

UC-405
SAND96-8216
Unlimited Release
Printed May 1996

**CHEMKIN-III: A FORTRAN CHEMICAL KINETICS PACKAGE FOR THE ANALYSIS OF GAS-
PHASE CHEMICAL AND PLASMA KINETICS**

Robert J. Kee, Fran M. Rupley, and Ellen Meeks
Thermal and Plasma Processes Department

and

James A. Miller
Combustion Chemistry Department

Sandia National Laboratories
Livermore, CA 94551-0969

ABSTRACT

This document is the user's manual for the third-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 a 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 equations of state, thermodynamic properties, and chemical production rates. CHEMKIN-III includes capabilities for treating multi-fluid plasma systems, that are not in thermal equilibrium. These new capabilities allow researchers to describe chemistry systems that are characterized by more than one temperature, in which reactions may depend on temperatures associated with different species; i.e. reactions may be driven by collisions with electrons, ions, or charge-neutral species. These new features have been implemented in such a way as to require little or no changes to CHEMKIN implementation for systems in thermal equilibrium, where all species share the same gas temperature.

ACKNOWLEDGMENTS

CHEMKIN-III now has the capability to handle weakly ionized plasma chemistry, especially for applications related to advanced semiconductor processing. This aspect of the work was supported, in large part, through a Cooperative Research and Development Agreement (CRADA) with SEMATECH. Dr. Andrew Labun, at Digital Equipment Corporation, has been very generous of his time and energies in suggesting the ways in which CHEMKIN can better meet the needs of the advanced semiconductor processing industry. Prof. Mark Cappelli at Stanford University provided an initial vision, which established the technical direction for the multi-fluid formulation that is implemented in CHEMKIN-III.

CHEMKIN-III also has enhanced capabilities to handle a variety of pressure-dependent unimolecular-falloff and bimolecular chemically activated processes. Dr. Gregory Smith of SRI International and Dr. Jan Hessler of Argonne National Laboratory were instrumental in establishing the technical formulations and provided important suggestions on the software implementation.

Finally, we are grateful to our many colleagues at Sandia and elsewhere, who have provided numerous suggestions and have patiently worked with us as applications have migrated from CHEMKIN-II to CHEMKIN-III. In particular, we acknowledge the efforts of Drs. Michael Coltrin, Gregory Evans, Joseph Grcar, Pauline Ho, William Houf, Richard Larson, Andrew Lutz, Chris Moen, Harry Moffat, and Jong Shon.

CONTENTS

	<u>Page</u>
LIST OF FIGURES.....	6
LIST OF TABLES	6
NOMENCLATURE.....	7
I. INTRODUCTION.....	9
Background.....	9
Structure and Use of CHEMKIN.....	10
Example for a Single-Temperature Neutral Gas: Hydrogen Oxidation.....	11
Example for a Multi-Temperature Plasma.....	14
Transportability.....	16
Organization of this Report.....	16
II. THERMODYNAMICS AND CHEMICAL RATE EXPRESSIONS.....	18
Choice of Variables	18
Equation of State and Conversion Formulas	18
Standard-State Thermodynamic Properties.....	21
Chemical Reaction Rate Expressions.....	25
III. THE MECHANICS OF USING CHEMKIN.....	36
Structure of CHEMKIN	36
Job Control	38
IV. USING THE INTERPRETER.....	39
Element Data.....	39
Species Data.....	42
Thermodynamic Data	43
Reaction Mechanism Description.....	46
V. QUICK REFERENCE GUIDE TO THE GAS-PHASE SUBROUTINE LIBRARY	59
Mnemonics	59
VI. ALPHABETICAL LISTING OF THE GAS-PHASE SUBROUTINE LIBRARY WITH DETAILED DESCRIPTIONS OF THE CALL LISTS	72
VII. SAMPLE PROBLEM	136
1. UNIX Shell Script for Running the Sample Problem	137
2. Sample Input to the Interpreter.....	139
3. Output from the Interpreter for the Sample Input.....	140
4. Sample User's FORTRAN Code: CONP	142
5. Input to the Sample FORTRAN Code, CONP.....	146
6. Output form the Sample FORTRAN Code, CONP.....	147
7. Summary of VODE Math Library Usage.....	149
REFERENCES.....	153
APPENDIX A. STORAGE ALLOCATION FOR THE WORK ARRAYS.....	154

LIST OF FIGURES

	<u>Page</u>
Figure 1. Sample neutral reaction mechanism as read by the CHEMKIN Interpreter.	12
Figure 2. Sample plasma reaction mechanism as read by the CHEMKIN Interpreter.	15
Figure 3. Rate constant as a function of pressure at fixed temperature for the unimolecular fall-off reaction $\text{CH}_3 + \text{CH}_3 (+\text{M}) \rightleftharpoons \text{C}_2\text{H}_6 (+\text{M})$. The Troe and Lindemann forms are illustrated as are the low- and high-pressure limiting forms.	31
Figure 4. Energy versus reaction coordinate diagram that illustrates the competition between a three-body recombination reaction, $\text{CH}_3 + \text{CH}_3 (+\text{M}) \rightleftharpoons \text{C}_2\text{H}_6 (+\text{M})$, and a chemically activated bimolecular reaction, $\text{CH}_3 + \text{CH}_3 (+\text{M}) \rightleftharpoons \text{C}_2\text{H}_5 + \text{H} (+\text{M})$.	31
Figure 5. Rate constant as a function of pressure at fixed temperature for the chemically activated reaction $\text{CH}_3 + \text{CH}_3 (+\text{M}) \rightleftharpoons \text{C}_2\text{H}_5 + \text{H} (+\text{M})$. The SRI and Lindemann forms are illustrated as are the low- and high-pressure limiting forms.	32
Figure 6. Schematic diagram showing the structure of the CHEMKIN package and its relationship to an application code.	37
Figure 7. A sample UNIX command procedure showing the steps required to compile and run an application code using the CHEMKIN package.	38
Figure 8. Equivalent ways to describe element information	40
Figure 9. Equivalent ways to describe species information	42
Figure 10. Examples of thermodynamic data input.	45
Figure 11. Examples of reaction data	49
Figure 12. Examples of auxiliary information definitions	54

LIST OF TABLES

	<u>Page</u>
Table I. Summary of the Rules for Element Data	41
Table II. Summary of the Rules for Species Data	43
Table III. Summary of the Rules for Thermodynamic Data	44
Table IV. Summary of the Rules for Reaction Data	49
Table V. Summary of the Rules for Auxiliary Information Data	55

NOMENCLATURE

		<u>CGS Units</u>
a_{nk}	Coefficients of fits to thermodynamic data	depends on n
a_n^o	Standard state specific Helmholtz free energy of the k th species	ergs / g
\bar{a}	Mean specific Helmholtz free energy of a mixture	ergs / g
A_k^o	Standard state molar Helmholtz free energy of the k th species	ergs / mole
\bar{A}	Mean molar Helmholtz free energy of a mixture	ergs / mole
A_i	Pre-exponential factor in the rate constant of the i th reaction	depends on reaction
c_{pk}	Specific heat capacity at constant pressure of the k th species	ergs / (g K)
\bar{c}_p	Mean specific heat capacity at constant pressure	ergs / (g K)
C_{pk}^o	Standard state molar heat capacity at constant pressure of the k th species	ergs / (mole K)
C_{pk}	Molar heat capacity at constant pressure of the k th species	ergs / (mole K)
\bar{C}_p	Mean molar heat capacity at constant pressure	ergs / (mole K)
c_{vk}	Specific heat capacity at constant volume of the k th species	ergs / (g K)
\bar{c}_v	Mean specific heat capacity at constant volume	ergs / (g K)
C_{vk}	Molar heat capacity at constant volume of the k th species	ergs / (mole K)
\bar{C}_v	Mean molar heat capacity 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
\bar{g}	Mean specific Gibbs free energy of a mixture	ergs / g
G_k^o	Standard state molar Gibbs free energy for the k th species	ergs / mole
\bar{G}	Mean molar Gibbs free energy of a mixture	ergs / mole
h_k	Specific enthalpy of the k th species	ergs / g
\bar{h}	Mean specific enthalpy of a mixture	ergs / g
H_k^o	Standard state molar enthalpy of the k th species	ergs / mole
H_k	Molar enthalpy of the k th species	ergs / mole
\bar{H}	Mean molar enthalpy of a mixture	ergs / mole
i	Reaction index	
I	Total number of reactions	
k	Species index	
k_{fi}	Forward rate constant of the i th reaction	depends on reaction
k_{ri}	Reverse rate constant of the i th reaction	depends on reaction
K	Total number of species	
K_{ci}	Equilibrium constant in concentration units for the i th reaction	depends on reaction
K_{pi}	Equilibrium constant in pressure units for the i th reaction	depends on reaction
$[M]$	Total molar concentration of a mixture	moles / cm ³

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

CGS Units

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	[cal / (mole K)]
s_k^o	Standard state specific entropy of the k th species	ergs / (g K)
\bar{s}	Mean specific entropy of a mixture	ergs / (g K)
S_k^o	Standard state molar entropy of the k th species	ergs / (mole K)
S_k	Molar entropy of the k th species	ergs / (mole K)
\bar{S}	Mean molar entropy of a mixture	ergs / (mole K)
T	Temperature	K
u_k	Specific internal energy of the k th species	ergs / g
\bar{u}	Mean specific internal energy of a mixture	ergs / g
U_k	Molar internal energy of the k th species	ergs / mole
\bar{U}	Mean molar 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 the k th species	moles / cm ³
W_k	Molecular weight of the k th species	g / mole
\bar{W}	Mean molecular weight of a mixture	g / mole

GREEK

α_{ki}	Enhanced third-body efficiency of the k th species 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 coefficient of the k th species in the i th reaction, $\nu_{ki} = \nu'_{ki} - \nu''_{ki}$	
ν'_{ki}	Stoichiometric coefficient of the k th reactant species in the i th reaction	
ν''_{ki}	Stoichiometric coefficient 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-III: A FORTRAN CHEMICAL KINETICS PACKAGE FOR THE ANALYSIS OF GAS-PHASE CHEMICAL AND PLASMA 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 package¹ and the surface chemistry package, Surface CHEMKIN-III.^{2,3} 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 application codes and data between different sites. Often such exchanges are hampered by machine-dependent or problem-specific coding.

Background

CHEMKIN-III is a revised, generalized version of CHEMKIN. The original CHEMKIN⁵ was published in 1980. CHEMKIN II⁶ expanded these capabilities, with inclusion of 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 was required in the original CHEMKIN. Other added capabilities in CHEMKIN II included 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.

The current extension of CHEMKIN allows for treatment of non-equilibrium multi-fluid systems. Multi-fluid systems are systems in which the momentum or energy for one or more species in a gas mixture differs significantly from that of the bulk mixture. In a plasma system, for example, ions and

[†] Copyright © 1996, Sandia Corporation. The U. S. Government retains a nonexclusive license in this software as prescribed in AL 88-1 and AL 91-7. Export of this program may require a license from the United States Government.

electrons may be subject to electric fields that do not affect the neutral species transport. To track the motion of these species, a separate momentum equation must be solved that includes the force exerted by the electric field on the charged species. Similarly, electrons are subject to Joule-heating as they move along applied electric fields, requiring solution of a separate electron energy equation. Species momentum and energy equations can be derived as second and third moments about the Boltzmann equation, just as the species conservation equation results from the first moment of the Boltzmann equation. When all species are in thermal equilibrium and none are subject to special external forces, these equations can be summed over all species in the gas mixture to give the traditional equations of motion for a thermal system. In the CHEMKIN-III formulations, we allow for different species temperatures to control reaction dynamics and to determine species thermodynamic properties. For systems that are in thermal equilibrium, however, these relations collapse back to the original CHEMKIN formulations and should have no consequence to the CHEMKIN user.

In addition to the multi-fluid generalization, new capabilities have been added for the inclusion of global reaction kinetics, with the option of user-specified reaction orders and the allowance of non-integer stoichiometric coefficients. This capability is useful both for plasma systems and also for describing thermal systems where information about detailed kinetics is not well known.

The application of CHEMKIN to non-equilibrium plasma systems still requires, at this point, that the kinetics coefficients can be specified independent of the problem or application. For example, some assumption must be made *a priori* about the electron-energy distribution function (EEDF) when specifying electron-impact kinetics. In reality the EEDF will depend on the reactor conditions, such as the local electric field magnitude or the degree of dissociation of a molecular gas. These conditions are problem-dependent, such that including these effects requires close coupling between the kinetics-rate determination and the EEDF determination. While we foresee a need in future CHEMKIN development for treatment of fundamental reaction cross-section data, we believe there is still fairly wide opportunity for the application of problem-independent kinetics in plasma modeling. Such applications include plasma conditions where the EEDF is nearly Maxwellian, such as near-thermal atmospheric-pressure plasma jets, or very low-pressure, high-electron-density systems for microelectronics processing.

New capabilities in this version of CHEMKIN also include additional reaction-rate fit types for expanded user flexibility. These fits have been specifically implemented for the incorporation of electron-impact kinetics, which are occasionally difficult to fit with Arrhenius expressions.

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 a particular set of governing equations. The programming required is reduced since the user need only call CHEMKIN subroutines which define the terms in the governing equations that relate to equations of state, chemical production, and thermodynamics, and combine the result to define the problem of interest.

Next, the user runs the Interpreter, which first reads the symbolic description of the reaction mechanism and then extracts the appropriate thermodynamic information for the species involved from the Thermodynamic Database.⁷ The database has essentially the same format as that used by the original NASA complex chemical equilibrium code by Gordon and McBride.⁸ 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 the gas — pressure or density, temperature(s), and species composition.

Example for a Single-Temperature Neutral Gas: Hydrogen Oxidation

The input file to the CHEMKIN Interpreter for a hydrogen-oxidation process is shown in Fig. 1. First, the file specifies the elements and species that appear in the mechanism, and then includes the reaction mechanism description. 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 for selected species are specified in the line following that for a reaction which contains an arbitrary third body, M. Exclamation marks signify the beginning of comments and the remainder of the line is ignored.

```

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 Neutral Reaction Mechanism as Read by the CHEMKIN Interpreter.

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

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

where T is the temperature, ρ the mass density, \bar{c}_p the mean specific heat, H_k the molar 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, with 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 fills 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, **chem.bin**, and LOUT is the logical file number for printed diagnostic and error messages. The call to CKINDEX provides index information about the reaction mechanism: MM is the number of elements contained in the species, KK is the number of gas-phase species, II is the number of reactions, and NFIT is the number of coefficients in the thermodynamic fits. 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., $RHO = \rho$, $CPB = \bar{c}_p$, $HML = H_k$, and $WDOT = \dot{\omega}_k$.

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 a conservation equation for an arbitrary reaction mechanism.

Example for a Multi-Temperature Plasma

The input file to the CHEMKIN Interpreter for a chlorine-plasma excitation process is shown in Fig.2. As with the previous hydrogen-oxidation example, the file first specifies the elements and species that appear in the mechanism and then describes the reaction mechanism. Here, electrons must be specified both as an element and as a species. The elemental composition of a unipositive ion is that of the corresponding neutral minus one electron. This information is given in the species thermodynamic data and will be described further in the next section. As in the thermal system, three Arrhenius coefficients are used by default to describe reaction rates for electron-impact kinetics. The auxiliary keyword 'TDEP' on a line following the reaction statement indicates that the reaction rate is a function of the temperature of the species specified in the slashes following the TDEP keyword.

As shown in Fig. 2, most of the plasma reactions require some auxiliary information beyond the Arrhenius coefficients to distinguish the reaction description from the default thermal reactions. TDEP is one example of an auxiliary keyword that specifies the temperature dependence of the reaction. EXCI is used typically to indicate an excitation reaction. Such "reactions" are often included to allow calculation of inelastic energy loss rates for electrons, without requiring the user to include all excited states as new species in the reaction mechanism. The auxiliary information following the keyword EXCI represents the energy-loss per collision in electron volts. The keyword DUP is included to allow multiple occurrences of reaction statements that have different rate coefficients or different auxiliary information, but otherwise appear identical. This is frequently necessary in the specification of multiple excitation reactions from the same ground-state species. The use of auxiliary keywords is described in greater detail in Section IV.

Another important aspect of the plasma reactions shown in Fig. 2 is that they are all specified as irreversible reactions. This is in contrast to thermal reactions, which are usually reversible and reverse rates can be calculated directly from species thermodynamic properties. In the case of electron kinetics, the interactions between electrons and neutral species can be intrinsically irreversible. While detailed balancing may be appropriate for near-thermal plasmas, the use of CHEMKIN thermodynamics is not appropriate for determining reverse rates. In such cases, the user should explicitly supply reverse kinetic parameters, or specify a reverse path as an additional irreversible reaction.

