUBMS     CKUBMS     CKUBMS     CKUBMS     CKUBMS     CKUBMS
           ******************************************
                     ************************
                           ******

SUBROUTINE CKUBMS (T, Y, ICKWRK, RCKWRK, UBMS)
   Returns the mean internal energy of the mixture in mass units;
   see Eq. (40).

INPUT
   T       - Temperature
                   cgs units - kelvins
                   Data type - real scalar
   Y       - Mass fractions of the species
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   UBMS    - Mean internal energy in mass units:
                   cgs units - ergs/gm
                   Data type - real scalar.


CKUML      CKUML      CKUML      CKUML      CKUML      CKUML      CKUML
           ******************************************
                     ************************
                           ******

SUBROUTINE CKUML (T, ICKWRK, RCKWRK, UML)
   Returns the internal energies in molar units; see Eq. (23).

INPUT
   T       - Temperature
                   cgs units - kelvins
                   Data type - real scalar
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   UML     - Internal energies in molar units for the species.
                   cgs units - ergs/mole
                   Data type - real array
                   Dimension UML(*) at least KK, the total number of species.
```

```
CKUMS      CKUMS      CKUMS      CKUMS      CKUMS      CKUMS      CKUMS
           ******************************************
                      ************************
                                 ******

SUBROUTINE CKUMS (T, ICKWRK, RCKWRK, UMS)
   Returns the internal energies in mass units; see Eq. (30).

INPUT
   T       - Temperature
                   cgs units - kelvins
                   Data type - real scalar
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   UMS     - Internal energies in mass units for the species.
                   cgs units - ergs/gm
                   Data type - real array
                   Dimension UMS(*) at least KK, the total number of species.


CKWC       CKWC       CKWC       CKWC       CKWC       CKWC       CKWC
           ******************************************
                      ************************
                                 ******

SUBROUTINE CKWC (T, C, ICKWRK, RCKWRK, WDOT)
   Returns the molar production rates of the species given the
   temperature and molar concentrations; see Eq. (49).

INPUT
   T       - Temperature
                   cgs units - kelvins
                   Data type - real scalar
   C       - Molar concentrations of the species
                   cgs units - mole/cm**3
                   Data type - real array
                   Dimension C(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   WDOT    - Chemical molar production rates of the species.
                   cgs units - moles/(cm**3*sec)
                   Data type - real array
                   Dimension WDOT(*) at least KK, the total number of species.
```

```
CKWL       CKWL       CKWL       CKWL       CKWL       CKWL       CKWL
           ******************************************
                     ************************
                           ******
```

```
   SUBROUTINE CKWL (ICKWRK, RCKWRK, WL)
      Returns a set of flags providing information on the wave length
      of photon radiation.

   INPUT
      ICKWRK - Array of integer workspace
                     Data type - integer array
                     Dimension ICKWRK(*) at least LENIWK.
      RCKWRK - Array of real work space.
                     Data type - real array
                     Dimension RCKWRK(*) at least LENRWK.
   OUTPUT
      WL     - Radiation wavelengths for the reactions.
               WL(I)= 0. reaction I does not have radiation as
                         either a reactant or product
               WL(I)=-A  reaction I has radiation of wavelength A
                         as a reactant
               WL(I)=+A  reaction I has radiation of wavelength A
                         as a product
               If A = 1.0 then no wavelength information was given;
                     cgs units - angstroms
                     Data type - real array
                     Dimension WL(*) at least II, the total number of reactions.
```

```
CKWT       CKWT       CKWT       CKWT       CKWT       CKWT       CKWT
           ******************************************
                     ************************
                           ******
```

```
   SUBROUTINE CKWT (ICKWRK, RCKWRK, WT)
      Returns the molecular weights of the species.

   INPUT
      ICKWRK - Array of integer workspace
                     Data type - integer array
                     Dimension ICKWRK(*) at least LENIWK.
      RCKWRK - Array of real work space.
                     Data type - real array
                     Dimension RCKWRK(*) at least LENRWK.
   OUTPUT
      WT     - Molecular weights of the species.
                     cgs units - gm/mole
                     Data type - real array
                     Dimension WT(*) at least KK, the total number of species.
```

```
CKWXP      CKWXP      CKWXP      CKWXP      CKWXP      CKWXP      CKWXP
           ****************************************
                      ************************
                                 ******

SUBROUTINE CKWXP (P, T, X, ICKWRK, RCKWRK, WDOT)
   Returns the molar production rates of the species given the
   pressure, temperature and mole fractions; see Eq. (49).

INPUT
   P       - Pressure.
                   cgs units - dynes/cm**2
                   Data type - real scalar
   T       - Temperature
                   cgs units - kelvins
                   Data type - real scalar
   X       - Mole fractions of the species
                   cgs units - none
                   Data type - real array
                   Dimension X(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   WDOT    - Chemical molar production rates of the species.
                   cgs units - moles/(cm**3*sec)
                   Data type - real array
                   Dimension WDOT(*) at least KK, the total number of species.


CKWXR      CKWXR      CKWXR      CKWXR      CKWXR      CKWXR      CKWXR
           ****************************************
                      ************************
                                 ******

SUBROUTINE CKWXR (RHO, T, X, ICKWRK, RCKWRK, WDOT)
   Returns the molar production rates of the species given the
   mass density, temperature and mole fractions; see Eq. (49).

INPUT
   RHO     - Mass density
                   cgs units - gm/cm**3
                   Data type - real scalar
   T       - Temperature
                   cgs units - kelvins
                   Data type - real scalar
   X       - Mole fractions of the species
                   cgs units - none
                   Data type - real array
                   Dimension X(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   WDOT    - Chemical molar production rates of the species.
                   cgs units - moles/(cm**3*sec)
                   Data type - real array
                   Dimension WDOT(*) at least KK, the total number of species.
```

```
CKWYP      CKWYP      CKWYP      CKWYP      CKWYP      CKWYP      CKWYP
           ******************************************
                      ************************
                                 ******

SUBROUTINE CKWYP (P, T, Y, ICKWRK, RCKWRK, WDOT)
   Returns the molar production rates of the species given the
   pressure, temperature and mass fractions; see Eq. (49).

INPUT
   P       - Pressure.
                   cgs units - dynes/cm**2
                   Data type - real scalar
   T       - Temperature
                   cgs units - kelvins
                   Data type - real scalar
   Y       - Mass fractions of the species
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   WDOT    - Chemical molar production rates of the species.
                   cgs units - moles/(cm**3*sec)
                   Data type - real array
                   Dimension WDOT(*) at least KK, the total number of species.


CKWYR      CKWYR      CKWYR      CKWYR      CKWYR      CKWYR      CKWYR
           ******************************************
                      ************************
                                 ******

SUBROUTINE CKWYR (RHO, T, Y, ICKWRK, RCKWRK, WDOT)
   Returns the molar production rates of the species given the
   mass density, temperature and mass fractions; see Eq. (49).