```

ELEMENTS E CL END
SPECIES E CL+ CL2+ CL- CL* CL CL2 END
REACTIONS KELVIN MOLECULES
! reaction rates from Maxwellian EEDF
E + CL2 => E + CL2      2.5141E-02  -1.4443E+00  1.6650E+04
  TDEP/E/      !vibrational excitation
  EXCI/ 0.07/
  DUP
E + CL2 => CL- + CL      5.8901E-09  -2.5619E-01  1.5834E+04
  TDEP/E/      !dissociative attachment
E + CL2 => 2CL + E      1.5356E-06  -3.4642E-01  7.0850E+04
  TDEP/E/      !dissociation
E + CL2 => E + CL2      6.3477E-06  -5.3987E-01  1.3920E+05
  TDEP/E/      !electronic excitation
  EXCI/ 9.25/
  DUP
E + CL2 => CL2+ + 2E     1.1227E-04  -6.0067E-01  1.8070E+05
  TDEP/E/      !ionization
E + CL- => CL + 2E      3.1197E-06  -2.8757E-01  7.2058E+04
  TDEP/E/      !detachment
E + CL => E + CL*      1.2363E-05  -6.1356E-01  1.3297E+05
  TDEP/E/      !4s excitation
E + CL => E + CL      1.2363E-05  -6.1356E-01  1.3297E+05
  TDEP/E/      !4s excitation energy loss
  EXCI/ 9.55/
  DUP
E + CL => E + CL      9.4444E-05  -7.3093E-01  1.5413E+05
  TDEP/E/      !3d excitation
  EXCI/11.65/
  DUP
E + CL => CL+ + 2E      2.3736E-04  -7.0894E-01  1.8374E+05
  TDEP/E/      !ionization
E + CL* => CL+ + 2E     2.6471E-05  -4.3906E-01  6.3670E+04
  TDEP/E/      !Cl* ionization
CL- + CL2+ => CL + CL2   5.00E-08  0.0          0
CL- + CL+  => 2CL       5.00E-08  0.0          0
END

```

Figure 2. Sample Plasma Reaction Mechanism as Read by the CHEMKIN Interpreter.

Consider a simple form of the electron conservation equation for a closed system:

$$\frac{d[X_e]}{dt} = \dot{\omega}_e,$$

where $[X_e]$ is the electron molar concentration and $\dot{\omega}_e$ the electron molar production rate. The representation of this equation begins with CHEMKIN subroutine calls:

```

CALL CKINIT(LENIWK, LENRWK, LENCWK, LINKCK, LOUT, ICKWRK, RCKWRK, CCKWRK)
CALL CKINDX(ICKWRK, RCKWRK, MM, KK, II, NFIT)
CALL PKINDX(ICKWRK, KEL, KKION)
CALL CKKTFL(ICKWRK, KTFL)
CALL CKWC(T, C, ICKWRK, RCKWRK, WDOT)

```

As in the hydrogen-oxidation example, the first call is to the initialization subroutine CKINIT. CKINDX provides general chemistry indices, while PKINDX provides plasma-specific index information. In this case, we call PKINDX to get KEL, the location in the species array of the electron. In other words, there is no requirement for the species 'E' to be in any particular order in the mechanism species list. KKION is the number of positive and negative ions in the chemistry mechanism. The call to CKKTFL initializes the species temperature flag array in the CHEMKIN workspace. Without this call, it is assumed that all species share a common temperature, which is always the first entry in the temperature array passed to CHEMKIN in all subsequent calls. KTFL is a user-defined vector that defines the locations in the temperature array that correspond to each species temperature. This allows the application code to define a different number of temperatures in the system than the total number of species. For example, in a two-temperature plasma, where T(1) is the gas temperature and T(2) is the electron temperature, the user sets KTFL(KEL) = 2, and all other entries are set to '1'. Finally, in the call to CKWC, T is the temperature array, and C is the vector of species molar concentrations. The output variable, WDOT, is the $\dot{\omega}_k$ vector, where $\dot{\omega}_e$ is the KELth entry.

Transportability

The CHEMKIN-III package was developed on UNIX workstations and Cray/UNICOS 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. With the distribution of CHEMKIN, we provide a software utility called CHANGE,⁹ which is an interactive routine that prompts the user to toggle between single and double precision versions of source code prior to compilation.

Organization of this Report

Chapter II is a compendium of the 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. For example, Eq. (3) in Chapter II gives mean molecular weight in terms of the mass fractions. Subroutine CKMMWY would 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 useful in formulating a chemically reacting flow problem. For each expression/equation, the subroutine(s) that evaluates it is named to the right, below the equation number.

Choice of Variables

The formulation of any problem requires that a set of state 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(s), 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.

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

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 sometimes a convenient variable because the rate of progress of chemical reactions depends directly on the molar concentration of the reactants and products.

Equation of State and Conversion Formulas

The equation of state used is that of a ideal, multi-fluid gas. Comparing to earlier versions of CHEMKIN, we have now allowed for a temperature to be specified for each species, T_k . The reader who is more familiar with thermal systems may not immediately recognize these formulations as representing the traditional ideal gas law. However, the formulations do collapse to the more usual thermal relations in the case where all species temperatures, T_k , are equal to the gas temperature. The general equation of state is given by:

$$P = \sum_{k=1}^K [X_k] R T_k, \tag{1}$$

[CKPY, CKPX, CKPC]

while the mean mass density is defined by:

$$\rho = \sum_{k=1}^K [X_k] W_k . \quad (2)$$

[CKRHOY, CKRHOX, CKRHOC]

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

$$\bar{W} = \frac{1}{\sum_{k=1}^K Y_k / W_k} , \quad (3)$$

[CKMMWY]

$$\bar{W} = \sum_{k=1}^K X_k W_k , \quad (4)$$

[CKMMWX]

or

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

[CKMMWC]

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

Mass fraction to mole fraction:

$$X_k = \frac{Y_k}{W_k \sum_{j=1}^K Y_j / W_j} = \frac{Y_k \bar{W}}{W_k} \quad (6)$$

[CKYTX]

Mass fraction to molar concentration:

$$[X_k] = \frac{P(Y_k / W_k)}{R \sum_{j=1}^K Y_j T_j / W_j} \quad (7)$$

[CKYTCP]

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

[CKYTCR]

Mole fraction to mass fraction:

$$Y_k = \frac{X_k W_k}{\sum_{j=1}^K X_j W_j} = \frac{X_k W_k}{\bar{W}} \quad (9)$$

[CKXTY]

Mole fraction to molar concentration:

$$[X_k] = X_k \frac{P}{R \sum X_k T_k} \quad (10)$$

[CKXTCP]

$$[X_k] = X_k \frac{\rho}{W} \quad (11)$$

[CKXTCR]

Molar concentration to mass fraction:

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

[CKCTY]

Molar concentration to mole fraction:

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

[CKCTX]

Standard-State Thermodynamic Properties

CHEMKIN presumes that the standard-state thermodynamic properties are thermally “perfect”, in that they are only functions of temperature, and are given in terms of polynomial fits to the molar heat capacities at constant pressure:

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

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

$$H_k^o = \int_{298}^{T_k} C_{pk}^o dT + H_k^o(298), \quad (15)$$

so that

$$\frac{H_k^o}{RT_k} = \sum_{n=1}^N \frac{a_{nk} T_k^{(n-1)}}{n} + \frac{a_{N+1,k}}{T_k}, \quad (16)$$

where the parameter $a_{N+1,k}R$ represents the standard heat of formation at 298 K.

The standard-state molar entropy is written as

$$S_k^o = \int_{298}^{T_k} \frac{C_{pk}^o}{T} dT + S_k^o(298), \quad (17)$$

so that

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

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.⁸ In this case, seven coefficients are needed for each of two temperature ranges.* These fits take the following form, where the temperatures are in Kelvin:

$$\frac{C_{pk}^o}{R} = a_{1k} + a_{2k} T_k + a_{3k} T_k^2 + a_{4k} T_k^3 + a_{5k} T_k^4, \quad (19)$$

[CKCPOR]

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

$$\frac{H_k^o}{RT_k} = a_{1k} + \frac{a_{2k}}{2} T_k + \frac{a_{3k}}{3} T_k^2 + \frac{a_{4k}}{4} T_k^3 + \frac{a_{5k}}{5} T_k^4 + \frac{a_{6k}}{T_k}, \quad (20)$$

[CKHORT]

$$\frac{S_k^o}{R} = a_{1k} \ln T_k + a_{2k} T_k + \frac{a_{3k}}{2} T_k^2 + \frac{a_{4k}}{3} T_k^3 + \frac{a_{5k}}{4} T_k^4 + a_{7k}. \quad (21)$$

[CKSOR]

Other thermodynamic properties are easily given in terms of C_p^o , H^o , and S^o . The specific heat capacity at constant volume $C_{v_k}^o$ is

$$C_{v_k}^o = C_{p_k}^o - R, \quad (22)$$

[CKCVML]

the internal energy U_k^o is

$$U_k^o = H_k^o - RT_k, \quad (23)$$

[CKUML]

the standard-state Gibbs free energy G_k^o is

$$G_k^o = H_k^o - T_k S_k^o, \quad (24)$$

[CKGML]

and the standard-state Helmholtz free energy A_k^o is

$$A_k^o = U_k^o - T_k S_k^o. \quad (25)$$

[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

$$c_{p_k} = \frac{C_{p_k}}{W_k}, \quad (26)$$

[CKCPMS]

$$h_k = \frac{H_k}{W_k}, \quad (27)$$

[CKHMS]

$$s_k^o = \frac{S_k^o}{W_k}, \quad (28)$$

[CKSMS]

$$c_{v_k} = \frac{C_{v_k}}{W_k}, \quad (29)$$

[CKCVMS]

$$u_k = \frac{U_k}{W_k}, \quad (30)$$

[CKUMS]

$$g_k^o = \frac{G_k^o}{W_k}, \quad (31)$$

[CKGMS]

and

$$a_k^o = \frac{A_k^o}{W_k}. \quad (32)$$

[CKCAMS]

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

$$\bar{C}_p = \sum_{k=1}^K C_{p_k} X_k, \quad (33)$$

[CKCPBL]

$$\bar{c}_p = \sum_{k=1}^K c_{p_k} Y_k = \bar{C}_p / \bar{W}, \quad (34)$$

[CKCPBS]

$$\bar{C}_v = \sum_{k=1}^K C_{v_k} X_k, \quad (35)$$

[CKCVBL]

and

$$\bar{c}_v = \sum_{k=1}^K c_{v_k} Y_k = \bar{C}_v / \bar{W}; \quad (36)$$

[CKCVBS]

the enthalpies are

$$\bar{H} = \sum_{k=1}^K H_k X_k, \quad (37)$$

[CKHBML]

and

$$\bar{h} = \sum_{k=1}^K h_k Y_k = \bar{H} / \bar{W}; \quad (38)$$

[CKHBMS]

and the internal energies are

$$\bar{U} = \sum_{k=1}^K U_k X_k, \quad (39)$$

[CKUBML]

and

$$\bar{u} = \sum_{k=1}^K u_k Y_k = \bar{U} / \bar{W}. \quad (40)$$

[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. The entropy is then

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

Thus the mixture-averaged entropies are

$$\bar{S} = \sum_{k=1}^K (S_k^o - R \ln X_k - R \ln(P/P_{atm})) X_k, \quad (42)$$

[CKSBML]

and

$$\bar{s} = \bar{S}/\bar{W}. \quad (43)$$

[CKSBMS]

Similarly, the mixture-averaged Gibbs and Helmholtz free energies are

$$\bar{G} = \sum_{k=1}^K [H_k - T_k (S_k^o - R \ln X_k - R \ln(P/P_{atm}))] X_k, \quad (44)$$

[CKGBML]

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

[CKGBMS]

$$\bar{A} = \sum_{k=1}^K [U_k - T_k (S_k^o - R \ln X_k - R \ln(P/P_{atm}))] X_k, \quad (46)$$

[CKABML]

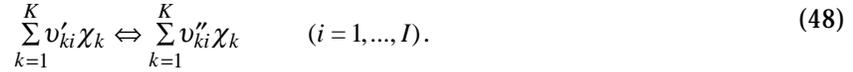
and

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

[CKABMS]

Chemical Reaction Rate Expressions

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



The stoichiometric coefficients v_{ki} are integer numbers and χ_k is the chemical symbol for the k th species. The superscript ' indicates forward stoichiometric coefficients, while '' indicates reverse stoichiometric coefficients. Normally, an elementary reaction involves only three or four species; hence the v_{ki} matrix is quite sparse for a large set of reactions. For non-elementary reactions, Eq. (48) also represents the reaction expression, but the stoichiometric coefficients may be non-integers. The use of real stoichiometric coefficients is a new option in CHEMKIN-III.

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

$$\dot{\omega}_k = \sum_{i=1}^I v_{ki} q_i \quad (k = 1, \dots, K), \quad (49)$$

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

where

$$v_{ki} = v''_{ki} - v'_{ki}. \quad (50)$$

[CKNU]

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

$$q_i = k_{f_i} \prod_{k=1}^K [X_k]^{v'_{ki}} - k_{r_i} \prod_{k=1}^K [X_k]^{v''_{ki}}, \quad (51)$$

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

where $[X_k]$ is the molar concentration of the k th species and k_{f_i} and k_{r_i} are the forward and reverse rate constants of the i th reaction. As indicated in Eq.(51), the rate-of-progress of a reaction is evaluated, by default, using the concentration of each reactant or product species raised to the power of its stoichiometric coefficient. Thus, the rate-of-progress of a reaction that includes species A with a coefficient of 2 will be second-order with respect to the concentration of A. Equation (51) is always valid when mass-action kinetics are obeyed, and when the mechanism is written in terms of elementary reactions. As it is often difficult to obtain elementary reaction kinetics, CHEMKIN-III includes the option for the user to define an arbitrary reaction order for a species in a reaction in place of the coefficients used in Eq.(33). This option is described further below.

The forward rate constants for the I reactions are generally assumed to have the following Arrhenius temperature dependence:

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

[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. In Eqs. (52-57), T refers to the gas temperature, unless auxiliary reaction information is provided to indicate that the reaction depends on a temperature associated with a particular species. Such information would be specified using the auxiliary keyword, TDEP, which is described further in Section IV. In the case where the TDEP keyword is included for reaction i , T represents the temperature of the species whose name follows the TDEP keyword.

In thermal systems, the reverse rate constants k_{r_i} are related to the forward rate constants through the equilibrium constants by

$$k_{r_i} = \frac{k_{f_i}}{K_{c_i}}. \quad (53)$$

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

$$K_{c_i} = K_{p_i} \left(\frac{P_{atm}}{RT} \right)^{\sum_{k=1}^K \nu_{k_i}}. \quad (54)$$

[CKEQYP, CKEQYR,
CKEQXP, CKEQXR, CKEQC]

The equilibrium constants K_{p_i} are obtained with the relationship

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

The Δ refers to the change that occurs in passing completely from reactants to products in the i th reaction; specifically,

$$\frac{\Delta S_i^o}{R} = \sum_{k=1}^K \nu_{k_i} \frac{S_k^o}{R}, \quad (56)$$

$$\frac{\Delta H_i^o}{RT} = \sum_{k=1}^K \nu_{k_i} \frac{H_k^0}{RT}. \quad (57)$$

For reactions involving electrons, the use of equilibrium constants to determine reverse rates is usually not appropriate. In some cases, detailed balancing on electron-driven reactions can be applied using the Saha equation (see, for example, Mitchner and Kruger¹⁰) that relates the ionization and electron-third-body recombination reactions to the species partition functions. While such relations can be used to calculate explicitly reverse rates from forward rates, they are not part of the built-in features of CHEMKIN. To avoid erroneous results, it is therefore required that all reactions involving electron species must either be specified as forward reactions only, or must include the reverse rate

* 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 that many users would rather use units of calories/mole for the activation energies even though ergs/mole are used elsewhere.

parameters explicitly stated using auxiliary keywords. The specification of reverse-rate parameters is described in more detail in Section IV.

Arbitrary Reaction Order

Often in real-world applications the elementary kinetics are not known. In some cases, an experimental measurement finds that the rate of reaction is proportional to the concentration of a species raised to a some arbitrary power (different from its stoichiometric coefficient). CHEMKIN-III allows the user to declare that the rate-of-progress of a reaction is proportional to the concentration of any species (regardless of whether that species even appears as a reactant or a product in the reaction) raised to any specified power. To modify the reaction order for the reaction in the forward or reverse direction, the user must declare the FORD or RORD auxiliary keywords, respectively, in the Interpreter input file. (These keywords are discussed in Chapter IV.)

When the reaction-order-dependence of reaction i is changed via the FORD or RORD keywords, the rate-of-progress variable q_i for the reaction is evaluated by :

$$q_i = k_{f_i} \prod_{k=1}^K [X_k]^{F_{ki}} - k_{r_i} \prod_{k=1}^K [X_k]^{R_{ki}}, \quad (58)$$

where F_{ki} is the reaction order specified through the FORD keyword and R_{ki} is the reaction order specified through the RORD keyword for species k . The default for species participating in reaction i is the normal mass-action kinetics values:

$$F_{ki} = v'_{ki} \quad (59)$$

$$R_{ki} = v''_{ki} \quad (60)$$

if an order-change parameter is not given for species k .

The user should exercise caution when specifying a change of reaction order, as such a change may produce unexpected and unphysical results in a kinetic simulation. For example, the user should consider the kinetics of the reverse reaction when changing reaction-orders for the forward reaction. Such a reaction may no longer satisfy microscopic reversibility. At equilibrium, elementary kinetics ensure that

$$k_{r_i} / k_{f_i} = \prod_{k=1}^K [X_k]^{v'_{ki}} / \prod_{k=1}^K [X_k]^{v''_{ki}} = \prod_{k=1}^K [X_k]^{v'_{ki} - v''_{ki}}. \quad (61)$$

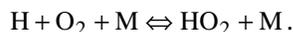
A reaction for which one has specified a change in reaction order will not have the proper equilibrium behavior unless

$$F_{ki} - R_{ki} = v'_{ki} - v''_{ki}, \quad (k = 1, \dots, K). \quad (62)$$

The user specifying F_{ki} may also wish to adjust R_{ki} such that Eq. (62) is satisfied; CHEMKIN-III does not do this automatically. Another alternative would be to simply specify that the reaction is irreversible.

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, such as



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:

$$q_i = \left(\sum_{k=1}^K (\alpha_{ki}) [X_k] \right) \left(k_{f_i} \prod_{k=1}^K [X_k]^{v'_{ki}} - k_{r_i} \prod_{k=1}^K [X_k]^{v''_{ki}} \right). \quad (63)$$

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

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

$$[M] = \sum_{k=1}^K [X_k]. \quad (64)$$

However, it is often the case that some species act more efficiently as third bodies than do others. The α_{ki} that differ from 1 must be specified by input to the CHEMKIN Interpreter.

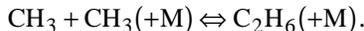
Pressure-Dependent Reactions

Under certain conditions, some reaction rate expressions depend on pressure as well as temperature. CHEMKIN provides for two kinds of such reactions: unimolecular/recombination fall-off reactions and chemically activated bimolecular reactions. Generally speaking, the rate for unimolecular/recombination fall-off reactions increases with increasing pressure, while the rate for chemically activated bimolecular reactions decreases with increasing pressure. In both cases, CHEMKIN makes available various expressions that blend smoothly between high- and low-pressure limiting rate expressions.

Unimolecular/Recombination Fall-off Reactions

As an example of a unimolecular/recombination fall-off reaction, consider methyl recombination. In the high-pressure limit, the appropriate description of the reaction is $\text{CH}_3 + \text{CH}_3 \rightleftharpoons \text{C}_2\text{H}_6$. In the low-pressure limit, a third-body collision is required to provide the energy necessary for the reaction to proceed, i.e., the appropriate description is $\text{CH}_3 + \text{CH}_3 + \text{M} \rightleftharpoons \text{C}_2\text{H}_6 + \text{M}$.

When such a reaction is at either limit, the (solely temperature-dependent) rate expressions discussed in the preceding 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,



There are several methods of representing the rate expressions in this fall-off region. The simplest one is due to Lindemann.¹¹ 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:

$$k_0 = A_0 T^{\beta_0} \exp(-E_0/R_c T), \quad (65)$$

$$k_\infty = A_\infty T^{\beta_\infty} \exp(-E_\infty/R_c T). \quad (66)$$

The rate constant at any pressure is then taken to be

$$k = k_\infty \left(\frac{P_r}{1 + P_r} \right) F, \quad (67)$$

where the reduced pressure P_r is given by

$$P_r = \frac{k_0 [M]}{k_\infty} \quad (68)$$

and $[M]$ is the concentration of the mixture, possibly including enhanced third-body efficiencies.[†] (For this example, note that the units for k are $\text{cm}^3/\text{moles}\cdot\text{sec}$, k_0 are $\text{cm}^6/\text{moles}^2\cdot\text{sec}$, and k_∞ are $\text{cm}^3/\text{moles}\cdot\text{sec}$.) If the F in Eq. (67) is unity, then this is the Lindemann form. The other descriptions involve more complex expressions for the function F .

In the Troe form,¹² F is given by

$$\log F = \left[1 + \left[\frac{\log P_r + c}{n - d(\log P_r + c)} \right]^2 \right]^{-1} \log F_{\text{cent}}. \quad (69)$$

[†] 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 $\text{CH}_3 + \text{CH}_3(+\text{N}_2) \rightleftharpoons \text{C}_2\text{H}_6(+\text{N}_2)$. In this case, the concentration of nitrogen $[\text{N}_2]$ would replace the total concentration in the mixture $[M]$ in these equations.

The constants in Eq. (69) are

$$c = -0.4 - 0.67 \log F_{\text{cent}}, \quad (70)$$

$$n = -0.75 - 1.27 \log F_{\text{cent}}, \quad (71)$$

$$d = 0.14, \quad (72)$$

and

$$F_{\text{cent}} = (1 - \alpha) \exp(-T/T^{***}) + \alpha \exp(T/T^*) + \exp(-T^{**}/T). \quad (73)$$

The four parameters α , 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.)

The approach taken at SRI International by Stewart et al.¹³ is in many ways similar to that taken by Troe, but the blending function F is approximated differently. Here, F is given by

$$F = d \left[a \exp\left(\frac{-b}{T}\right) + \exp\left(\frac{-T}{c}\right) \right]^X T^e \quad (74)$$

where

$$X = \frac{1}{1 + \log^2 P_r} \quad (75)$$

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. 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 a weak-collision efficiency factor, in the event that one wants to compute strong-collision rate parameters and correct them with such a factor.

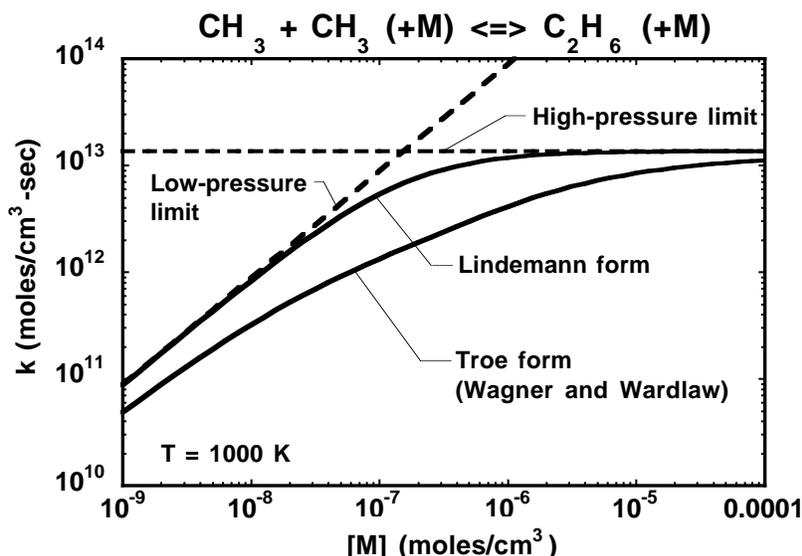


Figure 3. Rate constant as a function of pressure at fixed temperature for the unimolecular fall-off reaction $\text{CH}_3 + \text{CH}_3 (+\text{M}) \rightleftharpoons \text{C}_2\text{H}_6 (+\text{M})$. The Troe and Lindemann forms are illustrated as are the low- and high-pressure limiting forms.

Figure 3 illustrates the pressure dependence of rate expressions for the example reaction, $\text{CH}_3 + \text{CH}_3 (+\text{M}) \rightleftharpoons \text{C}_2\text{H}_6 (+\text{M})$, evaluated at a fixed temperature of 1000 K. Both the Lindemann and the Troe forms are shown, as well as the low- and high-pressure limits. The specific constants in fits to the Troe form ($A_0 = 1.135\text{E}36$, $\beta_0 = -5.246$, $E_0 = 1704.8\text{ cal/mole}$, $A_\infty = 6.22\text{E}16$, $\beta_\infty = -1.174$, $E_\infty = 635.8\text{ cal/mole}$, $\alpha = 0.405$, $T^{***} = 1120\text{ K}$, $T^* = 69.6\text{ K}$) are taken from Wagner and Wardlaw.¹⁴ For the relatively simple Lindemann case ($F=1$), the behavior of the limiting behavior is apparent. In the low-pressure limit, $[\text{M}] \rightarrow 0$, the denominator in Eq. (67) approaches unity and the rate expression becomes $k \rightarrow k_0[\text{M}]$. In the high-pressure limit, $[\text{M}] \rightarrow \infty$, the pressure-ratio factor approaches one and the rate expression becomes $k \rightarrow k_\infty$, i.e., a constant. For both the Troe and SRI forms, F approaches unity for both high and low pressures. Thus, all expressions recover the correct limiting behavior.

Chemically Activated Bimolecular Reactions

As an example of a chemically activated bimolecular reaction, consider $\text{CH}_3 + \text{CH}_3 (+\text{M}) \rightleftharpoons \text{C}_2\text{H}_5 + \text{H} (+\text{M})$. This reaction, which is endothermic, occurs through the same chemically activated C_2H_6^* adduct as does the recombination reaction $\text{CH}_3 + \text{CH}_3 (+\text{M}) \rightleftharpoons \text{C}_2\text{H}_6 (+\text{M})$. Figure 4 helps to illustrate the competition between these alternative channels using a reaction-energy diagram. As the pressure increases, deactivating collisions of C_2H_6^* with other molecules cause the rate coefficient for C_2H_6 formation to increase. At the same time, these deactivating collisions preclude the dissociation of C_2H_6^* into $\text{C}_2\text{H}_5 + \text{H}$, thus causing this rate coefficient to decrease with increasing pressure.

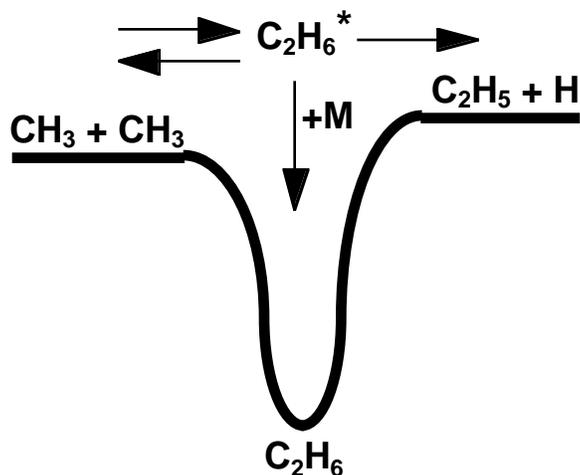


Figure 4. Energy versus reaction coordinate diagram that illustrates the competition between a three-body recombination reaction, $\text{CH}_3 + \text{CH}_3 (+\text{M}) \rightleftharpoons \text{C}_2\text{H}_6 (+\text{M})$, and a chemically activated bimolecular reaction, $\text{CH}_3 + \text{CH}_3 (+\text{M}) \rightleftharpoons \text{C}_2\text{H}_5 + \text{H} (+\text{M})$.

We assume the rate coefficient for a chemically activated bimolecular reaction to be described by the following function:

$$k = k_0 \left(\frac{1}{1 + P_r} \right)^F, \quad (76)$$

where $1/(1 + P_r)$ is analogous to the Lindemann form of Eq. (67). Note that in Eq.(76) k_0 is the pressure-independent factor, whereas in Eq. (67) it is k_∞ . The three choices for the F function are exactly the same as for the unimolecular fall-off reactions, i.e., the Lindemann ($F = 1$), Troe, or SRI forms.

Figure 5 illustrates the rate-expression behavior for the example reaction, $\text{CH}_3 + \text{CH}_3 (+\text{M}) \rightleftharpoons \text{C}_2\text{H}_6 (+\text{M})$. Both the Lindemann and the SRI formulations are shown, as well as the high- and low-pressure limiting cases. The specific constants for the SRI form ($A_0 = 10^{12.698}$, $\beta_0 = 0.099$, $E_0 = 10,600$ cal/mole, $A_\infty = 10^{-6.42}$, $\beta_\infty = 4.838$, $E_\infty = 7710$ cal/mole, $a = 1.641$, $b = 4334$, $c = 2725$) are taken from Stewart, Larson, and Golden¹⁵. (For this example, note that the units for k are $\text{cm}^3/\text{mole}\cdot\text{sec}$, k_0 are $\text{cm}^3/\text{mole}\cdot\text{sec}$, and k_∞ are $1/\text{sec}$.) The limiting cases are recognized easily from the behavior of Eq.(76). In the low-pressure limit, $[\text{M}] \rightarrow 0$, $P_r \ll 1$, causing the pressure-ratio factor in Eq.(76) to approach unity. Hence, $k \rightarrow k_0$, i.e., a pressure-independent function. In the high-pressure limit, $[\text{M}] \rightarrow \infty$, $P_r \gg 1$ and $k \rightarrow k_0/P_r = k_\infty/[\text{M}]$.

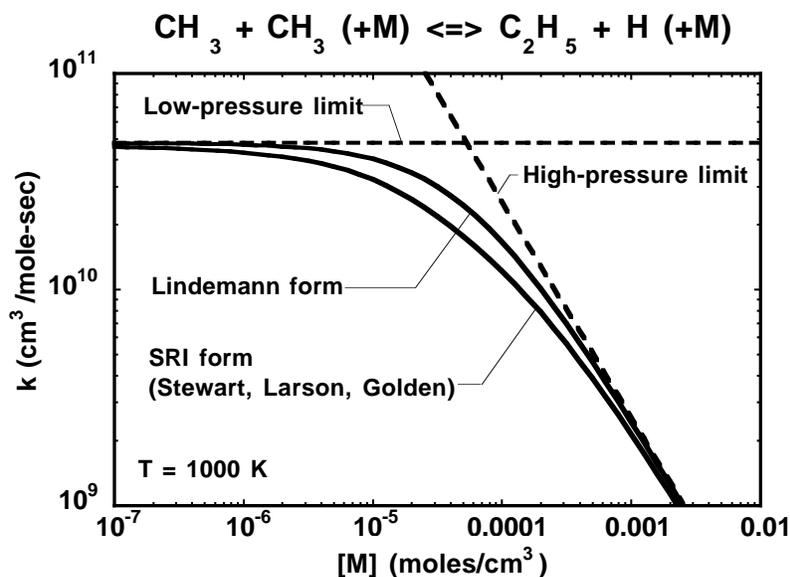


Figure 5. Rate constant as a function of pressure at fixed temperature for the chemically activated reaction $\text{CH}_3 + \text{CH}_3 (+\text{M}) \rightleftharpoons \text{C}_2\text{H}_6 (+\text{M})$. The SRI and Lindemann forms are illustrated as are the low- and high-pressure limiting forms.

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 Landau-Teller expression,

$$k_{f_i} = A_i \exp\left(\frac{B_i}{T^{1/3}} + \frac{C_i}{T^{2/3}}\right). \quad (77)$$

In CHEMKIN, we have provided the possibility to blend the Arrhenius expression with the Landau-Teller expression in the general expression, as follows:

$$k_{f_i} = A_i T^{\beta_i} \exp\left(\frac{-E_i}{R_c T} + \frac{B_i}{T^{1/3}} + \frac{C_i}{T^{2/3}}\right). \quad (78)$$

Clearly, by setting E_i and C_i to zero, the Arrhenius expression is recovered, and by setting B_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.

Other Allowable Rate Constant Fitting Options

In the accommodation of plasma reactions, we have included two new rate-expression forms in CHEMKIN-III. These fits require auxiliary keywords to specify additional parameters and to distinguish the expression from CHEMKIN's default of a modified Arrhenius form.

One form includes a polynomial fit to the logarithm of the temperature on which the reaction depends, as follows:

$$k_{f_i} = A_i T^{\beta_i} \exp\left(\frac{E_i}{T} + \sum_{n=1}^9 b_{n_i} (\ln T)^n\right). \quad (79)$$

This form is consistent with the rate-constant fits for electron-hydrogen and electron-helium processes in a publication by Janev, Langer, Evans, and Post,¹⁶ when the Arrhenius parameters β_i and E_i are zero. The user can specify this rate-constant fit option using the auxiliary keyword JAN.

A second form introduces a power series within the exponential of a modified Arrhenius expression, as follows:

$$k_{f_i} = A_i T^{\beta_i} \exp\left(\sum_{n=1}^4 \frac{b_n}{T^n}\right). \quad (80)$$

The user may specify this rate-constant expression using the auxiliary keyword FIT1. The use of auxiliary keywords are described in more detail in the next section.

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:

$$\dot{\omega}_k = \dot{C}_k - \dot{D}_k, \quad (81)$$

[CKCDYP, CKCDYR,
CKCDXP, CKCDXR,
CKCDC]

where, for non-three-body reactions,

$$\dot{C}_k = \sum_{i=1}^I v'_{ki} k_{r_i} \prod_{j=1}^K [X_j]^{v''_{ji}} + \sum_{i=1}^I v''_{ki} k_{f_i} \prod_{j=1}^K [X_j]^{v'_{ji}} \quad (82)$$

and

$$\dot{D}_k = \sum_{i=1}^I v'_{ki} k_{f_i} \prod_{j=1}^K [X_j]^{v'_{ji}} + \sum_{i=1}^I v''_{ki} k_{r_i} \prod_{j=1}^K [X_j]^{v''_{ji}}. \quad (83)$$

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 characteristic time for the destruction rate, i.e.,

$$\dot{\omega}_k = \dot{C}_k - \frac{[X_k]}{\tau_k}. \quad (84)$$

[CKCTYP, CKCTYR,
CKCTXP, CKCTXR, CKCTC]

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

$$\tau_k = \frac{[X_k]}{\dot{D}_k}. \quad (85)$$

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

$$\tau_k = \frac{[X_k]}{\dot{D}_k + \varepsilon}, \quad (86)$$

[CKCTYP, CKCTYR,
CKCTXP, CKCTXR, CKCTC]

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

In some numerical solution algorithms for chemically reacting flow, it is a significant computational savings to separate the temperature-dependent part of the rate expressions (i.e. the rate constants in most cases) from the concentration-dependent contribution. In particular, evaluation of Jacobian matrix elements through perturbation of solution variables often relies on numerous function evaluations and hence numerous calls to CHEMKIN to evaluate rate expressions. The temperature-dependent portion of the rate expression contains an exponential, which is computationally expensive to evaluate. When the temperature variable is not being perturbed, it is unnecessary to repeat this evaluation. To this end we have provided additional subroutines that either provide the temperature-dependent rate coefficients or, given the rate coefficients, return the species' net rates of production. The subroutine for evaluating the temperature-dependent rate constant for each reaction is called CKKFRT, while the subroutine that takes the rate constant as input and returns the species net rates of production is called CKWYPK. The use of these subroutines is described in more detail in Sections V and VI.

* 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. 6. 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 a binary file called **chem.bin**. This file must be opened in the user's code and a logical file number **LINCK** must be specified and passed to the CHEMKIN initialization routine (so that the file 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 (**therm.dat**).⁷ The input to the Interpreter is read on FORTRAN's standard input unit and printed output is directed to FORTRAN's standard output unit. User-specified filenames can thereby be directed to and from the Interpreter on the Interpreter command line for UNIX systems. 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 opened as **therm.dat** on FORTRAN unit 17. This 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 the input file; these data can replace or add to those 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 storage arrays (one integer, one floating point, and one character data type*) and then call the initialization subroutine CKINIT to create the storage arrays from the Linking File.[†] One or more of these arrays is required input to nearly every other subroutine in the CHEMKIN Package.

[‡] It is also possible to generate an ASCII Linking File rather than the default binary file, by performing a minor modification of the CHEMKIN -III Interpreter and Library source code using the CHANGE⁹ utility.

* 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.

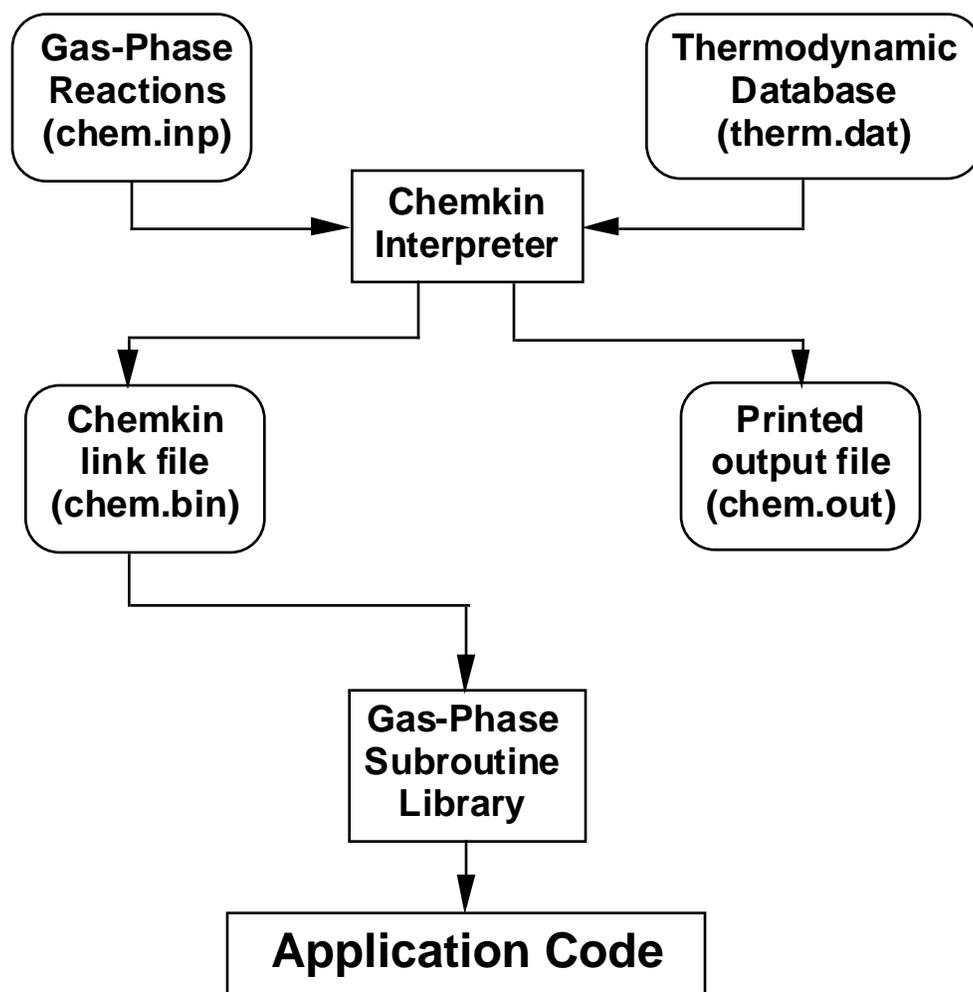


Figure 6. Schematic diagram showing the structure of the CHEMKIN package and its relationship to an application code.

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. With some FORTRAN compilers, it may be necessary to use an object library format* for the subroutine package, to assure 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. On unix systems, libraries are created with the 'ar' command.

Job Control

By example we show here how to compile and run CHEMKIN. Figure7 is an annotated UNIX command procedure that outlines the important steps. Even though the example is specific to UNIX systems, the same functionality must be invoked on any computer system.

UNIX Commands			Meaning
f77	-o chem.exe	ckinterp.f	Compile and link the interpreter using the f77 compiler. The executable is named chem.exe.
chem.exe	< chem.inp	>chem.out	Execute the interpreter. The interpreter will try to open the mechanism file chem.inp , as well as the Thermodynamic Database, therm.dat . These files are opened as units 5 and 17, respectively. The interpreter creates the binary linking file chem.bin and the ASCII output file chem.out . These files use units 25 and 6, respectively.
f77	-c	cklib.f	Compile the CHEMKIN Gas-Phase Subroutine library.
f77	-c	sample.f	Compile the user's FORTRAN program.
f77	-o sample.exe	sample.o cklib.o	Link the user's FORTRAN program with the CHEMKIN Gas-Phase Subroutine Library.
sample.exe	< sample.inp	> sample.out	Execute the user's program, reading 'sample.inp' as unit 5 for the user's input and creating 'sample.out' on unit 6 for the user's output.

Figure 7. A sample UNIX command procedure showing the steps required to compile and run an application code using the CHEMKIN package.

IV. USING THE INTERPRETER

The interpreter is used to read (from the Interpreter Input File) a symbolic description of an elementary chemical reaction mechanism and create a Linking File (**chem.bin**) 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 File includes information on elements, species, thermodynamic data, and the reaction mechanism. Input information in the Interpreter Input File is given in 80-column 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 Interpreter Input File and/or from a Thermodynamic Database (file **therm.dat**). 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 a compatible format to that used in the original NASA Chemical Equilibrium code by Gordon and McBride.⁸

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 reaction 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 8 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.