INPUT
   RHO     - Mass density
                   cgs units - gm/cm**3
                   Data type - real scalar
   T       - Temperature
                   cgs units - kelvins
                   Data type - real scalar
   Y       - Mass fractions of the species
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   WDOT    - Chemical molar production rates of the species.
                   cgs units - moles/(cm**3*sec)
                   Data type - real array
                   Dimension WDOT(*) at least KK, the total number of species.
```

```
CKXNUM     CKXNUM     CKXNUM     CKXNUM     CKXNUM     CKXNUM     CKXNUM
          ******************************************
                    **************************
                              ******

SUBROUTINE CKXNUM (LINE, NEXP, LOUT, NVAL, RVAL, KERR)
   This subroutine is called to parse a character string, LINE, that is
   composed of several blank-delimited substrings. Each substring is expected
   to represent a number, which is converted to entries in the array of real
   numbers, RVAL(*). NEXP is the number of values expected, and NVAL is the
   number of values found. This allows format-free input of numerical data.
   For example:
   input:  LINE     = " 0.170E+14 0 47780.0"
           NEXP     = 3, the number of values requested
           LOUT     = 6, a logical unit number on which to write
                      diagnostic messages
   output: NVAL     = 3, the number of values found
           RVAL(*)  = 1.700E+13, 0.000E+00, 4.778E+04
           KERR     = .FALSE.
INPUT
   LINE    - A character string
                Data type - CHARACTER*80
   NEXP    - Number of real values to be found in character string
                Data type - integer scalar
   LOUT    - Output unit for error messages.
                Data type - integer scalar
OUTPUT
   NVAL    - Number of real values found in character string.
                Data type - integer scalar
   RVAL    - Array of real values found
                Data type - real array
   KERR    - Error flag; KERR=.TRUE. if there is a syntax of dimensioning
             error.
                Data type - logical.


CKXTCP     CKXTCP     CKXTCP     CKXTCP     CKXTCP     CKXTCP     CKXTCP
          ******************************************
                    **************************
                              ******

SUBROUTINE CKXTCP (P, T, X, ICKWRK, RCKWRK, C)
   Returns the molar concentrations given the pressure,
   temperature and mole fractions; see Eq. (10).

INPUT
   P       - Pressure.
                cgs units - dynes/cm**2
                Data type - real scalar
   T       - Temperature
                cgs units - kelvins
                Data type - real scalar
   X       - Mole fractions of the species
                cgs units - none
                Data type - real array
                Dimension X(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                Data type - integer array
                Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                Data type - real array
                Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   C       - Molar concentrations of the species
                cgs units - mole/cm**3
                Data type - real array
                Dimension C(*) at least KK, the total number of species.
```

```
CKXTCR    CKXTCR    CKXTCR    CKXTCR    CKXTCR    CKXTCR    CKXTCR
          ********************************************
                    *************************
                              ******

SUBROUTINE CKXTCR (RHO, T, X, ICKWRK, RCKWRK, C)
   Returns the molar concentrations given the mass density,
   temperature and mole fractions; see Eq. (11).

INPUT
   RHO    - Mass density
                 cgs units - gm/cm**3
                 Data type - real scalar
   T      - Temperature
                 cgs units - kelvins
                 Data type - real scalar
   X      - Mole fractions of the species
                 cgs units - none
                 Data type - real array
                 Dimension X(*) at least KK, the total number of species.
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   C      - Molar concentrations of the species
                 cgs units - mole/cm**3
                 Data type - real array
                 Dimension C(*) at least KK, the total number of species.


CKXTY     CKXTY     CKXTY     CKXTY     CKXTY     CKXTY
          ******************************************
                    *************************
                              ******

SUBROUTINE CKXTY (X, ICKWRK, RCKWRK, Y)
   Returns the mass fractions given the mole fractions; see Eq. (9).

INPUT
   X      - Mole fractions of the species
                 cgs units - none
                 Data type - real array
                 Dimension X(*) at least KK, the total number of species.
   ICKWRK - Array of integer workspace
                 Data type - integer array
                 Dimension ICKWRK(*) at least LENIWK.
   RCKWRK - Array of real work space.
                 Data type - real array
                 Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   Y      - Mass fractions of the species
                 cgs units - none
                 Data type - real array
                 Dimension Y(*) at least KK, the total number of species.
```

```
CKYTCP    CKYTCP    CKYTCP    CKYTCP    CKYTCP    CKYTCP    CKYTCP
          ****************************************
                    ************************
                              ******

SUBROUTINE CKYTCP (P, T, Y, ICKWRK, RCKWRK, C)
   Returns the molar concentrations given the pressure,
   temperature and mass fractions; see Eq. (7).

INPUT
   P       - Pressure.
                   cgs units - dynes/cm**2
                   Data type - real scalar
   T       - Temperature
                   cgs units - kelvins
                   Data type - real scalar
   Y       - Mass fractions of the species
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   C       - Molar concentrations of the species
                   cgs units - mole/cm**3
                   Data type - real array
                   Dimension C(*) at least KK, the total number of species.


CKYTCR    CKYTCR    CKYTCR    CKYTCR    CKYTCR    CKYTCR    CKYTCR
          ****************************************
                    ************************
                              ******

SUBROUTINE CKYTCR (RHO,T, Y, ICKWRK, RCKWRK, C)
   Returns the molar concentrations given the mass density,
   temperature and mass fractions; see Eq. (8).

INPUT
   RHO     - Mass density
                   cgs units - gm/cm**3
                   Data type - real scalar
   T       - Temperature
                   cgs units - kelvins
                   Data type - real scalar
   Y       - Mass fractions of the species
                   cgs units - none
                   Data type - real array
                   Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
                   Data type - integer array
                   Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
                   Data type - real array
                   Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   C       - Molar concentrations of the species
                   cgs units - mole/cm**3
                   Data type - real array
                   Dimension C(*) at least KK, the total number of species.
```

```
 CKYTX      CKYTX      CKYTX      CKYTX      CKYTX      CKYTX      CKYTX
          ******************************************
                    ************************
                              ******

SUBROUTINE CKYTX (Y, ICKWRK, RCKWRK, X)
   Returns the mole fractions given the mass fractions; see Eq. (6).

INPUT
   Y       - Mass fractions of the species
             cgs units - none
             Data type - real array
             Dimension Y(*) at least KK, the total number of species.
   ICKWRK  - Array of integer workspace
             Data type - integer array
             Dimension ICKWRK(*) at least LENIWK.
   RCKWRK  - Array of real work space.
             Data type - real array
             Dimension RCKWRK(*) at least LENRWK.
OUTPUT
   X       - Mole fractions of the species
             cgs units - none
             Data type - real array
             Dimension X(*) at least KK, the total number of species.
```

VII. SAMPLE PROBLEM

Before applying CHEMKIN, the user must (1) define a system of governing equations, (2) define a reaction mechanism, and (3) choose a solution method. In this sample problem we will solve the equations describing constant pressure combustion for a hydrogen-air reaction mechanism. The governing energy and mass conservation equations are

$$\frac{dT}{dt} = -\frac{1}{\rho \bar{c}_p} \sum_{k=1}^{K} h_k \dot{\omega}_k W_K,$$

$$\frac{dY_k}{dt} = \frac{\dot{\omega}_k W_k}{\rho}, \quad k = 1, \ldots, K,$$

where T is temperature and Y_k are the mass fractions of the K species involved. The independent variable t is time. Other variables are ρ, mass density; $\bar{c}_p$, mean specific heat at constant pressure; h_k, the specific enthalpies of the species; $\dot{\omega}_k$, the molar production rates of the species; and W_k, the molecular weights of the species.

The governing system of ordinary differential equations and accompanying initial conditions form an initial value problem. The equations will be solved using the code LSODE[11] written by Alan Hindmarsh. We find this code to be highly reliable for the solution of wide range of stiff initial-value problems.

The Fortran code for solution of the sample problem is given in Section 4 below. After initializing Chemkin, the code reads the initial nonzero moles from input. It then repeatedly calls subroutine LSODE to obtain the solution at uniform print intervals. The governing equation formulation is found in SUBROUTINE FUN, which is called by LSODE.

The sections below present a VAX command procedure for the sample problem, Chemkin Interpreter input and output, and the input, Fortran code, and output for the sample problem. The last section describes how to use LSODE.

1. VAX Command Procedure

VAX/VMS Commands			Meaning
$assign	MECHANISM.DAT	FOR015	Assign the user's reaction mechanism to Fortran unit 15. This is the input file for the Chemkin Interpreter.
$assign	INTERP.OUT	FOR016	Assign the output file for printed output from the Chemkin Interpreter. The Interpreter writes to unit 16.
$assign	THERMO.DAT	FOR017	Assign the Thermodynamic Database to Fortran unit 17.
$assign	LINK.BIN	FOR025	Assign the Linking file to Fortran unit 25.
$run	INTERP.EXE		Execute the Interpreter.
$for	SAMPLE.FOR		Compile the user's Fortran program.
$assign	SAMPLE.INP	FOR005	Assign a file containing any input required by the user's program to Fortran unit 5.
$assign	SAMPLE.OUT	FOR006	Assign a file to accept any printed output from the user's program to Fortran unit 6.
$link	SAMPLE.OBJ, LSDOE CKLIB/LIB		Link the user's program with the Chemkin Gas-Phase Subroutine Library LSODE.
$run	SAMPLE		Execute the user's program.

2. Input to Interpreter

```
ELEMENTS  H O N END
SPECIES   H2 H O2 O OH HO2 H2O2 H2O N N2 NO END

REACTIONS
   H2+O2=2OH                          0.170E+14   0.00   47780
   OH+H2=H2O+H                        0.117E+10   1.30    3626  ! D-L&W
   O+OH=O2+H                          0.400E+15  -0.50       0  ! JAM 1986
   O+H2=OH+H                          0.506E+05   2.67    6290  ! KLEMM ET AL., 1986
   H+O2+M=HO2+M                       0.361E+18  -0.72       0  ! DIXON-LEWIS
         H2O/18.6/ H2/2.86/ N2/1.26/
   OH+HO2=H2O+O2                      0.750E+13   0.00       0  ! D-L
   H+HO2=2OH                          0.140E+15   0.00    1073  ! D-L
   O+HO2=O2+OH                        0.140E+14   0.00    1073  ! D-L
   2OH=O+H2O                          0.600E+09   1.30       0  ! COHEN-WEST
   H+H+M=H2+M                         0.100E+19  -1.00       0  ! D-L
         H2O/0.0/ H2/0.0/
   H+H+H2=H2+H2                       0.920E+17  -0.60       0
   H+H+H2O=H2+H2O                     0.600E+20  -1.25       0
   H+OH+M=H2O+M                       0.160E+23  -2.00       0  ! D-L
         H2O/5/
   H+O+M=OH+M                         0.620E+17  -0.60       0  ! D-L
         H2O/5/
   O+O+M=O2+M                         0.189E+14   0.00   -1788  ! NBS
   H+HO2=H2+O2                        0.125E+14   0.00       0  ! D-L
   HO2+HO2=H2O2+O2                    0.200E+13   0.00       0
   H2O2+M=OH+OH+M                     0.130E+18   0.00   45500
   H2O2+H=HO2+H2                      0.160E+13   0.00    3800
   H2O2+OH=H2O+HO2                    0.100E+14   0.00    1800
   O+N2=NO+N                          0.140E+15   0.00   75800
   N+O2=NO+O                          0.640E+10   1.00    6280
   OH+N=NO+H                          0.400E+14   0.00       0
END
```

3. Output from Interpreter

```
CHEMKIN INTERPRETER OUTPUT:  CHEMKIN-II Version 1.3, May 1989
                             DOUBLE PRECISION

                          --------------------
                          ELEMENTS     ATOMIC
                          CONSIDERED   WEIGHT
                          --------------------
                           1. H        1.00797
                           2. O        15.9994
                           3. N        14.0067
                          --------------------

-------------------------------------------------------------------------------
                    P  C
                    H  H
                    A  A
SPECIES             S  G  MOLECULAR   TEMPERATURE     ELEMENT COUNT
CONSIDERED          E  E  WEIGHT      LOW      HIGH   H  O  N
-------------------------------------------------------------------------------
  1. H2             G  0    2.01594   300.0  5000.0   2  0  0
  2. H              G  0    1.00797   300.0  5000.0   1  0  0
  3. O2             G  0   31.99880   300.0  5000.0   0  2  0
  4. O              G  0   15.99940   300.0  5000.0   0  1  0
  5. OH             G  0   17.00737   300.0  5000.0   1  1  0
  6. HO2            G  0   33.00677   300.0  5000.0   1  2  0
  7. H2O2           G  0   34.01474   300.0  5000.0   2  2  0
  8. H2O            G  0   18.01534   300.0  5000.0   2  1  0
  9. N              G  0   14.00670   300.0  5000.0   0  0  1
 10. N2             G  0   28.01340   300.0  5000.0   0  0  2
 11. NO             G  0   30.00610   300.0  5000.0   0  1  1
-------------------------------------------------------------------------------


     REACTIONS CONSIDERED                                  PRE EXP   TEMP EXP   ACT ENG

  1. H2+O2=2OH                                            0.170E+14   0.000  47780.000
  2. OH+H2=H2O+H                                           0.117E+10   1.300   3626.000
  3. O+OH=O2+H                                             0.400E+15  -0.500      0.000
  4. O+H2=OH+H                                             0.506E+05   2.670   6290.000
  5. H+O2+M=HO2+M                                          0.361E+18  -0.720      0.000
        H2O          Enhanced by     1.860E+01
        H2           Enhanced by     2.860E+00
        N2           Enhanced by     1.260E+00
  6. OH+HO2=H2O+O2                                         0.750E+13   0.000      0.000
  7. H+HO2=2OH                                             0.140E+15   0.000   1073.000
  8. O+HO2=O2+OH                                           0.140E+14   0.000   1073.000
  9. 2OH=O+H2O                                             0.600E+09   1.300      0.000
 10. H+H+M=H2+M                                            0.100E+19  -1.000      0.000
        H2O          Enhanced by     0.000E+00
        H2           Enhanced by     0.000E+00
 11. H+H+H2=H2+H2                                          0.920E+17  -0.600      0.000
 12. H+H+H2O=H2+H2O                                        0.600E+20  -1.250      0.000
 13. H+OH+M=H2O+M                                          0.160E+23  -2.000      0.000
        H2O          Enhanced by     5.000E+00
 14. H+O+M=OH+M                                            0.620E+17  -0.600      0.000
        H2O          Enhanced by     5.000E+00
 15. O+O+M=O2+M                                            0.189E+14   0.000  -1788.000
 16. H+HO2=H2+O2                                           0.125E+14   0.000      0.000
 17. HO2+HO2=H2O2+O2                                       0.200E+13   0.000      0.000
 18. H2O2+M=OH+OH+M                                        0.130E+18   0.000  45500.000
 19. H2O2+H=HO2+H2                                         0.160E+13   0.000   3800.000
 20. H2O2+OH=H2O+HO2                                       0.100E+14   0.000   1800.000
 21. O+N2=NO+N                                             0.140E+15   0.000  75800.000
 22. N+O2=NO+O                                             0.640E+10   1.000   6280.000
 23. OH+N=NO+H                                             0.400E+14   0.000      0.000

 NOTE:  A units mole-cm-sec-K, E units cal/mole


NO ERRORS FOUND ON INPUT...CHEMKIN LINKING FILE WRITTEN.