```

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

```

Figure 8. Equivalent Ways to Describe Element Information

* 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

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 one or more 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 to be a comment and will be ignored. Blank lines are ignored. Figure 9 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

SPEC H2
spec O2
```

Figure 9. 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 **therm.dat**) and/or read from the Interpreter Input File. 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 McBride⁸ 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 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.⁷

Table III. Summary of the Rules for Thermodynamic 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 to 18
	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).

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 10 shows some examples of thermodynamic property input. In these 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 - 31 of line 2 in Table III, which in this example is in the Thermodynamic Database.

```

THERMO
OH          1212860  1H  1          G  0300.00  5000.00  1000.00      1
0.02882730E+02 0.10139743E-02-0.02276877E-05 0.02174683E-09-0.05126305E-14  2
0.03886888E+05 0.05595712E+02 0.03637266E+02 0.01850910E-02-0.16761646E-05  3
0.02387202E-07-0.08431442E-11 0.03606781E+05 0.13588605E+01  4
OH+         1212860  1H  1E -1      G  0300.00  5000.00      1
0.02719058E+02 0.15085714E-02-0.05029369E-05 0.08261951E-09-0.04947452E-13  2
0.15763414E+06 0.06234536E+02 0.03326978E+02 0.13457859E-02-0.03777167E-04  3
0.04687749E-07-0.01780982E-10 0.15740294E+06 0.02744042E+02  4
OH-         1212860  1H  1E  1      G  0300.00  5000.00      1
0.02846204E+02 0.10418347E-02-0.02416850E-05 0.02483215E-09-0.07775605E-14  2
-0.01807280E+06 0.04422712E+02 0.03390037E+02 0.07922381E-02-0.01943429E-04  3
0.02001769E-07-0.05702087E-11-0.01830493E+06 0.12498923E+01  4
END

```

Figure 10. Examples of Thermodynamic Data Input.

* In the original Chemkin, the thermodynamic data preceded the species data.

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

Case 1: All thermodynamic data from database file

1. The database file is **therm.dat**.
2. No THERMO data required as input.

Case 2: Thermodynamic data from database and input files

1. The database file is **therm.dat**
2. Include the following lines in the Interpreter Input File, after the species data:
THERMO
Data in Table III format (lines 3 - 6 repeated) for each species not in the database or to override species in database
END

Case 3: All thermodynamic data from input file

1. No **therm.dat** file required
2. Include the following lines in the Interpreter Input File, after the species data:
THERMO ALL
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,¹¹ Troe,¹² or SRI* fall-off formulation[†]; it may involve a photon; it may depend on a species temperature other than that of the bulk gas.

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, KELVINS, or EVOLTS for E_j , and/or MOLES or MOLECULES for A_j . If MOLECULES is specified, then the units for A_j are cm-molecules-sec-K. If units are not specified, A_j and E_j must be in cm-mole-sec-K and cal/mole, respectively. Note that T is always in Kelvin. 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.¹³, 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 is 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.

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 or real 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. Note: non-integer coefficients are allowed in CHEMKIN-III, but the element balance in the reaction must still be maintained.

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 be 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 the 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, (+H₂O) 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 the auxiliary information line (described below). Although this information is not used internally in the CHEMKIN routines, it is available to the user through a subroutine call.
- 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 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 in 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 11, 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 + H                       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(*)                            1.3A5  0  0
0.5H2 + 0.5O2 = OH                       ! example of real coefficients
END                                         ! END statement is optional;
                                           ! <eof> condition is equivalent

```

Figure 11. 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 between 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, unless the continuation character '&' is included at the end of the line
 5. Continuation to the following line is indicated by an '&' at the end of the reaction 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 six reactants or six 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).

Neutral Third Body and Pressure Dependent Reaction Parameters

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. (63)]. 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 parentheses, say (+H₂O), then one or more auxiliary information lines must follow to define the pressure-dependence parameters. The Arrhenius coefficients on the reaction line are for the high-pressure limit (A_∞ , β_∞ , and E_∞) for unimolecular fall-off reactions and represent the low-pressure limit (A_o , β_o , and E_o) for chemically activated bimolecular reactions (see the description of these reaction types in Section II. For all pressure-dependent reactions an auxiliary information line must follow to specify either the low-pressure limit Arrhenius parameters (for fall-off reactions) or the high-pressure limit Arrhenius parameters (for chemically activated reactions). For fall-off reactions, the keyword LOW must appear on the auxiliary information line, with three rate parameters A_o , β_o , and E_o [Eq. (65)]. For chemically activated bimolecular reactions, the keyword HIGH must appear on the auxiliary information line, with the three rate parameters A_∞ , β_∞ , and E_∞ [Eq. (66)]. There are then three possible interpretations of the pressure-dependent reaction:

- 1) To define the Lindemann¹¹ formulation of a pressure-dependent reaction, no additional parameters are defined.
- 2) To define a Troe¹² pressure-dependent reaction, in addition to the LOW or HIGH parameters, the keyword TROE followed by three or four parameters must be included in the following order: a , T^{***} , T^* , and T^{**} [Eq. (73)]. The fourth parameter is optional and if omitted, the last term in Eq. (73) is not used.
- 3) To define an SRI pressure-dependent reaction*, in addition to the LOW or HIGH parameters, the keyword SRI followed by three or five parameters must be included in the following order: a , b , c , d , and e [Eq. (74)]. The fourth and fifth parameters are options. If only the first three are stated, then by default $d = 1$ and $e = 0$.

* SRI refers to the formulation of Stewart et al.¹³, who are at SRI International, Menlo Park, CA.

Landau-Teller reactions

To specify Landau-Teller parameters, the keyword LT must be followed by two parameters — the coefficients B_i and C_i from Eq. (77). 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 REV (see below under “Reverse Rate Parameters”), 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.

Optional Rate Fit Expressions

To specify the optional rate-constant fit expression described by Eq.(79), the keyword JAN must be followed by nine parameters — the coefficients b_n from Eq.(79). The Arrhenius parameters A_i , β_i , and E_i are taken from the numbers specified on the reaction line itself. If fewer than nine parameters are required for the fit, the user must provide zeros for the remainder of the parameters in the auxiliary parameter list.

To specify the optional rate-constant fit expression described by Eq.(80), the keyword FIT1 must be followed by four parameters — the coefficients b_n from Eq.(80). Again, the Arrhenius parameters A_i , β_i , and E_i are read from the reaction line.

Radiation Wavelength Parameter

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.

Species Temperature Dependence

When solving multi-fluid problems that involve multiple temperatures (for example, electron temperature and neutral gas temperature), the auxiliary information data may follow the reaction to specify the species on whose temperature the reaction depends. Here the species name follows the auxiliary keyword TDEP. This option causes the reaction rate constant to be evaluated using the specified species temperature and the rate parameters given in the reaction data. In the case when there is more than one temperature defined in the system, the application code must call the CHEMKIN subroutine CKKTFLL to indicate which temperature in the temperature array corresponds to each species.

Energy Loss Parameter

Auxiliary data may be used to specify the energy loss per reaction event by specifying the keyword EXCI, followed by the value of the energy loss per event in units of electron volts. This option overrides the calculation of energy loss from the change in enthalpy determined by the reaction description and the thermodynamic data of the reactants and products. The option is useful in describing electron-impact excitation reactions, for example, where the user does not wish to keep track

of the excited-species density, but wants to include the energy loss to the electrons due to the excitation process.

Plasma Momentum-Transfer Collision Frequency Options

To indicate that the reaction parameters describe the momentum-transfer collision frequency for electrons, the auxiliary keyword MOME may be used. This keyword requires no supplemental data, but changes the treatment of the reaction-rate coefficients. The option causes the reaction to be flagged as an electron momentum-transfer reaction, and assumes that the reaction rate constant is in units of $\text{cm}^3/\text{mole-s}$ or $\text{cm}^3/\text{molecule-s}$, depending on the units specified in the REACTION statement. These reactions are treated as special cases when CHEMKIN subroutines evaluate reaction rates-of-progress, as described in Section II.

Auxiliary data may also be used to flag a reaction as representing collision cross-section information for the determination of ion momentum-transfer collision frequencies in a plasma simulation. Here the auxiliary keyword is XSMI, and no auxiliary parameters are required. In this case the units of the evaluated rate-constant are assumed to be in cm^2 , and are left as such when CHEMKIN subroutines evaluate rates of progress for other reactions. Again, the use of this option is described in more detail in Section II.

Reverse Rate Parameters

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. This option overrides the reverse rates that would be normally computed by satisfying microscopic reversibility through the equilibrium constant, Eq. (53).

Reaction Order Parameters

Auxiliary data may be included to override the reaction order for a species, using the auxiliary keywords FORD or RORD, for forward and reverse reaction descriptions, respectively. Each occurrence of these keywords must be followed by the species name and the new reaction order. This option overrides the values of ν'_{ki} and ν''_{ki} in Eq. (51), for the species included in the auxiliary data.

Reaction Units

It is sometimes convenient to specify the units for a particular reaction rate fit that may differ from the default units specified for other reaction expressions in the chemistry mechanism. In this case, the user employs the auxiliary keyword UNITS. This keyword must be followed by one or more of the following unit descriptors: MOLE(CULE), CAL, KCAL, JOUL, KJOU, KELV(IN), or EVOL(TS), where the letters in parentheses are optional. The inclusion of MOLE(CULE) would indicate that the reaction rate expression is in units of $\text{molecules}/\text{cm}^3$ rather than moles/cm^3 . The remaining unit descriptors specify the units of the energy E_i in the rate expression. Note that the units of T in the rate expression are always in Kelvin.

Duplicate Reaction Descriptions

It sometime 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.

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 Figure12. 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 **chem.bin** 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, with 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/      ! enhanced third-body efficiencies

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

! The following reactions allow plasma kinetics descriptions
E + E + AR+ <=> AR + E 1.414E+39  -4.500  0.00      ! Mansbach & Keck
  TDEP/E/  REV/6.807E+31  -3.0  364218./ !electron temperature dependence

E + AR => AR + E       4.9E-7  0.162  8.7634E3
  TDEP/E/  MOMO      !Momentum-transfer collision frequency

AR+ + AR => AR+ + AR   1.E-16  0.0  0.0      !units of cm^2
  XSMI      !Ion momentum-transfer collision cross-section

E + AR => AR + E       2.235E16  0.0  3.47E5
  TDEP/E/  EXCI/11.60/ ! metastable excitation reaction
  DUP

END                      !END line is optional

```

Figure 12. 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.
 11. In ion momentum-transfer collision cross-section reactions there must be exactly two reactant species, one of which must be an ion.
 12. In electron momentum-transfer collision frequency reactions, there must be exactly two reactant species, one of which must be the electron.
-

Error Checks

The Interpreter checks each input line for proper syntax and writes 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 listed below:

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 corresponding dimension in the Interpreter.

Species Data

If a species is declared twice, a diagnostic message is printed, but 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 corresponding dimension in the Interpreter.

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 appear as reactants and/or products.

HV declared as a reactant and as a product.

There are more reactions than the corresponding dimension in the Interpreter.

There are more than six reactants or six products.

Auxiliary Data

There is an unknown or misspelled keyword or enhanced third-body species name.

Parameters for a keyword are not enclosed in slashes.

The wrong number of parameters appear for a keyword.

There are duplicate keywords for a reaction.

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

LOW or HIGH is not found after a pressure-dependent reaction.

TROE and SRI are both found.

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

LT or REV are given for a fall-off reaction.

There are more than ten enhanced third bodies.

There are more than or less than two reactants specified with XSMI or MOME keywords.

An ionic species is not specified as a reactant with the XSMI keyword.

The electron is not a reactant when using the MOME keyword.

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. Detailed descriptions of the subroutines are included alphabetically in Chapter VI.

Mnemonics

There are some good rules of thumb that explain the subroutine naming conventions. All subroutine names (with the exception of PKINDX) 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 \bar{C}_p) in a thermodynamic property subroutine name indicates that it returns mean mixture 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 input and output 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. Subroutines whose call lists have changed since the release of CHEMKIN II are indicated by a double asterisk.