WORKING SPACE REQUIREMENTS ARE
   INTEGER:      461
   REAL:         469
   CHARACTER:     14
```

4. User's Fortran Code

```
      PROGRAM CONP
C
C     Integration of adiabatic, constant pressure kinetics problems
C
C*****double precision
      IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER(I-N)
C*****END double precision
C*****single precision
C     IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
C*****END single precision
C
      PARAMETER (LENIWK=4000, LENRWK=4000, LENCWK=500, NK=5, NLMAX=55,
     1     LRW=1000, LIW=100, LIN=5, LOUT=6, LINCK=25, KMAX=50)
      DIMENSION X(KMAX), Z(KMAX), ELWRK(LRW), IELWRK(LIW), VAL(10)
C
      COMMON /PARAM/ ICKWRK(4000), RCKWRK(4000), KK, P, RU, WT(50),
     1        WDOT(50), H(50)
C
      CHARACTER CCKWRK(LENCWK)*16, KSYM(KMAX)*16, LINE*80
      LOGICAL KERR, IERR
      DATA KERR/.FALSE./, X/KMAX*0.0/, KSYM/KMAX*' '/
      EXTERNAL FUN
C
C     Open the Chemkin LINK file
C
      OPEN(UNIT=LINCK, STATUS='OLD', FORM='UNFORMATTED')
C
C     Initialize Chemkin
C
      CALL CKINIT (LENIWK, LENRWK, LENCWK, LINCK, LOUT, ICKWRK,
     1     RCKWRK, CCKWRK)
      CALL CKINDX (ICKWRK, RCKWRK, MM, KK, II, NFIT)
C
      IF (KK .GT. KMAX) THEN
      WRITE(LOUT, *)  ' Species dimension too small...must be at least ',KK
      STOP
      ENDIF
C
      CALL CKSYMS(CCKWRK, LOUT, KSYM, IERR)
      IF (IERR) KERR = .TRUE.
      CALL CKWT(ICKWRK, RCKWRK, WT)
      CALL CKRP(ICKWRK, RCKWRK, RU, RUC, PATM)
C
C     Pressure and temperature
C
      WRITE(LOUT, '(/A)') ' ADIABATIC FIXED PRESSURE PROBLEM'
      WRITE(LOUT, '(/A)') ' INPUT PRESSURE(ATM) AND TEMPERATURE(K)'
      READ (LIN,   *) PA, T
      WRITE(LOUT,7105) PA, T
      P = PA*PATM
```

```
C
C     Initial nonzero moles
C
   40 CONTINUE
      LINE = ' '
      WRITE(LOUT, '(/A)') ' INPUT MOLES OF NEXT SPECIES'
      READ(LIN,  '(A)', END=45)  LINE
      WRITE(LOUT, '(X,A)')  LINE
      ILEN = INDEX(LINE, '!')
      IF (ILEN .EQ. 1) GO TO 40
C
      IF (ILEN .NE. 1) THEN
            ILEN = ILEN - 1
            IF (ILEN .LE. 0)  ILEN = LEN(LINE)
            IF (INDEX(LINE(:ILEN), 'END') .EQ. 0)  THEN
C
                  IF (LINE(:ILEN) .NE. ' ')  THEN
                        CALL (CKSNUM(LINE(:ILEN), 1, LOUT, KSYM, KK, KNUM,
     1                                          NVAL, VAL, IERR)
                        IF (IERR)  THEN
                              WRITE(LOUT,*)  ' Error reading moles...'
                              KERR = .TRUE.
                        ELSE
                              X(KNUM) = VAL(1)
                        ENDIF
                  ENDIF
                  GO TO 40
            ENDIF
      ENDIF
C
   45 CONTINUE
C
C     Final time and print interval
C
      WRITE(LOUT, '(/A)') ' INPUT FINAL TIME AND DT'
      READ (LIN,   *) T2, DT
      WRITE(LOUT,7105) T2, DT
C
      IF (KERR) STOP
C
C     Normalize the mole fractions
C
      XTOT = 0.00
      DO 50 K=1,KK
            XTOT = XTOT + X(K)
   50 CONTINUE
      DO 55 K=1,KK
            X(K) = X(K) / XTOT
   55 CONTINUE
C
C     Initial conditions and mass fractions
C
      TT1  = 0.0
      Z(1) = T
```

```
      CALL CKXTY (X, ICKWRK, RCKWRK, Z(2))
C
C     Integration control parameters for LSODE
C
      TT2  = TT1
      NEQ  = KK + 1
      MF   = 22
      ITOL = 1
      IOPT = 0
      RTOL = 1.0E-6
      ITASK = 1
      ATOL = 1.0E-15
      ISTATE= 1
      NLINES=NLMAX + 1
C
C     Integration loop
C
  250 CONTINUE
      IF (NLINES .GE. NLMAX) THEN
C
C           Print page heading
C
            WRITE(LOUT, 7003)
            WRITE(LOUT, 7100) (KSYM(K)(:10), K=1,MIN(NK,KK))
            NLINES = 1
C
            DO 200 K1 = NK+1, KK, NK
                  WRITE(LOUT, 7110) (KSYM(K)(:10),K=K1, MIN(K1+NK-1, KK))
                  NLINES = NLINES + 1
  200 CONTINUE
      ENDIF
C
C     Print the solution
C
      T = Z(1)
      CALL CKYTX (Z(2), ICKWRK, RCKWRK, X)
C
      WRITE(LOUT, 7105) TT1, T, (X(K), K=1,MIN(NK,KK))
      NLINES = NLINES + 1
C
      DO 300 K1 = NK+1, KK, NK
            WRITE(LOUT, 7115) (X(K), K=K1, MIN(K1+NK-1,KK))
            NLINES = NLINES + 1
  300 CONTINUE
C
      IF (TT2 .GE. T2) STOP
      TT2 = MIN(TT2 + DT, T2)
C
C     Call the differential equation solver
C
  350 CONTINUE
      CALL LSODE (FUN, NEQ, Z, TT1, TT2, ITOL, RTOL, ATOL, ITASK,ISTATE, IOPT,
     1                  ELWRK, LRW, IELWRK, LIW, JAC, MF)
C
```

```
      IF (ISTATE .LE. -2) THEN
            IF (ISTATE .EQ. -1) THEN
                  ISTATE = 2
                  GO TO 350
            ELSE
                  WRITE(LOUT,*) ' ISTATE=',ISTATE
                  STOP
            ENDIF
      ENDIF
      GO TO 250
C
C           FORMATS
C
 7003 FORMAT (1H1)
 7100 FORMAT (2X, 'T(SEC)', 6X, 'TMP(K)', 6X, 5(1X,A10))
 7105 FORMAT (12E11.3)
 7110 FORMAT (26X, 5(1X,A10))
 7115 FORMAT (22X, 10E11.3)
      END
C
```

```
      SUBROUTINE FUN (N, TIME, Z, ZP)
C
C*****double  precision
      IMPLICIT DOUBLE PRECISION(A-H,O-Z), INTEGER(I-N)
C*****END  double  precision
C*****single  precision
C     IMPLICIT REAL (A-H,O-Z), INTEGER(I-N)
C*****END  single  precision
C
      DIMENSION Z(N), ZP(N)
      COMMON /PARAM/ ICKWRK(4000), RCKWRK(4000), KK, P, RU, WT(50),
     1               WDOT(50), H(50)
C
C     Variables in Z are:  Z(1)   = T
C           Z(K+1) = Y(K)
C
C     Call  Chemkin  subroutines
C
      CALL CKRHOY (P, Z(1), Z(2), ICKWRK, RCKWRK, RHO)
      CALL CKCPBS (Z(1), Z(2), ICKWRK, RCKWRK, CPB)
      CALL CKWYP  (P, Z(1), Z(2), ICKWRK, RCKWRK, WDOT)
      CALL CKHMS  (Z(1), ICKWRK, RCKWRK, H)
C
C     Form  governing  equation
C
      SUM = 0.0
      DO 100 K=1,KK
           ZP(K+1) = WDOT(K) * WT(K) / RHO
           SUM = SUM + H(K) * WDOT(K) * WT(K)
  100 CONTINUE
      ZP(1) = -SUM / (RHO*CPB)
C
      RETURN
      END
```

5. Input to Fortran Code