1. INITIALIZATION

SUBROUTINE CKINDX (ICKWRK, RCKWRK, MM, KK, II, NFIT)*

Returns a group of indices defining the size of the particular reaction mechanism

SUBROUTINE CKINIT (LENICK, LENRCK, LENCCK, LINC, LOUT, ICKWRK, RCKWRK, CCKWRK, IFLAG)**

Reads the linkfile and creates the internal work arrays ICKWRK, RCKWRK and CCKWRK. CKINIT must be called before any other CHEMKIN subroutine can be used, as the work arrays must be available as their input.

SUBROUTINE CKKTFL (ICKWRK, KTFL)

Allows the user to assign a location in the temperature array to use for each gas-phase species.

SUBROUTINE CKLEN (LINC, LOUT, LENI, LENR, LENC, IFLAG)

Returns the lengths required for work arrays.

SUBROUTINE CKPNT (LSAVE, LOUT, NPOINT, VERS, PREC, LENI, LENR, LENC, KERR)

Reads from a file information about a CHEMKIN linkfile, and pointers for work arrays.

SUBROUTINE CKSAVE (LOUT, LSAVE, ICKWRK, RCKWRK, CCKWRK)

Writes to a binary file information about a CHEMKIN linkfile, pointers for the CHEMKIN Library, and CHEMKIN work arrays.

SUBROUTINE PKINDX (ICKWRK, KELECT, KKION)

Returns plasma indices for the particular reaction mechanism.

2. INFORMATION ABOUT ELEMENTS

SUBROUTINE CKAWT (ICKWRK, RCKWRK, AWT)

Returns the atomic weights of the elements

SUBROUTINE CKCOMP (IST, IRAY, II, I)*

Returns the index of an element of a reference character string array which corresponds to a character string; leading and trailing blanks are ignored.

SUBROUTINE CKSYME (CCKWRK, LOUT, ENAME, KERR)*

Returns the character strings of element names.

3. INFORMATION ABOUT SPECIES

SUBROUTINE CKCHRG (ICKWRK, RCKWRK, KCHARG)

Returns the electronic charges of the species.

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

SUBROUTINE CKION (ICKWRK, KION)
Returns the ion species indices

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

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

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

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

4. INFORMATION ABOUT REACTIONS

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

SUBROUTINE CKFAL (NDIM, ICKWRK, RCKWRK, IFOP, IFLO, KFAL, FPAR)
Returns a set of flags indicating whether a reaction has pressure-dependent behavior and an array of parameters.

SUBROUTINE CKHRX (I, HML, ICKWRK, RCKWRK, HRXI)
Returns the molar heat of reaction I.

SUBROUTINE CKIEXC (ICKWRK, RCKWRK, IEXC, EEXC)
Returns a set of flags indicating whether the reactions are excitation reactions and, if so, the energy loss per event in eV.

SUBROUTINE CKIMOM (ICKWRK, IMOM)
Returns a set of flags indicating whether the reactions are electron momentum-transfer collision frequencies and, if so, the index of the species with which the electron collides.

SUBROUTINE CKINU (I, NDIM, ICKWRK, RCKWRK, NSPEC, KI, NU)
Returns a count of species in a reaction, and their indices and stoichiometric coefficients; see Eq. (50).

SUBROUTINE CKIORD (IDIM, KDIM, ICKWRK, RCKWRK, NFORD, IFORD, FORD, NRORD, IRORD, RORD)
Returns the count and indices of reactions with modified species order and the order values for the species.

SUBROUTINE CKIRNU (IDIM, NDIM, ICKWRK, RCKWRK, NIRNU, IRNU, NSPEC,
KI, RNU)

Returns the count and indices of reactions with real stoichiometric coefficients, counts of species in the reactions, and the species indices and coefficients; see Eq. (50).

SUBROUTINE CKITDE (ICKWRK, RCKWRK, ITDE)

Returns a set of flags indicating whether the reactions are non-thermal, and if so, returns the index of the species on which the reaction depends.

SUBROUTINE CKITR (ICKWRK, RCKWRK, ITHB, IREV)

Returns a set of flags indicating whether the reactions are reversible or whether they contain arbitrary third bodies

SUBROUTINE CKIXSM (ICKWRK, IXSM, IXSK)

Returns a set of flags indicating whether the reactions are ion momentum-transfer cross sections.

SUBROUTINE CKNU (KDIM, ICKWRK, RCKWRK, NUKI)

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

SUBROUTINE CKNUF (KDIM, ICKWRK, RCKWRK, NUKI)

Returns the forward stoichiometric coefficients for reactions; by definition, reactants' coefficients are negative; see Eq. (50). Contrast this subroutine with subroutine CKNU, which returns the net stoichiometric coefficients for a reaction.

SUBROUTINE CKRAEX (I, RCKWRK, RA)*

Get/put the Pre-exponential coefficient of the Ith reaction

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.

SUBROUTINE CKTHB (KDIM, ICKWRK, RCKWRK, AKI)

Returns matrix of enhanced third body coefficients; see Eq. (63).

SUBROUTINE CKWL (ICKWRK, RCKWRK, WL)

Returns a set of flags providing information on the wave length of photon radiation

5. GAS CONSTANTS AND UNITS

SUBROUTINE CKRP (ICKWRK, RCKWRK, RU, RUC, PA)

Returns universal gas constants and the pressure of one standard atmosphere

6. EQUATION OF STATE AND MOLE-MASS CONVERSION

SUBROUTINE CKMMWC (C, ICKWRK, RCKWRK, WTM)

Returns the mean molecular weight of the gas mixture given molar concentrations; see Eq. (5).

SUBROUTINE CKMMWX (X, ICKWRK, RCKWRK, WTM)

Returns the mean molecular weight of the gas mixture given mole fractions; see Eq. (4).

SUBROUTINE CKMMWY (Y, ICKWRK, RCKWRK, WTM)

Returns the mean molecular weight of the gas mixture given mass fractions; see Eq. (3).

SUBROUTINE CKPC (RHO, T, C, ICKWRK, RCKWRK, P)

Returns the pressure of the gas mixture given mass density, temperature(s) and molar concentrations; see Eq. (1).

SUBROUTINE CKPX (RHO, T, X, ICKWRK, RCKWRK, P)

Returns the pressure of the gas mixture given mass density, temperature(s) and mole fractions; see Eq. (1).

SUBROUTINE CKPY (RHO, T, Y, ICKWRK, RCKWRK, P)

Returns the pressure of the gas mixture given mass density, temperature(s) and mass fractions; see Eq. (1).

SUBROUTINE CKRHOC (P, T, C, ICKWRK, RCKWRK, RHO)

Returns the mass density of the gas mixture given pressure, temperature(s) and molar concentrations; see Eq. (2).

SUBROUTINE CKRHOX (P, T, X, ICKWRK, RCKWRK, RHO)

Returns the mass density of the gas mixture given pressure, temperature(s) and mole fractions; see Eq. (2).

SUBROUTINE CKRHOY (P, T, Y, ICKWRK, RCKWRK, RHO)

Returns the mass density of the gas mixture given pressure, temperature(s) and mass fractions; see Eq. (2).

SUBROUTINE CKCTX (C, ICKWRK, RCKWRK, X)

Returns the mole fractions given molar concentrations; see Eq. (13).

SUBROUTINE CKCTY (C, ICKWRK, RCKWRK, Y)

Returns the mass fractions given molar concentrations; see Eq. (12).

SUBROUTINE CKXTCP (P, T, X, ICKWRK, RCKWRK, C)

Returns the molar concentrations given pressure, temperature(s) and mole fractions; see Eq. (10).

SUBROUTINE CKXTCR (RHO, T, X, ICKWRK, RCKWRK, C)

Returns the molar concentrations given mass density, temperature(s), and mole fractions; see Eq. (11).

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

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

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

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

7. THERMODYNAMIC PROPERTIES (NONDIMENSIONAL)

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

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

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

SUBROUTINE CKMXTP (ICKWRK, MAXTP)
Returns the maximum number of temperatures used in fitting the
thermodynamic properties of the species.

SUBROUTINE CKRHEX (K, RCKWRK, A6)
Returns an array of the sixth thermodynamic polynomial
coefficients for a species, or changes their value,
depending on the sign of K.

SUBROUTINE CKSMH (T, ICKWRK, RCKWRK, SMH)*
Returns the array of entropies minus enthalpies for species.
It is normally not called directly by the user.

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

8. THERMODYNAMIC PROPERTIES (MASS UNITS)

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

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

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

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

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

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

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

9. THERMODYNAMIC PROPERTIES (MOLAR UNITS)

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

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

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

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

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

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

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

10. MEAN THERMODYNAMIC PROPERTIES (MASS UNITS)

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

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

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

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

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

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

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

11. MEAN THERMODYNAMIC PROPERTIES (MOLAR UNITS)

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

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

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

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

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

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

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

12. CHEMICAL PRODUCTION RATES

SUBROUTINE CKCDC (T, C, ICKWRK, RCKWRK, CDOT, DDOT)
Returns the molar creation and destruction rates of the species given temperature(s) and molar concentrations; see Eq. (81).

SUBROUTINE CKCDXP (P, T, X, ICKWRK, RCKWRK, CDOT, DDOT)
Returns the molar creation and destruction rates of the species given pressure, temperature(s) and mole fractions; see Eq. (81).

SUBROUTINE CKCDXR (RHO, T, X, ICKWRK, RCKWRK, CDOT, DDOT)
Returns the molar creation and destruction rates of the species given mass density, temperature(s) and mole fractions; see Eq. (81).

SUBROUTINE CKCDYP (P, T, Y, ICKWRK, RCKWRK, CDOT, DDOT)
Returns the molar creation and destruction rates of the species given mass density, temperature(s) and mass fractions; see Eq. (81).

SUBROUTINE CKCDYR (RHO, T, Y, ICKWRK, RCKWRK, CDOT, DDOT)
Returns the molar creation and destruction rates of the species given mass density, temperature(s) and mass fractions; see Eq. (81).

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).

SUBROUTINE CKCTC (T, C, ICKWRK, RCKWRK, CDOT, TAU)
Returns the molar creation rates and characteristic destruction times of the species given temperature(s) and molar concentrations; see Eqs. (84) and (86).

SUBROUTINE CKCTXP (P, T, X, ICKWRK, RCKWRK, CDOT, TAU)
Returns the molar creation rates and characteristic destruction times of the species given pressure, temperature(s) and mole fractions; see Eqs. (84) and (86).

SUBROUTINE CKCTXR (RHO, T, X, ICKWRK, RCKWRK, CDOT, TAU)
Returns the molar creation rates and characteristic destruction times of the species given mass density, temperature(s) and mole fractions; see Eqs. (84) and (86).

SUBROUTINE CKCTYP (P, T, Y, ICKWRK, RCKWRK, CDOT, TAU)

Returns the molar creation rates and characteristic destruction times of the species given mass density, temperature(s) and mass fractions; see Eqs. (84) and (86).

SUBROUTINE CKCTYR (RHO, T, Y, ICKWRK, RCKWRK, CDOT, TAU)

Returns the molar creation rates and characteristic destruction times of the species given mass density, temperature(s) and mass fractions; see Eqs. (84) and (86).

SUBROUTINE CKKFKR (P, T, X, ICKWRK, RCKWRK, FWDK, REVK)

Returns the forward and reverse reaction rates for reactions given pressure, temperature(s) and mole fractions.

SUBROUTINE CKKFRT (P, T, ICKWRK, RCKWRK, RKFT, RKRT)

Returns the forward and reverse reaction rates for reactions given pressure and temperature(s).

SUBROUTINE CKRDEX (I, RCKWRK, RD)*

Get/put the perturbation factor of the Ith reaction

SUBROUTINE CKWC (T, C, ICKWRK, RCKWRK, WDOT)

Returns the molar production rates of the species given temperature(s) and molar concentrations; see Eq. (49).

SUBROUTINE CKWXP (P, T, X, ICKWRK, RCKWRK, WDOT)

Returns the molar production rates of the species given pressure, temperature(s) and mole fractions; see Eq. (49).

SUBROUTINE CKWXR (RHO, T, X, ICKWRK, RCKWRK, WDOT)

Returns the molar production rates of the species given mass density, temperature(s) and mole fractions; see Eq. (49).

SUBROUTINE CKWYP (P, T, Y, ICKWRK, RCKWRK, WDOT)

Returns the molar production rates of the species given pressure, temperature(s) and mass fractions; see Eq. (49).

SUBROUTINE CKWYPK (P, T, Y, RKFT, RKRT, ICKWRK, RCKWRK, WDOT)

Returns the molar production rates of the species given pressure, temperature(s), mass fractions, and forward and reverse rates coefficients; see Eq. (49).

SUBROUTINE CKWYR (RHO, T, Y, ICKWRK, RCKWRK, WDOT)

Returns the molar production rates of the species given mass density, temperature and mass fractions; see Eq. (49).

13. EQUILIBRIUM CONSTANTS AND RATE OF PROGRESS VARIABLES

SUBROUTINE CKEQC (T, C, ICKWRK, RCKWRK, EQKC)

Returns the equilibrium constants of the reactions given temperature(s) and molar concentrations; see Eq. (54).

SUBROUTINE CKEQXP (P, T, X, ICKWRK, RCKWRK, EQKC)

Returns the equilibrium constants for reactions given pressure, temperature(s) and mole fractions; see Eq. (54).

SUBROUTINE CKEQXR (RHO, T, X, ICKWRK, RCKWRK, EQKC)

Returns the equilibrium constants of the reactions given mass density, temperature(s) and mole fractions; see Eq. (54).

SUBROUTINE CKEQYP (P, T, Y, ICKWRK, RCKWRK, EQKC)

Returns the equilibrium constants for reactions given pressure temperature(s) and mass fractions; see Eq. (54).

SUBROUTINE CKEQYR (RHO, T, Y, ICKWRK, RCKWRK, EQKC)

Returns the equilibrium constants of the reactions given mass density, temperature(s) and mass fractions; see Eq. (54).

SUBROUTINE CKQC (T, C, ICKWRK, RCKWRK, Q)

Returns the rates of progress for reactions given temperature(s) and molar concentrations; see Eqs. (51) and (63).

SUBROUTINE CKQXP (P, T, X, ICKWRK, RCKWRK, Q)

Returns the rates of progress for reactions given pressure, temperature(s) and mole fractions; see Eqs. (51) and (63).

SUBROUTINE CKQYP (P, T, Y, ICKWRK, RCKWRK, Q)

Returns the rates of progress for reactions given pressure, temperature(s) and mass fractions; see Eqs. (51) and (63).

SUBROUTINE CKQYR (RHO, T, Y, ICKWRK, RCKWRK, Q)

Returns the rates of progress for reactions given mass density, temperature(s) and mass fractions; see Eqs. (51) and (63).

14. UTILITIES

SUBROUTINE CKCRAY (LINE, NN, KRAY, LOUT, NDIM, NRAY, NF, KERR)

Searches a character string, LINE, and compares the space-delimited substrings in LINE, to an array of character strings, KRAY;

if a substring in LINE is located in KRAY, the index of its location in KRAY is stored in the integer array NRAY. For example, the subroutine might be called to assign CHEMKIN species indices to a given list of species names. This application is illustrated 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
        NDIM      = 10, the dimension of array NRAY(*)
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.
```

SUBROUTINE CKI2CH (NUM, STR, I, KERR)

Returns a character string representation of an integer and the character count of the string.

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

Searches a character string LINE from last to first character, to create a substring IPAR containing NPAR blank-delimited numbers; ISTART is the column of LINE containing IPAR. This allows format-free input of combined alpha-numeric data. For example,

```
input:  LINE*80   = "t1 t2 dt 300.0 3.0E3 50"
        NPAR      = 3, the number of substrings requested
        LOUT      = 6, a logical unit number for diagnostic messages
output:  IPAR*80  = "300.0 3.0E3 50"
        corresponding to the substrings in LINE.
        ISTART    = 11, the starting column in LINE of the NPAR substrings
        KERR      = .FALSE.
```

SUBROUTINE CKR2CH (RNUM, STR, I, KERR)

Returns a 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)

Search a character string, LINE, for (1) a character substring which may also appear in an array of character substrings KRAY, and (2) some number of character substrings representing numbers.

In the case of (1), if the character substring appears in KRAY, KNUM is its index position.

In the case of (2), the character substrings are converted to NVAL real numbers and stored in RVAL, until NEXP are converted.

This allows format-free input of combined alpha-numeric data.

For example, 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 for 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)

Returns an array of substrings in a character string with blanks as the delimiter

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

Searches a character string, LINE, for NEXP space-delimited substrings representing numbers, until NVAL real values are converted and stored in the array, RVAL.

This allows format-free input of numerical data. For example:

input:	LINE	= "0.170E+14 0 47780."
	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, 0.00E+00, 4.778E+04
	KERR	= .FALSE.

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. Subroutines whose call lists have changed since the release of CHEMKIN II, are indicated by a double 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(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
RA(*) - Real array, pre-exponential constants for reactions;
dimension at least II, the total reaction count.
cgs units, mole-cm-sec-K
RB(*) - Real array, temperature dependence exponents for
reactions;
dimension at least II, total reaction count.
cgs units none
RE(*) - Real array, activation energies for reactions;
dimension at least II, the total reaction count.
cgs units, K

```

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 pressure, temperature(s), and mole fractions; see Eq. (46).