```
1 1000
H2 1
O2 3
N2 .1
END
3.0E-4 3.0E-5
```

6. Output from Fortran Code

```
CKLIB:  Chemical Kinetics Library
        CHEMKIN-II Version 1.6, June 1989
        DOUBLE PRECISION

ADIABATIC FIXED PRESSURE PROBLEM

INPUT PRESSURE(ATM) AND TEMPERATURE(K)
 0.100E+01  0.100E+04

INPUT MOLES OF NEXT SPECIES
H2 1

INPUT MOLES OF NEXT SPECIES
O2 3

INPUT MOLES OF NEXT SPECIES
N2 .1

INPUT MOLES OF NEXT SPECIES
END

INPUT FINAL TIME AND DT
 0.300E-03  0.300E-04

 T(SEC)       TMP(K)        H2          H           O2          O           OH
                            HO2         H2O2        H2O         N           N2
                            NO
 0.000E+00  0.100E+04  0.244E+00  0.000E+00  0.732E+00  0.000E+00  0.000E+00
                       0.000E+00  0.000E+00  0.000E+00  0.000E+00  0.244E-01
                       0.000E+00
 0.300E-04  0.100E+04  0.244E+00  0.814E-05  0.732E+00  0.424E-05  0.144E-05
                       0.129E-04  0.103E-07  0.258E-04  0.180E-20  0.244E-01
                       0.372E-19
 0.600E-04  0.196E+04  0.891E-02  0.169E-01  0.625E+00  0.571E-01  0.411E-01
                       0.175E-03  0.357E-04  0.224E+00  0.228E-09  0.262E-01
                       0.165E-07
 0.900E-04  0.235E+04  0.367E-02  0.332E-02  0.657E+00  0.235E-01  0.392E-01
                       0.846E-04  0.445E-05  0.246E+00  0.192E-08  0.271E-01
                       0.163E-05
 0.120E-03  0.243E+04  0.258E-02  0.185E-02  0.665E+00  0.165E-01  0.352E-01
                       0.693E-04  0.254E-05  0.251E+00  0.229E-08  0.272E-01
                       0.437E-05
 0.150E-03  0.246E+04  0.216E-02  0.139E-02  0.669E+00  0.138E-01  0.330E-01
                       0.641E-04  0.197E-05  0.254E+00  0.235E-08  0.273E-01
                       0.729E-05
 0.180E-03  0.248E+04  0.197E-02  0.120E-02  0.670E+00  0.125E-01  0.319E-01
                       0.619E-04  0.173E-05  0.255E+00  0.237E-08  0.273E-01
                       0.102E-04
 0.210E-03  0.248E+04  0.188E-02  0.111E-02  0.671E+00  0.119E-01  0.313E-01
                       0.609E-04  0.162E-05  0.255E+00  0.238E-08  0.273E-01
                       0.131E-04
 0.240E-03  0.249E+04  0.183E-02  0.106E-02  0.671E+00  0.116E-01  0.310E-01
                       0.604E-04  0.157E-05  0.256E+00  0.239E-08  0.273E-01
                       0.159E-04
 0.270E-03  0.249E+04  0.181E-02  0.104E-02  0.672E+00  0.115E-01  0.308E-01
                       0.602E-04  0.154E-05  0.256E+00  0.240E-08  0.273E-01
                       0.188E-04
 0.300E-03  0.249E+04  0.179E-02  0.103E-02  0.672E+00  0.114E-01  0.307E-01
                       0.600E-04  0.152E-05  0.256E+00  0.241E-08  0.273E-01
                       0.217E-04
```

7. LSODE Summary

```
      subroutine lsode (f, neq, y, t, tout, itol, rtol, atol, itask,
     1        istate, iopt, rwork, lrw, iwork, liw, jac, mf)
      external f, jac
      integer neq, itol, itask, istate, iopt, lrw, iwork, liw, mf
      double precision y, t, tout, rtol, atol, rwork
      dimension neq(1), y(1), rtol(1), atol(1), rwork(lrw), iwork(liw)
c-----------------------------------------------------------------------
c this is the march 30, 1987 version of
c lsode.. livermore solver for ordinary differential equations.
c this version is in double precision.
c
c lsode solves the initial value problem for stiff or nonstiff
c systems of first order ode-s,
c     dy/dt = f(t,y) ,  or, in component form,
c     dy(i)/dt = f(i) = f(i,t,y(1),y(2),...,y(neq)) (i = 1,...,neq).
c lsode is a package based on the gear and gearb packages, and on the
c october 23, 1978 version of the tentative odepack user interface
c standard, with minor modifications.
c-----------------------------------------------------------------------
c  reference..
c     alan c. hindmarsh,  odepack, a systematized collection of ode
c     solvers, in scientific computing, r. s. stepleman et al. (eds.),
c     north-holland, amsterdam, 1983, pp. 55-64.
c-----------------------------------------------------------------------
c author and contact    alan c. hindmarsh,
c                       computing and mathematics research div., l-316
c                       lawrence livermore national laboratory
c                       livermore, ca 94550.
c-----------------------------------------------------------------------
c summary of usage.
c
c communication between the user and the lsode package, for normal
c situations, is summarized here.  this summary describes only a subset
c of the full set of options available.  see the full description for
c details, including optional communication, nonstandard options,
c and instructions for special situations.  see also the example
c problem (with program and output) following this summary.
c
c a. first provide a subroutine of the form..
c       subroutine f (neq, t, y, ydot)
c       dimension y(neq), ydot(neq)
c which supplies the vector function f by loading ydot(i) with f(i).
c
c b. next determine (or guess) whether or not the problem is stiff.
c stiffness occurs when the jacobian matrix df/dy has an eigenvalue
c whose real part is negative and large in magnitude, compared to the
c reciprocal of the t span of interest.  if the problem is nonstiff,
c use a method flag mf = 10.  if it is stiff, there are four standard
c choices for mf, and lsode requires the jacobian matrix in some form.
c this matrix is regarded either as full (mf = 21 or 22),or banded
c (mf = 24 or 25).  in the banded case, lsode requires two half-bandwidth
```

```
c parameters ml and mu.  these are, respectively, the widths of the lower
c and upper parts of the band, excluding the main diagonal.  thus the
c band consists of the locations (i,j) with i-ml .le. j .le. i+mu, and the full
c bandwidth is ml+mu+1.
c
c c. if the problem is stiff, you are encouraged to supply the jacobian
c directly (mf = 21 or 24), but if this is not feasible, lsode will
c compute it internally by difference quotients (mf = 22 or 25).
c if you are supplying the jacobian, provide a subroutine of the form..
c       subroutine jac (neq, t, y, ml, mu, pd, nrowpd)
c       dimension  y(neq),  pd(nrowpd,neq)
c which supplies df/dy by loading pd as follows..
c     for a full jacobian (mf = 21), load pd(i,j) with df(i)/dy(j),
c the partial derivative of f(i) with respect to y(j).  (ignore the
c ml and mu arguments in this case.)
c     for a banded jacobian (mf = 24), load pd(i-j+mu+1,j) with
c df(i)/dy(j), i.e. load the diagonal lines of df/dy into the rows of
c pd from the top down.
c     in either case, only nonzero elements need be loaded.
c
c d. write a main program which calls subroutine lsode once for
c each point at which answers are desired.  this should also provide
c for possible use of logical unit 6 for output of error messages
c by lsode.  on the first call to lsode, supply arguments as follows..
c f        =  name of subroutine for right-hand side vector f.
c             this name must be declared external in calling program.
c neq      =  number of first order ode-s.
c y        =  array of initial values, of length neq.
c t        =  the initial value of the independent variable.
c tout     =  first point where output is desired (.ne. t).
c itol     =  1 or 2 according as atol (below) is a scalar or array.
c rtol     =  relative tolerance parameter (scalar).
c atol     =  absolute tolerance parameter (scalar or array).
c             the estimated local error in y(i) will be controlled so as
c             to be roughly less (in magnitude) than
c                  ewt(i) = rtol*abs(y(i)) + atol      if itol = 1, or
c                  ewt(i) = rtol*abs(y(i)) + atol(i)   if itol = 2.
c             thus the local error test passes if, in each component,
c             either the absolute error is less than atol (or atol(i)),
c             or the relative error is less than rtol.
c             use rtol = 0.0 for pure absolute error control, and
c             use atol = 0.0 (or atol(i) = 0.0) for pure relative error
c             control.  caution.. actual (global) errors may exceed these
c             local tolerances, so choose them conservatively.
c itask    =  1 for normal computation of output values of y at t = tout.
c istate   =  integer flag (input and output).  set istate = 1.
c iopt     =  0 to indicate no optional inputs used.
c rwork    =  real work array of length at least..
c             20 + 16*neq                      for mf = 10,
c             22 +  9*neq + neq**2             for mf = 21 or 22,
c             22 + 10*neq + (2*ml + mu)*neq    for mf = 24 or 25.
c lrw      =  declared length of rwork (in user-s dimension).
```