INPUT
P - Real scalar, pressure.
cgs units, dynes/cm**2
T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
X(*) - Real array, mole fractions of the mixture;
dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
ABML - Real scalar, mean Helmholtz free energy.
cgs units, ergs/mole

```

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 pressure, temperature(s) and mass fractions; see Eq. (47).

INPUT

- P - Real scalar, pressure.
cgs units, dynes/cm**2
- T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
- Y(*) - Real array, mass fractions of the mixture;
dimension at least KK, the total species count.
- ICKWRK(*) - Integer workspace array; dimension at least LENICK.
- RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

- ABMS - Real scalar, mean Helmholtz free energy.
cgs units, ergs/gm

```

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(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
- ICKWRK(*) - Integer workspace array; dimension at least LENICK.
- RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

- AML(*) - Real array, standard state Helmholtz free energies
for species;
dimension at least KK, the total species count.
cgs units, ergs/mole

```

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(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K

ICKWRK(*) - Integer workspace array; dimension at least LENICK.

RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

AMS(*) - Real array, standard state Helmholtz free energies
for species;
dimension at least KK, the total species count.
cgs units, ergs/gm

```

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 species; see Eqns. (19)-(21).

INPUT

- NDIM1 - Integer scalar, first dimension of A, the three-dimensional array of thermodynamic fit coefficients; NDIM1 must be at least NPCP2, the total number of coefficients for one temperature range.
- NDIM2 - Integer scalar, second dimension of A; NDIM2 must be at least MXTP-1, the total number of temperature ranges.
- ICKWRK(*) - Integer workspace array; dimension at least LENICK.
- RCKWRK(*) - Real workspace array; dimension at least LENRCK.
- MAXTP - Integer scalar, number of temperatures used to divide the temperature ranges of thermodynamic fits.

OUTPUT

- NT(*) - Integer array, total number of temperatures used in fitting coefficients of thermodynamic properties for the species; dimension at least KK, the total species count.
- TMP(*,*) - Real matrix, temperatures for dividing the thermodynamic fits for species; dimension at least MAXTP for the first, and at least KK for the second, the total species count.
cgs units, K
- A(*,*,*) - Real three-dimensioned array of fit coefficients to the thermodynamic data for species; dimension exactly NPCP2 for the first, exactly MAXTP-1 for the second, and at least KKTOT for the third, the total species count.
The indicies in A(N,L,K) mean-
N = 1,NN represent polynomial coefficients in CP/R
 $CP/R(K) = A(1,L,K) + A(2,L,K)*T + A(3,L,K)*T**2 + \dots$
N = NN+1 is for the formation enthalpies, i.e.,
HO/R = A(NN+1,L,K)
N = NN+2 is for the formation entropies, i.e.,
SO/R = A(NN+2,L,K)
L = 1 is for temperature <= TMP(2,K)
L = 2 is for TMP(2,K) < temperature <= TMP(3)
:
L = (NTMP-1) is for TMP(NTMP-1) <= temperature;
K is the species index

```

CKAWT      CKAWT      CKAWT      CKAWT      CKAWT      CKAWT      CKAWT
*****
*****
*****

```

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

INPUT
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
AWT(*) - Real array, atomic weights of the elements;
dimension at least MM, the total element count.
cgs units, gm/mole

```

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 temperature(s) and molar concentrations; see Eq. (81).

INPUT
T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
C(*) - Real array, concentrations of the species;
dimension at least KK, the total species count.
cgs units, mole/cm**3
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
CDOT(*) - Real array, chemical creation rates of the species;
dimension at least KK, the total species count.
cgs units, mole/(cm**3*sec)
DDOT(*) - Real array, chemical destruction rates of the species;
dimension at least KK, the total species count.
cgs units, moles/(cm**3*sec)

```

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(s) and mole fractions; see Eq. (81).

INPUT

P - Real scalar, pressure.
cgs units, dynes/cm**2

T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K

X(*) - Real array, mole fractions of the mixture;
dimension at least KK, the total species count.

ICKWRK(*) - Integer workspace array; dimension at least LENICK.

RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

CDOT(*) - Real array, chemical creation rates of the species;
dimension at least KK, the total species count.
cgs units, mole/(cm**3*sec)

DDOT(*) - Real array, chemical destruction rates of the species;
dimension at least KK, the total species count.
cgs units, moles/(cm**3*sec)

```

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 mass density, temperature(s) and mole fractions; see Eq. (81).

INPUT

RHO - Real scalar, mass density.
cgs units, gm/cm**3

T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K

X(*) - Real array, mole fractions of the mixture;
dimension at least KK, the total species count.

ICKWRK(*) - Integer workspace array; dimension at least LENICK.

RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

CDOT(*) - Real array, chemical creation rates of the species;
dimension at least KK, the total species count.
cgs units, mole/(cm**3*sec)

DDOT(*) - Real array, chemical destruction rates of the species;
dimension at least KK, the total species count.
cgs units, moles/(cm**3*sec)

```

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(s) and mass fractions; see Eq. (81).

INPUT

- P - Real scalar, pressure.
cgs units, dynes/cm**2
- T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
- Y(*) - Real array, mass fractions of the mixture;
dimension at least KK, the total species count.
- ICKWRK(*) - Integer workspace array; dimension at least LENICK.
- RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

- CDOT(*) - Real array, chemical creation rates of the species;
dimension at least KK, the total species count.
cgs units, mole/(cm**3*sec)
- DDOT(*) - Real array, chemical destruction rates of the species;
dimension at least KK, the total species count.
cgs units, moles/(cm**3*sec)

```

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 mass density, temperature(s) and mass fractions; see Eq. (81).

INPUT

- RHO - Real scalar, mass density.
cgs units, gm/cm**3
- T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
- Y(*) - Real array, mass fractions of the mixture;
dimension at least KK, the total species count.
- ICKWRK(*) - Integer workspace array; dimension at least LENICK.
- RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

- CDOT(*) - Real array, chemical creation rates of the species;
dimension at least KK, the total species count.
cgs units, mole/(cm**3*sec)
- DDOT(*) - Real array, chemical destruction rates of the species;
dimension at least KK, the total species count.
cgs units, moles/(cm**3*sec)

```

CKCHRG      CKCHRG      CKCHRG      CKCHRG      CKCHRG      CKCHRG      CKCHRG
*****
*****
*****

```

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

INPUT
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
KCHARG(*) - Integer array, electronic charges of the species;
dimension at least KK, the total species count.
KCHARG(K)=-2 indicates that species K has two
excess electrons.

```

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 which corresponds to a character string;
leading and trailing blanks are ignored.

INPUT
IST - Character string; length determined by application
program.
IRAY(*) - Character string array; dimension at least II, the total
number of character strings for be searched.
II - Integer scalar, the length of IRAY to be searched.

OUTPUT
I - Integer scalar, the first array index in IRAY of a
character string IST, or 0 if IST is not found.

```

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 scalar; species index number.
Q(*) - Real array, rates of progress for reactions;
dimension at least II, the total reaction count.
cgs units, moles/(cm**3*sec)
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
CIK(*) - Real array, contributions of the reactions to the production rate of species K;
dimension least II, the total reaction count.
cgs units, mole/(cm**3*sec)

```

CKCPBL    CKCPBL    CKCPBL    CKCPBL    CKCPBL    CKCPBL    CKCPBL
*****
*****
*****

```

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

INPUT
T(*) - Real array, temperature(s); dimension is determined by the application program to be the total number of species temperatures, nominally 1.
cgs units, K
X(*) - Real array, mole fractions of the mixture;
dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
CPBML - Real scalar, mean specific heat at constant pressure.
cgs units, ergs/(mole*K)

```

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(*) - Real array, temperature(s); dimension is determined by the application program to be the total number of species temperatures, nominally 1.
 cgs units, K
Y(*) - Real array, mass fractions of the mixture; dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
CPBMS - Real scalar, mean specific heat at constant pressure.
 cgs units - ergs/(gm*K)

```

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(*) - Real array, temperature(s); dimension is determined by the application program to be the total number of species temperatures, nominally 1.
 cgs units, K
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
CPML(*) - Real array, specific heats at constant pressure for the species;
 dimension at least KK, the total species count.
 cgs units, ergs/(mole*K)

```

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(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
CPMS(*) - Real array, specific heats at constant pressure for
species;
dimension at least KK, the total species count.
cgs units, ergs/(gm*K)

```

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(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
CPOR(*) - Real array, nondimensional specific heats at constant
pressure for species;
dimension at least KK, the total species count.

```

CKCRAY      CKCRAY      CKCRAY      CKCRAY      CKCRAY      CKCRAY      CKCRAY
*****
*****
*****

```

SUBROUTINE CKCRAY (LINE, NN, KRAY, LOUT, NDIM, NRAY, NF, KERR)
Searches a character string, LINE, and compares the space-delimited substrings in LINE, to an array of character strings, KRAY; if a substring in LINE is located in KRAY, the index of its location in KRAY is stored in the integer array NRAY. For example, the subroutine might be called to assign CHEMKIN species indices to a given list of species names. This application is illustrated 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.
        NDIM      = 10, the dimension of array NRAY(*)
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      - Character string.
KRAY(*)   - Character string array; dimension at least NN.
NN        - Integer scalar, total character string count of KRAY.
LOUT      - Integer scalar, formatted output file unit.
NDIM      - Integer scalar, dimension of the integer array NRAY.

```

OUTPUT

```

NRAY(*)   - Integer array, indices of the elements of KRAY
           which correspond to the substrings in LINE;
           dimension at least NDIM.
NF        - Integer scalar, count of correspondences found.
KERR      - Logical, syntax or dimensioning Error flag.

```

```

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(s) and molar concentrations; see Eqs. (84) and (86).

INPUT

T(*) - Real array, temperature(s); dimension is determined by the application program to be the total number of species temperatures, nominally 1.
cgs units, K

C(*) - Real array, concentrations of species; dimension at least KK, the total species count.
cgs units, mole/cm**3

ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

CDOT(*) - Real array, chemical creation rates of the species; dimension at least KK, the total species count.
cgs units, mole/(cm**3*sec)

TAU(*) - Real array, characteristic destruction times of species; dimension at least KK, the total species count.
cgs units, sec

```

CKCTX      CKCTX      CKCTX      CKCTX      CKCTX      CKCTX      CKCTX
*****
*****
*****

```

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

INPUT

C(*) - Real array, concentrations of the species; dimension at least KK, the total species count.
cgs units - mole/cm**3

ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

X(*) - Real array, mole fraction of the mixture; dimension at least KK, the total species count.

```

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 pressure, temperature(s) and mole fractions; see Eqs. (84) and (86).

```

INPUT
P          - Real scalar, pressure.
             cgs units, dynes/cm**2
T(*)       - Real array, temperature(s); dimension is determined by
             the application program to be the total number of
             species temperatures, nominally 1.
             cgs units, K
X(*)       - Real array, mole fractions of the mixture;
             dimension at least KK, the total species count.
ICKWRK(*)  - Integer workspace array; dimension at least LENICK.
RCKWRK(*)  - Real workspace array; dimension at least LENRCK.

OUTPUT
CDOT(*)    - Real array, chemical creation rates of the species;
             dimension at least KK, the total species count.
             cgs units, mole/(cm**3*sec)
TAU(*)     - Real array, characteristic destruction times of the
             species;
             dimension at least KK, the total species count.
             cgs units, sec

```

```

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 mass density, temperature(s) and mole fractions; see Eqs. (84) and (86).

```

INPUT
RHO        - Real scalar, mass density.
             cgs units, gm/cm**3
T(*)       - Real array, temperature(s); dimension is determined by
             the application program to be the total number of
             species temperatures, nominally 1.
             cgs units, K
X(*)       - Real array, mole fractions of the mixture;
             dimension at least KK, the total species count.
ICKWRK(*)  - Integer workspace array; dimension at least LENICK.
RCKWRK(*)  - Real workspace array; dimension at least LENRCK.

OUTPUT
CDOT(*)    - Real array, chemical creation rates of the species;
             dimension at least KK, the total species count.
             cgs units, mole/(cm**3*sec)
TAU(*)     - Real array, characteristic destruction times of species;
             dimension at least KK, the total species count.
             cgs units, sec

```

```

CKCTY      CKCTY      CKCTY      CKCTY      CKCTY      CKCTY      CKCTY
*****
*****
*****

```

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

INPUT
C(*) - Real array, concentrations of the species;
dimension at least KK, the total species count.
cgs units, mole/cm**3
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
Y(*) - Real array, mass fractions of the mixture;
dimension at least KK, the total species count.

```

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 mass density, temperature(s) and mass fractions; see Eqs. (84) and (86).

INPUT
P - Real scalar, pressure.
cgs units, dynes/cm**2
T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
Y(*) - Real array, mass fractions of the mixture;
dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
CDOT(*) - Real array, chemical creation rates of the species;
dimension at least KK, the total species count.
cgs units, mole/(cm**3*sec)
TAU(*) - Real array, characteristic destruction times of the
species;
dimension at least KK, the total species count.
cgs units, sec

```

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 mass density, temperature(s) and mass fractions; see Eqs. (84) and (86).

```

INPUT
RHO      - Real scalar, mass density.
          cgs units, gm/cm**3
T(*)     - Real array, temperature(s); dimension is determined by
          the application program to be the total number of
          species temperatures, nominally 1.
          cgs units, K
Y(*)     - Real array, mass fractions of the mixture;
          dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
CDOT(*)  - Real array, chemical creation rates of the species;
          dimension at least KK, the total species count.
          cgs units, mole/(cm**3*sec)
TAU(*)   - Real array, characteristic destruction times of the
          species;
          dimension at least KK, the total species count.
          cgs units, sec

```

```

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(*)     - Real array, temperature(s); dimension is determined by
          the application program to be the total number of
          species temperatures, nominally 1.
          cgs units, K
X(*)     - Real array, mole fractions of the mixture;
          dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
CVBML    - Real scalar, mean specific heat at constant volume.
          cgs units, ergs/(mole*K)

```

```

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(*)        - Real array, temperature(s); dimension is determined by
              the application program to be the total number of
              species temperatures, nominally 1.
              cgs units, K
Y(*)        - Real array, mass fractions of the mixture;
              dimension at least KK, the total species count.
ICKWRK(*)   - Integer workspace array; dimension at least LENICK.
RCKWRK(*)   - Real    workspace array; dimension at least LENRCK.

OUTPUT
CVBMS       - Real scalar, mean specific heat at constant volume.
              cgs units, ergs/(gm*K)

```

```

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(*)        - Real array, temperature(s); dimension is determined by
              the application program to be the total number of
              species temperatures, nominally 1.
              cgs units, K
ICKWRK(*)   - Integer workspace array; dimension at least LENICK.
RCKWRK(*)   - Real    workspace array; dimension at least LENRCK.

OUTPUT
CVML(*)     - Real array, specific heats at constant volume for
              species;
              dimension at least KK, the total species count.
              cgs units, ergs/(mole*K)

```

```

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(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
CVMS(*) - Real array, specific heats at constant volume for
species;
dimension at least KK, the total species count.
cgs units, ergs/(gm*K)

```

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

```

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

INPUT
T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
C(*) - Real array, concentrations of the species;
dimension at least KK, the total species count.
cgs units, mole/cm**3
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
EQKC(*) - Real array, equilibrium constants in concentration units
for reactions;
dimension at least II, the total reaction count.
cgs units, (mole/cm**3)**some power, depending on
the reaction

```

CKEQXP      CKEQXP      CKEQXP      CKEQXP      CKEQXP      CKEQXP      CKEQXP
*****
*****
*****

```

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

```

INPUT
P          - Real scalar, pressure.
             cgs units, dynes/cm**2
T(*)       - Real array, temperature(s); dimension is determined by
             the application program to be the total number of
             species temperatures, nominally 1.
             cgs units, K
X(*)       - Real array, mole fractions of the mixture;
             dimension at least KK, the total species count.
ICKWRK(*)  - Integer workspace array; dimension at least LENICK.
RCKWRK(*)  - Real workspace array; dimension at least LENRCK.

OUTPUT
EQKC(*)    - Real array, equilibrium constants for reactions;
             dimension at least II, the total reaction count.
             cgs units, (mole/cm**3)**some power, depending on
             the reaction

```

```

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(s) and mole fractions; see Eq. (54).

```

INPUT
RHO        - Real scalar, mass density.
             cgs units, gm/cm**3
T(*)       - Real array, temperature(s); dimension is determined by
             the application program to be the total number of
             species temperatures, nominally 1.
             cgs units, K
X(*)       - Real array, mole fractions of the mixture;
             dimension at least KK, the total species count.
             species.
ICKWRK(*)  - Integer workspace array; dimension at least LENICK.
RCKWRK(*)  - Real workspace array; dimension at least LENRCK.

OUTPUT
EQKC(*)    - Real array, equilibrium constants in concentration units
             for reactions;
             dimension at least II, the total reaction count.
             cgs units, (mole/cm**3)**some power, depending on
             the reaction

```

```

CKEQYP      CKEQYP      CKEQYP      CKEQYP      CKEQYP      CKEQYP      CKEQYP
*****
*****
*****

```

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

INPUT

P - Real scalar, pressure.
cgs units, dynes/cm**2

T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K

Y(*) - Real array, mass fractions of the mixture;
dimension at least KK, the total species count.

ICKWRK(*) - Integer workspace array; dimension at least LENICK.

RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

EQKC(*) - Real array, equilibrium constants in concentration units
for reactions;
dimension at least II, the total reaction count.
cgs units, (mole/cm**3)**some power, depending on
the reaction

```

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(s) and mass fractions; see Eq. (54).

INPUT

RHO - Real scalar, mass density.
cgs units; gm/cm**3

T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K

Y(*) - Real array, mass fractions of the mixture;
dimension at least KK, the total species count.

ICKWRK(*) - Integer workspace array; dimension at least LENICK.

RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

EQKC(*) - Real array, equilibrium constants in concentration units
for reactions;
dimension at least II, the total reaction count.
cgs units; (mole/cm**3)**some power, depending on
the reaction

```

CKFAL      CKFAL      CKFAL      CKFAL      CKFAL      CKFAL      CKFAL
*****
*****
*****

```

SUBROUTINE CKFAL (NDIM, ICKWRK, RCKWRK, IFOP, IFLO, KFAL, FPAR)
Returns a set of flags indicating whether a reaction has pressure-dependent behavior and an array of parameters.