```
c iwork     =  integer work array of length at least..
c                  20          for mf = 10,
c                  20 + neq    for mf = 21, 22, 24, or 25.
c              if mf = 24 or 25, input in iwork(1),iwork(2) the lower
c              and upper half-bandwidths ml,mu.
c liw       =  declared length of iwork (in user-s dimension).
c jac       =  name of subroutine for jacobian matrix (mf =     21 or 24).
c              if used, this name must be declared external in calling
c              program.  if not used, pass a dummy name.
c mf        =  method flag.  standard values are..
c              10 for nonstiff (adams) method, no jacobian used.
c              21 for stiff (bdf) method, user-supplied full jacobian.
c              22 for stiff method, internally generated full jacobian.
c              24 for stiff method, user-supplied banded jacobian.
c              25 for stiff method, internally generated banded jacobian.
c note that the main program must declare arrays y, rwork, iwork,
c and possibly atol.
c
c e. the output from the first call (or any call) is..
c         y =  array of computed values of y(t) vector.
c         t =  corresponding value of independent variable (normally tout).
c istate    =  2  if lsode was successful, negative otherwise.
c              -1 means excess work done on this call (perhaps wrong mf).
c              -2 means excess accuracy requested (tolerances too small).
c              -3 means illegal input detected (see printed message).
c              -4 means repeated error test failures (check all inputs).
c              -5 means repeated convergence failures (perhaps bad jacobian
c                 supplied or wrong choice of mf or tolerances).
c              -6 means error weight became zero during problem. (solution
c                 component i vanished, and atol or atol(i) = 0.)
c
c f. to continue the integration after a successful return, simply
c reset tout and call lsode again.  no other parameters need be reset.
c
c-----------------------------------------------------------------------------
```

APPENDIX A. STORAGE ALLOCATION FOR THE WORK ARRAYS

The work arrays contain all the pertinent information about the species and the reaction mechanism. They also contain some work space needed by various routines for internal manipulations. If a user wishes to modify a CKLIB subroutine or to write new routines, he will probably want to use the work arrays directly. The starting addresses for information stored in the work arrays are found in the labeled common block, COMMON /CKSTRT/, and are explained below.