INPUT

NDIM - Integer scalar, first dimension of the matrix FPAR; NDIM must be greater than or equal to NFAR, the maximum number of supplemental rate parameters, which is currently equal to 8.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

IFOP(*) - Integer array, flags indicating pressure-dependent behavior; dimension at least II, the total reaction count. IFOP(I) indicates the pressure-dependent behavior of reaction I:
0 - No pressure dependency
1 - Lindeman form (3 parameters)
2 - SRI form (8 parameters)
3 - Troe form (6 parameters)
4 - Troe form (7 parameters)

IFLO(*) - Integer array, flags indication pressure-dependency; dimension at least II, the total reaction count. IFLO(I) indicates
0 - unimolecular fall-off,
1 - chemically activated bi-molecular.

KFAL(*) - Integer array, flags indicating type of bath-gas concentration to be used in expressions (see footnote on page 27); dimension at least II, the total reaction count. KFAL(I) indicates the type of reaction I:
0 - Use total concentration of gas mixture (with the added capability of using enhanced third body coefficients) (default)
K - Use the concentration of species K

FPAR(*,*) - Real matrix, pressure dependency parameters; dimension at least NFAR for the first, the maximum number of parameters (currently 8), and at least II for the second, the total reaction count.
The number of parameters depends on the particular functional form indicated by the IFOP array: FPAR(1,I), FPAR(2,I), FPAR(3,I) are always the parameters entered on the LOW auxiliary keyword line in the CHEMKIN interpreter input file.
FPAR(1,I) = Pre-exponential for low pressure limiting rate constant
cgs units, mole-cm-sec-K
FPAR(2,I) = Temperature dependence exponents for the low pressure limiting rate constants.
FPAR(3,I) = Activation energy for the low pressure limiting rate constant.
cgs units, K
Additional FPAR values depend on IFOP:

```

IFOP(I) = 2:
  FPAR(4,I) = a          (See Eqn. (74))
  FPAR(5,I) = b (Kelvin) (See Eqn. (74))
  FPAR(6,I) = c (Kelvin) (See Eqn. (74))
  FPAR(7,I) = d          (See Eqn. (74))
  FPAR(8,I) = e          (See Eqn. (74))
IFOP(I) = 3:
  FPAR(4,I) = a          (See Eqn. (73))
  FPAR(5,I) = T*** (Kelvin) (See Eqn. (73))
  FPAR(6,I) = T* (Kelvin) (See Eqn. (73))
IFOP(I) = 4:
  FPAR(4,I) = a          (See Eqn. (73))
  FPAR(5,I) = T*** (Kelvin) (See Eqn. (73))
  FPAR(6,I) = T* (Kelvin) (See Eqn. (73))
  FPAR(7,I) = T** (Kelvin) (See Eqn. (73))

```

```

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 pressure, temperature(s) and mole fractions; see Eq. (44).

INPUT

- P - Real scalar, pressure.
cgs units, dynes/cm**2
- T(*) - Real array, temperature(s); dimension is determined by the application program to be the total number of species temperatures, nominally 1.
cgs units, K
- X(*) - Real array, mole fractions of the mixture;
dimension at least KK, the total species count.
- ICKWRK(*) - Integer workspace array; dimension at least LENICK.
- RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

- GBML - Real scalar, mean Gibbs free energy.
cgs units, ergs/mole

```

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 pressure, temperature(s), and mass fractions; see Eq. (45).

INPUT
P - Real scalar, pressure.
cgs units, dynes/cm**2
T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
Y(*) - Real array, mass fractions of the mixture;
dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
GBMS - Real scalar, mean Gibbs free energy.
cgs units, ergs/gm

```

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(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
GML(*) - Real array, standard state gibbs free energies for
the species;
dimension at least KK, the total species count.
cgs units, ergs/mole

```

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(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
GMS(*) - Real array, standard state Gibbs free energies for
the species;
dimension at least KK, the total species count.
cgs units, ergs/gm

```

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(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
X(*) - Real array, mole fractions of the mixture;
dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
HBML - Real scalar, mean enthalpy.
cgs units - ergs/mole

```

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(*)        - Real array, temperature(s); dimension is determined by
              the application program to be the total number of
              species temperatures, nominally 1.
              cgs units, K
Y(*)        - Real array, mass fractions of the mixture;
              dimension at least KK, the total species count.
ICKWRK(*)   - Integer workspace array; dimension at least LENICK.
RCKWRK(*)   - Real workspace array; dimension at least LENRCK.

```

```

OUTPUT
HBMS        - Real scalar, mean enthalpy.
              cgs units, ergs/gm

```

```

CKHML      CKHML      CKHML      CKHML      CKHML      CKHML      CKHML
*****
*****
*****

```

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

```

INPUT
T(*)        - Real array, temperature(s); dimension is determined by
              the application program to be the total number of
              species temperatures, nominally 1.
              cgs units, K
ICKWRK(*)   - Integer workspace array; dimension at least LENICK.
RCKWRK(*)   - Real workspace array; dimension at least LENRCK.

```

```

OUTPUT
HML(*)      - Real array, enthalpies for species;
              dimension at least KK, the total species count.
              cgs units, ergs/mole

```

```

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

```

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

```

INPUT
T(*)      - Real array, temperature(s); dimension is determined by
           the application program to be the total number of
           species temperatures, nominally 1.
           cgs units, K
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
HMS(*)    - Real array, enthalpies for species;
           dimension at least KK, the total species count.
           cgs units, ergs/gm

```

```

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

```

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

```

INPUT
T(*)      - Real array, temperature(s); dimension is determined by
           the application program to be the total number of
           species temperatures, nominally 1.
           cgs units, K
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
HORT(*)   - Real array, nondimensional enthalpies for species;
           dimension at least KK, the total species count.

```

```

CKHRX      CKHRX      CKHRX      CKHRX      CKHRX      CKHRX      CKHRX
*****
*****
*****

```

SUBROUTINE CKHRX (I, HML, ICKWRK, RCKWRK, HRXI)
Returns the molar heat of reaction I.

INPUT
I - Integer scalar, reaction index.
HML(*) - Real array, molar enthalpies for species;
dimension at least KK, the total species count.
cgs units, ergs/mole
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
HRXI - Real scalar, molar heat of reaction I.
cgs units, ergs/mole

```

CKI2CH     CKI2CH     CKI2CH     CKI2CH     CKI2CH     CKI2CH     CKI2CH
*****
*****
*****

```

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

INPUT
NUM - Integer scalar, to be converted to a character string;
the maximum magnitude of NUM is machine-dependent.

OUTPUT
STR - Character string, left-justified character representation
of NUM.
I - Integer scalar, the non-blank character count of STR.
KERR - Logical, character length error flag.

```

CKIEXC      CKIEXC      CKIEXC      CKIEXC      CKIEXC      CKIEXC      CKIEXC
*****
*****
*****

```

SUBROUTINE CKIEXC (ICKWRK, RCKWRK, IEXC, EEXC)
Returns a set of flags indicating whether the reactions are
excitation reactions and, if so, the energy loss per event in eV.

INPUT
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
IEXC(*) - Integer array, excitation-only reaction flag;
dimension at least II, the total reaction count.
IEXC(I)= -1 reaction I is not an excitation-only reax
IEXC(I)= 1 reaction I is an excitation reaction
EEXC(*) - Real array, excitation energy loss per event in forward
direction for reactions;
dimension at least II, the total reaction count.

```

CKIMOM      CKIMOM      CKIMOM      CKIMOM      CKIMOM      CKIMOM      CKIMOM
*****
*****
*****

```

SUBROUTINE CKIMOM (ICKWRK, IMOM)
Returns a set of flags indicating whether the reactions are
electron momentum-transfer collision frequencies and, if so,
the index of the species with which the electron collides.

INPUT
ICKWRK(*) - Integer workspace array; dimension at least LENICK.

OUTPUT
IMOM(*) - Integer array, electron momentum-transfer collision
frequency flags for reactions;
dimension at least II, the total reaction count.
IMOM(I)= -1 reaction I is not a mom-transfer coll freq
IMOM(I)= K reaction I is a mom-transfer coll frequency
and K is species index of the electron's
collision partner

```

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(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

MM - Integer scalar, mechanism total element count.
KK - Integer scalar, mechanism total species count.
II - Integer scalar, mechanism total reaction count.
NFIT - Integer scalar, number of coefficients in fits to
thermodynamic data for a temperature range;
NFIT=number of coefficients in polynomial fits to CP/R
plus 2.

```

CKINIT      CKINIT      CKINIT      CKINIT      CKINIT      CKINIT      CKINIT
*****
*****
*****

```

SUBROUTINE CKINIT (LENICK, LENRCK, LENCCK, LINC, LOUT, ICKWRK,
RCKWRK, CCKWRK, IFLAG)**
Reads the linkfile and creates the internal work arrays ICKWRK,
RCKWRK and CCKWRK. CKINIT must be called before any other CHEMKIN
subroutine can be used, as the work arrays must be available as
their input.

INPUT

LENICK - Integer scalar, length of the integer work array, ICKWRK.
LENRCK - Integer scalar, length of the real work array, RCKWRK.
LENCCK - Integer scalar, length of the character work array, CCKWRK.
LINC - Integer scalar, linkfile input file unit number.
LOUT - Integer scalar, formatted output file unit number.

OUTPUT

ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.
CCKWRK(*) - Character string workspace array;
dimension at least LENCCK.
IFLAG - Integer scalar to indicate successful reading of
linkfile; IFLAG>0 is an error type.

```

CKINU      CKINU      CKINU      CKINU      CKINU      CKINU      CKINU
*****
*****
*****

```

SUBROUTINE CKINU (I, NDIM, ICKWRK, RCKWRK, NSPEC, KI, NU)
Returns a count of species in a reaction, and their indices
and stoichiometric coefficients; see Eq. (50).

INPUT

- I - Integer scalar, index of a reaction;
I must be positive, and less than or equal to NII,
the total reaction count.
- NDIM - Integer scalar, dimension of the arrays KI and NU;
NDIM must be at least MAXSP, the maximum number of
species allowed in a reaction.
- ICKWRK(*) - Integer workspace array; dimension at least LENICK.
- RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

- NSPEC - Integer scalar, the total species count for reaction I.
- KI(*) - Integer array, species indices for those in
reaction I; dimension at least MAXSP, the maximum
number of species allowed in a reaction.
KI(N) is the index of the Nth species in reaction I.
- NU(*) - Integer array, stoichiometric coefficients for those
in reaction I;
dimension at least MAXSP, the maximum number of
species allowed in a reaction.
NU(N) is the stoichiometric coefficient of the Nth
Nth species in reaction I, and
NU < 0 if the Nth species is a reactant;
NU > 0 if the Nth species is a product.

```

CKION      CKION      CKION      CKION      CKION      CKION      CKION
*****
*****
*****

```

SUBROUTINE CKION (ICKWRK, KION)
Returns the ion species indices

INPUT

- ICKWRK(*) - Integer workspace array; dimension at least LENICK.
- RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

- KION(*) - Integer array, ion species indices;
dimension at least NKKI, the total ion count.

```

CKIORD      CKIORD      CKIORD      CKIORD      CKIORD      CKIORD      CKIORD
*****
*****
*****

```

```

SUBROUTINE CKIORD (IDIM, KDIM, ICKWRK, RCKWRK, NFORD, IFORD, FORD,
                  NRORD, IRORD, RORD)

```

Returns the count and indices of reactions with modified species order and the order values for the species.

INPUT

- IDIM - Integer scalar, dimension of arrays IFORD and IRORD; IDIM must be at least NORD, the total number of reactions with modified species orders.
- KDIM - Integer scalar, first dimension of the arrays FORD and RORD; KDIM must be at least NKK, the total species count.
- ICKWRK(*) - Integer workspace array; dimension at least LENICK.
- RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

- NFORD - Integer scalar, total number of reactions with modified forward species orders.
- IFORD(*) - Integer array, indices of reactions with modified forward species orders; dimension at least NFORD.
- FORD(*,*) - Real matrix, the modified forward species orders for the NFORD reactions; dimension at least NKK for the first, the total species count, and at least NFORD for the second. FORD(K,N) is the forward order of species K for the Nth change-order reaction.
- NRORD - Integer scalar, total number of reactions with modified reverse species orders.
- IRORD(*) - Integer array, indices of reactions with modified reverse species orders; dimension at least NRORD.
- RORD(*,*) - Real matrix, the modified reverse species orders for the NRORD reactions; dimension at least NKK for the first, the total species count, and at least NRORD for the second. RORD(K,N) is the reverse order of species K for the Nth change-order reaction.

```

CKIRNU      CKIRNU      CKIRNU      CKIRNU      CKIRNU      CKIRNU      CKIRNU
*****
*****
*****

```

```

SUBROUTINE CKIRNU (IDIM, NDIM, ICKWRK, RCKWRK, NIRNU, IRNU, NSPEC,
                  KI, RNU)

```

Returns the count and indices of reactions with real stoichiometric coefficients, counts of species in the reactions, and the species indices and coefficients; see Eq. (50).

INPUT

- IDIM - Integer scalar, dimension of the arrays IRNU and NSPEC, and the second dimension of matrices KI and RNU; IDIM must be at least NIRNU, the number of reactions with real stoichiometric coefficients.
- NDIM - Integer scalar, first dimension of matrices KI and RNU; NDIM must be at least MAXSP, the maximum number of species allowed in a reaction.
- ICKWRK(*) - Integer workspace array; dimension at least LENICK.
- RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

- NIRNU - Integer scalar, total number of reactions with real stoichiometric coefficients.
- IRNU(*) - Integer array, indices of reactions with real stoichiometric coefficients; dimension at least NIRNU.
- NSPEC(*) - Integer array, total number of species in a reaction; dimension at least NIRNU.
- KI(*,*) - Integer matrix, species indices for species in the NIRNU reactions; dimension at least MAXSP for the first, and at least NIRNU for the second. KI(M,N) is the species index of the Mth species in the Nth real coefficient reaction.
- RNU(*,*) - Real matrix, stoichiometric coefficients for species in the NIRNU reactions; dimension at least MAXSP for the first, and at least NIRNU for the second. RNU(M,N) is the stoichiometric coefficient of the Mth species in the Nth real coefficient reaction, and RNU(M,*) < 0 if the Mth species is a reactant; RNU(M,*) > 0 if the Mth species is a product.

```

CKITDE      CKITDE      CKITDE      CKITDE      CKITDE      CKITDE      CKITDE
*****
*****
*****

```

```

SUBROUTINE CKITDE (ICKWRK, RCKWRK, ITDE)

```

Returns a set of flags indicating whether the reactions are non-thermal, and if so, returns the index of the species on which the reaction depends.

INPUT

- ICKWRK(*) - Integer workspace array; dimension at least LENICK.
- RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

- ITDE(*) - Integer array, electron-impact flags for reactions; dimension at least II, the total reaction count. ITDE(I) = -1 reaction I is not a third-body reaction; ITDE(I) = K reaction I is a third-body reaction with temperature dependence on species # K

```

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(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
ITHB(*) - Integer array, third-body indices for reactions; dimension at least II, the total reaction count.
ITHB(I)= -1 reaction I is not a third-body reactions
ITHB(I)= 0 reaction I 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.
IREV(*) - Integer array, reversibility indices and species count (reactants plus products) for reactions; dimension at least II, the total reaction count.
IREV(I)=+N, reversible reaction I has N species
IREV(I)=-N, irreversible reaction I has N species

```

CKIXSM     CKIXSM     CKIXSM     CKIXSM     CKIXSM     CKIXSM     CKIXSM
*****
*****
*****

```

SUBROUTINE CKIXSM (ICKWRK, IXSM, IXSK)
Returns a set of flags indicating whether the reactions are ion momentum-transfer cross sections.

INPUT
ICKWRK(*) - Integer workspace array; dimension at least LENICK.

OUTPUT
IXSM(*) - Integer array, ion momentum-transfer cross-section flag; dimension at least II, the total reaction count.
IXSM(I)= -1 reaction I is not a ion mom-transfer x-sec
IXSM(I)= KI reaction I is a ion mom-trans cross-section and KI is the ion species index
IXSK(*) - Integer array, species indices for the collision partner of the ion momentum-transfer cross-section reactions; dimension at least II, the total reaction count.

```

CKKFKR      CKKFKR      CKKFKR      CKKFKR      CKKFKR      CKKFKR      CKKFKR
*****
*****
*****

```

SUBROUTINE CKKFKR (P, T, X, ICKWRK, RCKWRK, FWDK, REVK)
Returns the forward and reverse reaction rates for reactions
given pressure, temperature(s) and mole fractions.

```

INPUT
P          - Real scalar, pressure.
            cgs units, dynes/cm**2
T(*)       - Real array, temperature(s); dimension is determined by
            the application program to be the total number of
            species temperatures, nominally 1.
            cgs units, K
X(*)       - Real array, mole fractions of the mixture;
            dimension at least KK, the total species count.
ICKWRK(*)  - Integer workspace array; dimension at least LENICK.
RCKWRK(*)  - Real      workspace array; dimension at least LENRCK.

OUTPUT
FWDK(*)    - Real array, forward reaction rates for reactions;
            dimension at least II, the total reaction count.
            cgs units, depends on the reaction
REVK(*)    - Real array, reverse reaction rates for reactions;
            dimension at least II, the total reaction count.
            cgs units, depends on the reaction

```

```

CKKFRT      CKKFRT      CKKFRT      CKKFRT      CKKFRT      CKKFRT      CKKFRT
*****
*****
*****

```

SUBROUTINE CKKFRT (P, T, ICKWRK, RCKWRK, RKFT, RKRT)
Returns the forward and reverse reaction rates for reactions
given pressure and temperature(s).