```
   COMMON /CKSTRT/ NMM , NKK , NII , MXSP, MXTB, MXTP, NCP , NCP1,
  1                NCP2, NCP2T,NPAR, NLAR, NFAR, NLAN, NFAL, NREV,
  2                NTHB, NRLT, NWL,  IcMM, IcKK, IcNC, IcPH, IcCH,
  3                IcNT, IcNU, IcNK, IcNS, IcNR, IcLT, IcRL, IcRV,
  4                IcWL, IcFL, IcFO, IcKF, IcTB, IcKN, IcKT, NcAW,
  5                NcWT, NcTT, NcAA, NcCO, NcRV, NcLT, NcRL, NcFL,
  6                NcKT, NcWL, NcRU, NcRC, NcPA, NcK1, NcK2, NcK3,
  7                NcK4, NcI1, NcI2, NcI3, NcI4
```

INDEX CONSTANTS.

- NMM - Total number of elements in problem.
- NKK - Total number of species in problem.
- NII - Total number of reactions in problem.
- MXSP - Maximum number of species (reactants plus products) allowed for any reaction, unless changed in the interpreter, MXSP=6.
- MXTB - Maximum number of enhanced third-bodies allowed for any reaction; unless changed in the interpreter, MXTB=10.
- MXTP - Maximum number of temperatures allowed in fits of thermodynamic properties for any species; unless changed in the interpreter and the thermodynamic database, MXTP=3.
- NCP - Number of polynomial coefficients to fits of CP/R for a species; unless changed in the interpreter and the thermodynamic database, NCP=5.
- NCP1 - NCP + 1
- NCP2 - NCP + 2
- NCP2T - Total number of thermodynamic fit coefficients for the species; unless changed, NCP2T = (MXTP-1)*NCP2 = 14.
- NPAR - Number of parameters required in the rate expression for the reactions; in the current formulation NPAR=3.
- NLAR - Number of parameters required for Landau-Teller reactions; NLAR=4.
- NFAR - Number of parameters allowed for fall-off reactions; NFAR=8.
- NLAN - Total number of Landau-Teller reactions.
- NFAL - Total number of fall-off reactions.
- NREV - Total number of reactions with reverse parameters.
- NTHB - Total number of reactions with third-bodies.
- NRLT - Total number of Landau-Teller reactions with reverse parameters.
- NWL - Total number of reactions with radiation wavelength enhancement factors.

STARTING ADDRESSES FOR THE CHARACTER WORK SPACE, CCKWRK.

- IcMM - Starting address of an array of the NMM element names. CCKWRK(IcMM+M-1) is the name of the Mth element.
- IcKK - Starting address of an array of the NKK species names. CCKWRK(icKK+M-1) is the name of the Kth species.

STARTING ADDRESSES FOR THE INTEGER WORK SPACE, ICKWRK.

- IcNC - Starting address of an array of the elemental content of the NMM elements in the NKK species. ICKWRK(IcNC+(K-1)*NMM+M-1) is the number of atoms of the Mth element in the Kth species.
- IcPH - Starting address of an array of phases of the NKK species. ICKWRK(IcPH+K-1) = -1, the Kth species is a solid
  = 0, the Kth species is a gas
  = +1, the Kth species is a liquid
- IcCH - Starting address of an array of the electronic charges of the NKK species. ICKWRK(IcCH+K-1) = -2, the Kth species has two excess electrons.
- IcNT - Starting address of an array of the number of temperatures used to fit thermodynamic coefficients for the NKK species. ICKWRK(IcNT+K-1) = N, N temperatures were used in the fit for the Kth species.
- IcNU - Starting address of a matrix of stoichiometric coefficients of the MXSP species in the NII reactions. ICKWRK(IcNU+(I-1)*MXSP+N-1) is the coefficient of the Nth participant species in the Ith reaction.
- IcNK - Starting address of a matrix of species index numbers for the MXSP species in the NII reactions. ICKWRK(IcNK+(I-1)*MXSP+N-1) = K, the species number of the Nth participant species in the Ith reaction.

IcNS - Starting address of an array of the total number of participant species for the NII reactions, and the reversibility of the reactions.
ICKWRK(IcNS+I-1) = +N, the Ith reaction is reversible and has N participant species (reactants + products)
= -N, the Ith reaction is irreversible and has N participant species (reactants + products)

IcNR - Starting address of an array of the number of reactants only for the NII reactions.
ICKWRK(IcNR+I-1) is the total number of reactants in the Ith reaction.

IcLT - Starting address of an array of the NLAN reaction numbers for which Landau-Teller parameters have been given.
ICKWRK(IcLT+N-1) is the reaction number of the Nth Landau-Teller reaction.

IcRL - Starting address of an array of the NRLT reaction numbers for which reverse Landau-Teller parameters have been given.
ICKWRK(IcRL+N-1) is the reaction number of the Nth reaction with reverse Landau-Teller parameters.

IcRV - Starting address of an array of the NREV reaction numbers for which reverse Arhennius coefficients have been given.
ICKWRK(IcRV+N-1) is the reaction number of the Nth reaction with reverse coefficients.

IcWL - Starting address of an array of the NWL reactions numbers for which radiation wavelength has been given. ICKWRK(IcWL+N-1) is the reaction number of the Nth reaction with wavelength enhancement.

IcFL - Starting address of an array of the NFAL reaction numbers with fall-off parameters. ICKWRK(IcFL+N-1) is the reaction number of the Nth fall-off reaction.

IcFO - Starting address of an array describing the type of the NFAL fall-off reactions. ICKWRK(IcFO+N-1) is the type of the Nth fall-off reaction:
1 for 3-parameter Lindemann form
2 for 6- or 8-parameter SRI form
3 for 6-parameter Troe form
4 for 7-parameter Troe form

IcKF - Starting address of an array of the third-body species numbers for the NFAL fall-off reactions.
ICKWRK(IcKF+N-1) = 0: the concentration of the third-body is the total of the concentrations of all species in the problem
= K: the concentration of the third-body is the concentration of species K.

IcTB - Starting address of an array of reaction numbers for the NTHB third-body reactions. ICKWRK(IcTB+N-1) is the reaction number of the Nth third-body reaction.

IcKN - Starting address of an array of the number of enhanced third bodies for the NTHB third-body reactions.
ICKWRK(IcKN+N-1) is the number of enhanced species for the Nth third-body reaction.

IcKT - Starting address of an array of species numbers for the MXTB enhanced 3rd bodies in the NTHB third-body reactions.
ICKWRK(IcTB+(N-1)*MXTB+L-1) is the species number of the Lth enhanced species in the Nth third-body reaction.

STARTING ADDRESSES FOR THE REAL WORK SPACE, RCKWRK.

NcAW - Starting address of an array of atomic weights of the NMM elements (gm/mole).
RCKWRK(NcAW+M-1) is the atomic weight of element M.

NcWT - Starting address of an array of molecular weights for the NKK species (gm/mole).
RCKWRK(NcWT+K-1) is the molecular weight of species K.

NcTT - Starting address of an array of MXTP temperatures used in the fits of thermodynamic properties of the NKK species (kelvins).
RCKWRK(NcTT+(K-1)*MXTP+N-1) is the Nth temperature for the Kth species.

NcAA - Starting address of a three-dimensional array of coefficients for the NCP2 fits to the thermodynamic properties for the NKK species, for (MXTP-1) temperature ranges.
RCKWRK(NcAA+(L-1)*NCP2+(K-1)*NCP2T+N-1) = A(N,L,K);
A(N,L,K),N=1,NCP2T = polynomial coefficients in the fits for the Kth species and the Lth temperature range, where the total number of temperature ranges for the Kth species is ICKWRK(IcNT+K-1) - 1.

NcCO - Starting address of an array of NPAR Arrhenius parameters for the NII reactions. RCKWRK(NcCO+(I-1)*NPAR+(L-1)) is the Lth parameter of the Ith reaction, where
L=1 is the pre-exponential factor (mole-cm-sec-K),
L=2 is the temperature exponent, and
L=3 is the activation energy (kelvins).

NcRV - Starting address of an array of NPAR reverse Arrhenius parameters for the NREV reactions. RCKWRK(NcRV+(N-1)*NPAR+(L-1)) is the Lth reverse parameter for the Nth reaction with reverse parameters defined, where
L=1 is the pre-exponential factor (mole-cm-sec-K),
L=2 is the temperature exponent, and
L=3 is the activation energy (kelvins).
The reaction number is ICKWRK(IcRV+N-1).

NcLT - Starting location of an array of the NLAR parameters for the NLAN Landau-Teller reactions. RCKWRK(NcLT+(N-1)*NLAR+(L-1)) is the Lth Landau-Teller parameter for the Nth Landau-Teller reaction, where
L=1 is B(I) (Eq. 72) (kelvins**1/3), and
L=2 is C(I) (Eq. 72) (kelvins**2/3).
The reaction number is ICKWRK(IcLT+N-1).

NcRL - Starting location of an array of the NLAR reverse parameters for the NRLT Landau-Teller reactions for which reverse parameters were given. RCKWRK(NcRL+(N-1)*NLAR+(L-1)) is the Lth reverse parameter for the Nth reaction with reverse Landau-Teller parameters, where
L=1 is B(I) (Eq. 72) (kelvins**1/3), and
L=2 is C(I) (Eq. 72) (kelvins**2/3).
The reaction number is ICKWRK(IcRL+N-1).

NcFL - Starting location of an array of the NFAR fall-off parameters for the NFL fall-off reactions. RCKWRK(NcFL+(N-1)*NFAR+(L-1)) is the Lth fall-off parameter for the Nth fall-off reaction, where the low pressure limits are defined by
L=1 is the pre-exponential factor (mole-cm-sec-K),
L=2 is the temperature exponent, and
L=3 is the activation energy (kelvins).
Additional parameters define the centering, depending on the type of formulation -
Troe: L=4 is the Eq. 68 parameter a,
L=5 is the Eq. 68 parameter T*** (kelvins),
L=6 is the Eq. 68 parameter T* (kelvins), and
L=7 is the Eq. 68 parameter T** (kelvins).
SRI: L=4 is the Eq. 69 parameter a,
L=5 is the Eq. 69 parameter b (kelvins),
L=6 is the Eq. 69 parameter c (kelvins),
L=7 is the Eq. 69 parameter d, and
L=8 is the Eq. 69 parameter e.
The reaction number is ICKWRK(IcFL+N-1), and the type of formulation is ICKWRK(IcFO+N-1).

NcWL - Starting location of an array of wavelengths for the NWL wavelength-enhanced reactions.
RCKWRK(NcWL+N-1) is the wavelength enhancement (angstroms) for the Nth wavelength-enhanced reaction;
the reaction number is ICKWRK(IcWL+N-1).

NcKT - Starting location of an array of MXTB enhancement factors for the NTHB third-body reactions. RCKWRK(NcKT+(N-1)*MXTB+(L-1)) is the enhancement factor for the Lth enhanced species in the Nth third-body reaction; the reaction number is ICKWRK(IcTB+N-1), and the Lth enhanced species index number is ICKWRK(IcKT+(N-1)*MXTB+L-1).

NcRU - RCKWRK(NcRU) is the universal gas constant (ergs/mole-K).

NcRC - RCKWRK(NcRC) is the universal gas constant (cal/mole-K).

NcPA - RCKWRK(NcPA) is the pressure of one standard atmosphere (dynes/cm**2).

NcK1 - Starting addresses of arrays of internal work space
NcK2
NcK3 space of length NKK
NcK4

NcI1 - Starting addresses of arrays of internal work space
NcI2
NcI3 space of length NII
NcI4

The linking file consists of the following binary records:

```
 1) Index constants and information about linking file:
    KERR, LENI, LENR, LENC, NMM,  NKK,  NII,  MXSP, MXTB,
    MXTP, NCP,  NPAR, NLAR, NFAR, NREV, NFAL, NTHB,
    NLAN, NRLT, NWL, NCHRG
    Where KERR = logical which indicates if there was
                 an error in the Chemkin interpreter input.
          LENI = required length of ICKWRK.
          LENR = required length of RCKWRK.
          LENC = required length of CCKWRK.
          NCHRG= total number of species with an electronic
                 charge not equal to zero.

 2) Element information:
    ((CCKWRK(IcMM + M-1),                          !element names
     RCKWRK(NcAW + M-1)),                          !atomic weights
     M=1,NMM)

 3) Species information:
    ((CCKWRK(IcKK+K-1),                            !species names
     (ICKWRK(IcNC+(K-1)*NMM+M-1),M=1,MMM),         !composition
     ICKWRK(IcPH+K-1),                             !phase
     ICKWRK(IcCH+K-1),                             !charge
     RCKWRK(NcWT+K-1),                             !molecular weight
     ICKWRK(IcNT+K-1),                             !# of fit temps
     (RCKWRK(NcTT+(K-1)*MXTP + L-1),L=1,MXTP),     !array of temps
     ((RCKWRK(NcAA+(L-1)*NCP2+(K-1)*NCP2T+N-1),    !fit coeff'nts
               N=1,NCP2), L=1,(MXTP-1))),
     K = 1,NKK)

 4) Reaction information (if NII>0):
    (ICKWRK(IcNS+I-1),                             !# of species
     ICKWRK(IcNR+I-1),                             !# of reactants
     (RCKWRK(NcCO+(I-1)*NPAR+N-1), N=1,NPAR),      !Arr. coefficients
     (ICKWRK(IcNU+(I-1)*MXSP+N-1),                 !stoic coef
     ICKWRK(IcNK+(I-1)*MXSP+N-1), N=1,MXSP),       !species numbers
     I = 1,NII)

 5) Reverse parameter information (if NREV>0):
    (ICKWRK(IcRV+N-1),                             !reaction numbers
     (RCKWRK(NcRV+(N-1)*NPAR+L-1),L=1,NPAR),       !reverse coefficients
     N = 1,NREV)

 6) Fall-off reaction information (if NFAL>0):
    (ICKWRK(IcFL+N-1),                             !reaction numbers
     ICKWRK(IcFO+N-1),                             !fall-off option
     ICKWRK(IcKF+N-1),                             !3rd-body species
     (RCKWRK(NcFL+(N-1)*NFAR+L-1),L=1,NFAR),       !fall-off parameters
     N=1,NFAL)

 7) Third-body reaction information (if NTHB>0):
    (ICKWRK(IcTB+N-1),                             !reaction numbers
     ICKWRK(IcKN+N-1),                             !# of 3rd bodies
     (ICKWRK(IcKT+(N-1)*MXTB+L-1),                 !3rd-body species
     RCKWRK(NcKT+(N-1)*MXTB+L-1),L=1,MXTB),        !enhancement factors
     N=1,NTHB)

 8) Landau-Teller reaction information (if NLAN>0):
    (ICKWRK(IcLT+N-1),                             !reaction numbers
     (RCKWRK(NcLT+(N-1)*NLAR+L-1),L=1,NLAR),       !L-T parameters
     N=1,NLAN)

 9) Reverse Landau-Teller reaction information (if NRLT>0):
    (ICKWRK(IcRL+N-1),                             !reaction numbers
     (RCKWRK(NcRL+(N-1)*NLAR+L-1),L=1,NLAR),       !rev. L-T parameters
     N=1,NRLT)

10) Photon radiation reaction information (if NWL>0):
    (ICKWRK(IcWL+N-1),                             !reaction numbers
     RCKWRK(NcWL+N-1),                             !wavelength factor
     N=1,NWL)
```

REFERENCES

1. Kee, R. J., Warnatz, J, and Miller, J. A., "A Fortran Computer Code Package for the Evaluation of Gas-Phase Viscosities, Conductivities, and Diffusion Coefficients," Sandia National Laboratories Report, SAND83-8209 (1983).
2. Kee, R. J., Dixon-Lewis, G., Warnatz, J., Coltrin, M. E., Miller, J. A., "A Fortran Computer Package for the Evaluation of Gas-Phase, Multicomponent Transport Properties," Sandia National Laboratories report, in preparation.
3. Coltrin, M. E., Kee, R. J., and Rupley, F. M., to be published.
4. Kee, R. J. and Miller, J. A., "A Structured Approach to the Computational Modeling of Chemical Kinetics and Molecular Transport in Flowing Systems," *Springer Series in Chemical Physics* **47**, 196 (also Sandia National Laboratories Report, SAND80-8003) (1986).
5. Kee, R. J., Miller, J. A. and Jefferson, T. H., "Chemkin: A General-Purpose, Problem-Independent, Transportable, Fortran Chemical Kinetics Code Package," Sandia National Laboratories Report, SAND80-8003 (1980).
6. Kee, R. J., Rupley, F. M., and Miller, J. A., "The Chemkin Thermodynamic Database," Sandia National Laboratories Report, SAND87-8215 (1987).
7. Gordon S. and McBride, B. J., "Computer Program for Calculation of Complex Chemical Equilibrium Compositions, Rocket Performance, Incident and Reflected Shocks and Chapman-Jouguet Detonations, NASA SP-273 (1971).
8. Lindemann, F., *Trans. Faraday Soc.* **17**, 598 (1922).
9. Gilbert, R. G., Luther, K., and Troe, J., "Theory of Unimolecular Reactions in the Fall-Off Range," *Ber. Bunsenges. Phys. Chem.* **87**, 169–177 (1983).
10. Stewart, P. H., Larson, C. W., and Golden, D., "Pressure and Temperature Dependence of Reactions Proceeding via a Bound Complex. 2. Application to $2CH_3 \rightarrow C_2H_5 + H$," *Combust. and Flame* **75**, 25 (1989).
11. Hindmarsh, A. C. "ODEPACK, A Systemized Collection of ODE Solvers," in *Scientific Computing*, edited by R. S. Stephen et al., Vol 1, *IMACS Trans. on Scientific Computation* (North-Holland, Amsterdam), pp. 55–64 (1983).

UNLIMITED RELEASE

INITIAL DISTRIBUTION

C. K. Westbrook, LLNL, L-298

1126 M. E. Coltrin
1126 P. Ho

8000 J. C. Crawford
Attn: E. E. Ives, 8100
P. L. Mattern, 8300
R. C. Wayne, 8400
P. E. Brewer, 8500
8200 R. J. Detry
Attn: W. D. Wilson, 8230
R. C. Dougherty, 8270
R. A. Baroody, 8280
8240 C. W. Robinson
Attn: G. A. Benedetti, 8241
M. R. Birnbaum, 8242
M. L. Callabresi, 8243
C. M. Hartwig, 8244
8244 R. S. Larson
8245 N. E. Bergan
8245 G. H. Evans
8245 W. G. Houf
8245 R. J. Kee (200)
8245 A. E. Lutz
8245 E. Meeks
8245 F. M. Rupley
8351 J.-Y. Chen
8351 R. W. Dibble
8351 W. L. Flower
8353 D. W. Chandler
8353 J. A. Miller
8354 J. Cavolowsky
8357 C. F. Melius
8361 M. D. Allendorf
8361 R. E. Mitchell
8362 D. L. Siebers
8363 W. T. Ashurst
8363 P. K. Barr
8363 H. A. Dwyer

8535 Publications for OSTI (10)
8535 Publications/Technical Library Processes, 3141
3141 Technical Library Processes Division (3)
8524-2 Central Technical Files (3)