```

INPUT
T(*)       - Real array, temperature(s); dimension is determined by
            the application program to be the total number of
            species temperatures, nominally 1.
            cgs units, K
ICKWRK(*)  - Integer workspace array; dimension at least LENICK.
RCKWRK(*)  - Real      workspace array; dimension at least LENRCK.

OUTPUT
RKFT(*)    - Real array, forward reaction rates for reactions;
            dimension at least II, the total reaction count.
            cgs units, depends on the reaction
RKRT(*)    - Real array, reverse reaction rates for reactions;
            dimension at least II, the total reaction count.
            cgs units, depends on the reaction

```

```

CKKTFL      CKKTFL      CKKTFL      CKKTFL      CKKTFL      CKKTFL      CKKTFL
*****
*****
*****

```

SUBROUTINE CKKTFL (ICKWRK, KTFL)
 Allows the user to assign a location in the temperature array
 to use for each gas-phase species.

INPUT
 ICKWRK(*) - Integer workspace array; dimension at least LENICK.
 KTFL(*) - Integer array, indices into the temperature(s) for
 species;
 dimension at least KK, the total species count.
 Default value stored in ICKWRK is set to 1 in CKINIT.

```

CKLEN      CKLEN      CKLEN      CKLEN      CKLEN      CKLEN      CKLEN
*****
*****
*****

```

SUBROUTINE CKLEN (LINC, LOUT, LENI, LENR, LENC, IFLAG)
 Returns the lengths required for work arrays.

INPUT
 LINC - Integer scalar, input file unit for the linkfile.
 LOUT - Integer scalar, formatted output file unit.

OUTPUT
 LENI - Integer scalar, minimum length required for the
 integer work array.
 LENR - Integer scalar, minimum length required for the
 real work array.
 LENC - Integer scalar, minimum length required for the
 character work array.
 IFLAG - Integer scalar, indicates successful reading of
 linkfile; IFLAG>0 indicates error type.

```

CKMMWC      CKMMWC      CKMMWC      CKMMWC      CKMMWC      CKMMWC      CKMMWC
*****
*****
*****

```

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

INPUT
 C(*) - Real array, concentrations of the species;
 dimension at least KK, the total species count.
 cgs units, mole/cm³
 ICKWRK(*) - Integer workspace array; dimension at least LENICK.
 RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
 WTM - Real scalar, mean molecular weight of the mixture.
 cgs units, gm/mole

```

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

```

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

INPUT
X(*) - Real array, mole fractions of the mixture;
dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
WTM - Real scalar, mean molecular weight of the mixture.
cgs units, gm/mole

```

CKMMWY      CKMMWY      CKMMWY      CKMMWY      CKMMWY      CKMMWY      CKMMWY
*****
*****
*****

```

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

INPUT
Y(*) - Real array, mass fractions of the mixture;
dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
WTM - Real scalar, mean molecular weight of the mixture.
cgs units, gm/mole

```

CKMXTP      CKMXTP      CKMXTP      CKMXTP      CKMXTP      CKMXTP      CKMXTP
*****
*****
*****

```

SUBROUTINE CKMXTP (ICKWRK, MAXTP)
Returns the maximum number of temperatures used in fitting the thermodynamic properties of the species.

INPUT
ICKWRK(*) - Integer workspace array; dimension at least LENICK.

OUTPUT
MXTP - Integer scalar, maximum number of temperatures used to fit the thermodynamic properties of the species.

```

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

```

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

INPUT

MDIM - Integer scalar, first dimension of the matrix NCF;
MDIM must be equal to or greater than MM, the total
element count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

NCF(*,*) - Real matrix, the elemental composition of the species;
dimension at least MM for the first, the total element
count, and at least KK for the second, the total species
count.
NCF(M,K) is the quantity of the element M in species K.

```

CKNPAR     CKNPAR     CKNPAR     CKNPAR     CKNPAR     CKNPAR     CKNPAR
*****
*****
*****

```

SUBROUTINE CKNPAR (LINE, NPAR, LOU, IPAR, ISTART, KERR)
Searches a character string LINE from last to first character,
to create a substring IPAR containing NPAR blank-delimited numbers;
ISTART is the column of LINE containing IPAR. This allows format-
free input of combined alpha-numeric data. For example,

```

input: LINE*80 = "t1 t2 dt 300.0 3.0E3 50"
      NPAR      = 3, the number of substrings requested
      LOU       = 6, a logical unit number on which to write
                diagnostic messages.
output: IPAR*80 = "300.0 3.0E3 50"
      ISTART    = 13, the starting column in LINE of the
                NPAR substrings
      KERR      = .FALSE.

```

INPUT

LINE - Character string; length determined by calling routine.
NPAR - Integer scalar, number of substrings expected.
LOU - Integer scalar, output unit for printed diagnostics.

OUTPUT

IPAR - Character string, subset of LINE, containing only the
NPAR substrings.
ISTART - Integer scalar, starting location in LINE of the NPAR
substrings.
KERR - Logical, character length or syntax error flag.

```

CKNU      CKNU      CKNU      CKNU      CKNU      CKNU      CKNU
*****
*****
*****

```

SUBROUTINE CKNU (KDIM, ICKWRK, RCKWRK, NUKI)
Returns the stoichiometric coefficients of the reactions;
see Eq. (50).

INPUT

KDIM - Integer scalar, first dimension of the matrix NUKI;
KDIM must be greater than or equal to KK, the total
species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

NUKI(*,*) - Integer matrix, stoichiometric coefficients of the
species in the reactions; dimension at least KK for
the first, the total species count, and at least II
for the second, the total reaction count.
NUKI(K,I) is the stoichiometric coefficient of
species K in reaction I.

```

CKNUF     CKNUF     CKNUF     CKNUF     CKNUF     CKNUF     CKNUF
*****
*****
*****

```

SUBROUTINE CKNUF (KDIM, ICKWRK, RCKWRK, NUKI)
Returns the forward stoichiometric coefficients for reactions;
by definition, reactants' coefficients are negative; see Eq. (50).
Contrast this subroutine with subroutine CKNU, which returns the
net stoichiometric coefficients for a reaction.

INPUT

KDIM - Integer scalar, first dimension of the matrix NUKI;
KDIM must be greater than or equal to KK, the total
species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

NUFKI(*,*) - Integer matrix, stoichiometric coefficients of the
species in the forward direction of the reactions
(reactants only); dimension at least KK in the first,
the total species count, and at least II for the
second, the total reaction count.
NUFKI(K,I) is the stoichiometric coefficient of
species K in forward direction of reaction I.

```

CKPC      CKPC      CKPC      CKPC      CKPC      CKPC      CKPC
*****
*****
*****

```

SUBROUTINE CKPC (RHO, T, C, ICKWRK, RCKWRK, P)
Returns the pressure of the gas mixture given mass density,
temperature(s) and molar concentrations; see Eq. (1).

```

INPUT
RHO      - Real scalar, mass density.
          cgs units, gm/cm**3
T(*)     - Real array, temperature(s); dimension is determined by
          the application program to be the total number of
          species temperatures, nominally 1.
          cgs units, K
C(*)     - Real array, concentrations of the species;
          dimension at least KK, the total species count.
          cgs units, mole/cm**3
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
P        - Real scalar, pressure.
          cgs units, dynes/cm**2

```

```

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

```

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

```

INPUT
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
KPHASE(*) - Integer array, phases of the species;
          dimension at least KK, the total species count.
          KPHASE(K)=-1, species K is solid
          KPHASE(K)= 0, species K is gaseous
          KPHASE(K)=+1, species K is liquid

```

```

CKPNT      CKPNT      CKPNT      CKPNT      CKPNT      CKPNT      CKPNT
*****
*****
*****

```

```

SUBROUTINE CKPNT (LSAVE, LOUT, NPOINT, VERS, PREC, LENI, LENR,
                 LENC, KERR)

```

Reads from a file information about a CHEMKIN linkfile, and pointers for work arrays.

INPUT

```

LSAVE      - Integer scalar, input unit for binary data file.
LOUT       - Integer scalar, formatted output file unit number.

```

OUTPUT

```

NPOINT     - Integer scalar, total pointers count.
VERS       - Real scalar, version number of the CHEMKIN linkfile.
PREC       - Character string, machine precision of the linkfile.
LENI       - Integer scalar, length required for integer work array.
LENR       - Integer scalar, length required for real work array.
LENC       - Integer scalar, length required for character work array.
KERR       - Logical, error flag.

```

```

CKPX      CKPX      CKPX      CKPX      CKPX      CKPX      CKPX
*****
*****
*****

```

```

SUBROUTINE CKPX (RHO, T, X, ICKWRK, RCKWRK, P)

```

Returns the pressure of the gas mixture given mass density, temperature(s) and mole fractions; see Eq. (1).

INPUT

```

RHO        - Real scalar, mass density.
             cgs units, gm/cm**3
T(*)       - Real array, temperature(s); dimension is determined by
             the application program to be the total number of
             species temperatures, nominally 1.
             cgs units, K
X(*)       - Real array, mole fractions of the mixture;
             dimension at least KK, the total species count.
ICKWRK(*)  - Integer workspace array; dimension at least LENICK.
RCKWRK(*)  - Real workspace array; dimension at least LENRCK.

```

OUTPUT

```

P          - Real scalar, pressure.
             cgs units, dynes/cm**2

```

```

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

```

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

INPUT
RHO - Real scalar, mass density.
 cgs units, gm/cm**3
T(*) - Real array, temperature(s); dimension is determined by
 the application program to be the total number of
 species temperatures, nominally 1.
 cgs units, K
Y(*) - Real array, mass fractions of the mixture;
 dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
P - Real scalar, pressure.
 cgs units, dynes/cm**2

```

CKQC      CKQC      CKQC      CKQC      CKQC      CKQC      CKQC
*****
*****
*****

```

SUBROUTINE CKQC (T, C, ICKWRK, RCKWRK, Q)
Returns the rates of progress for reactions given temperature(s)
and molar concentrations; see Eqs. (51) and (63).

INPUT
T(*) - Real array, temperature(s); dimension is determined by
 the application program to be the total number of
 species temperatures, nominally 1.
 cgs units, K
C(*) - Real array, concentrations of the species;
 dimension at least KK, the total species count.
 cgs units, mole/cm**3
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
Q(*) - Real array, rates of progress for reactions;
 dimension at least II, the total reaction count.
 cgs units, moles/(cm**3*sec)

```

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

```

SUBROUTINE CKQXP (P, T, X, ICKWRK, RCKWRK, Q)
Returns the rates of progress for reactions given pressure,
temperature(s) and mole fractions; see Eqs. (51) and (63).

INPUT
P - Real scalar, pressure.
cgs units, dynes/cm**2
T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
X(*) - Real array, mole fractions of the mixture;
dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
Q(*) - Real array, rates of progress for reactions;
dimension at least II, the total reaction count.
cgs units, moles/(cm**3*sec)

```

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

```

SUBROUTINE CKQYP (P, T, Y, ICKWRK, RCKWRK, Q)
Returns the rates of progress for reactions given pressure,
temperature(s) and mass fractions; see Eqs. (51) and (63).

INPUT
P - Real scalar, pressure.
cgs units, dynes/cm**2
T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
Y(*) - Real array, Mass fractions of the mixture;
dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
Q(*) - Real array, rates of progress for reactions;
dimension at least II, the total reaction count.
cgs units, moles/(cm**3*sec)

```

CKQYR      CKQYR      CKQYR      CKQYR      CKQYR      CKQYR      CKQYR
*****
*****
*****

```

SUBROUTINE CKQYR (RHO, T, Y, ICKWRK, RCKWRK, Q)
Returns the rates of progress for reactions given mass density,
temperature(s) and mass fractions; see Eqs. (51) and (63).

INPUT

RHO - Real scalar, mass density.
cgs units, gm/cm**3

T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K

Y(*) - Real array, mass fractions of the mixture;
dimension at least KK, the total species count.

ICKWRK(*) - Integer workspace array; dimension at least LENICK.

RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

Q(*) - Real array, rates of progress for reactions;
dimension at least II, the total reaction count.
cgs units, moles/(cm**3*sec)

```

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 - Real scalar, to be converted to a string;
the maximum magnitude of RNUM is machine-dependent.

OUTPUT

STR - Character string, left-justified representation of RNUM;
i.e., RNUM= 0.0 returns STR=" 0.00"
RNUM= -10.5 returns STR="-1.05E+01"
RNUM= 1.86E-100 returns in STR=" 1.86E-100"
the minimum length of STR required is 5

I - Integer scalar, total non-blank characters in RNUM.

KERR - Logical, character length error flag.

```

CKRAEX      CKRAEX      CKRAEX      CKRAEX      CKRAEX      CKRAEX      CKRAEX
*****
*****
*****

```

```

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

```

```

INPUT
I          - Integer scalar, reaction index;
            I > 0 gets RA(I) from RCKWRK
            I < 0 puts RA(I) into RCKWRK
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real      workspace array; dimension at least LENRCK.
If I < 1:
RA         - Real scalar, pre-exponential coefficient for reaction I.
            cgs units, mole-cm-sec-K

```

```

OUTPUT
If I > 1:
RA         - Real scalar, pre-exponential coefficient for reaction I.
            cgs units, mole-cm-sec-K

```

```

CKRDEX      CKRDEX      CKRDEX      CKRDEX      CKRDEX      CKRDEX      CKRDEX
*****
*****
*****

```

```

SUBROUTINE CKRDEX (I, RCKWRK, RD)*
Get/put the perturbation factor of the Ith reaction

```

```

INPUT
I          - Integer scalar, reaction index;
            I > 0 gets RD(I) from RCKWRK
            I < 0 puts RD(I) into RCKWRK
RCKWRK(*) - Real      workspace array; dimension at least LENRCK.
If I < 1:
RD         - Real scalar, perturbation factor for reaction I;
            cgs units, mole-cm-sec-K.

```

```

OUTPUT
If I > 1:
RD         - Real scalar, perturbation factor for reaction I;
            cgs units, mole-cm-sec-K.

```

```

CKRHEX      CKRHEX      CKRHEX      CKRHEX      CKRHEX      CKRHEX      CKRHEX
*****
*****
*****

```

SUBROUTINE CKRHEX (K, RCKWRK, A6)
Returns an array of the sixth thermodynamic polynomial coefficients for a species, or changes their value, depending on the sign of K.

INPUT
K - Integer scalar, species index;
K>0 gets A6(*) from RCKWRK,
K<0 puts A6(*) into RCKWRK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
A6(*) - Real array, the 6th thermodynamic polynomial coefficients for species K, over the number of fit temperature ranges; dimension at least (MXTP-1), where MXTP is the maximum number of temperatures used to divide the thermodynamic fits.

```

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

```

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

INPUT
P - Real scalar, pressure.
cgs units, dynes/cm**2
T(*) - Real array, temperature(s); dimension is determined by the application program to be the total number of species temperatures, nominally 1.
cgs units, K
C(*) - Real array, concentrations of the species;
dimension at least KK, the total species count.
cgs units, mole/cm**3
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
RHO - Real scalar, mass density.
cgs units, gm/cm**3

```

CKRHOX      CKRHOX      CKRHOX      CKRHOX      CKRHOX      CKRHOX      CKRHOX
*****
*****
*****

```

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

```

INPUT
P          - Real scalar, pressure.
             cgs units, dynes/cm**2
T(*)       - Real array, temperature(s); dimension is determined by
             the application program to be the total number of
             species temperatures, nominally 1.
             cgs units, K
X(*)       - Real array, mole fractions of the mixture;
             dimension at least KK, the total species count.
ICKWRK(*)  - Integer workspace array; dimension at least LENICK.
RCKWRK(*)  - Real workspace array; dimension at least LENRCK.

```

```

OUTPUT
RHO        - Real scalar, mass density.
             cgs units, gm/cm**3

```

```

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

```

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

```

INPUT
P          - Real scalar, pressure.
             cgs units, dynes/cm**2
T(*)       - Real array, temperature(s); dimension is determined by
             the application program to be the total number of
             species temperatures, nominally 1.
             cgs units, K
Y(*)       - Real array, mass fractions of the mixture;
             dimension at least KK, the total species count.
ICKWRK(*)  - Integer workspace array; dimension at least LENICK.
RCKWRK(*)  - Real workspace array; dimension at least LENRCK.

```

```

OUTPUT
RHO        - Real scalar, mass density.
             cgs units, gm/cm**3

```

```

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(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
RU - Real scalar, universal gas constant.
cgs units, 8.314510E7 ergs/(mole*K)
RUC - Real scalar, universal gas constant used only in
conjunction with activation energy.
preferred units, RU / 4.184 cal/(mole*K)
PA - Real scalar, pressure of one standard atmosphere.
cgs units, 1.01325E6 dynes/cm**2

```

CKSAVE   CKSAVE   CKSAVE   CKSAVE   CKSAVE   CKSAVE   CKSAVE
*****
*****
*****

```

SUBROUTINE CKSAVE (LOUT, LSAVE, ICKWRK, RCKWRK, CCKWRK)
Writes to a binary file information about a CHEMKIN linkfile,
pointers for the CHEMKIN Library, and CHEMKIN work arrays.

INPUT
LOUT - Integer scalar, formatted output file unit number.
LSAVE - Integer scalar, binary output file unit number.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.
CCKWRK(*) - Character string workspace array;
dimension at least LENCCK.

```

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
pressure, temperature(s) and mole fractions; see Eq. (42).

```

INPUT
P          - Real scalar, pressure.
            cgs units, dynes/cm**2
T(*)       - Real array, temperature(s); dimension is determined by
            the application program to be the total number of
            species temperatures, nominally 1.
            cgs units, K
X(*)       - Real array, mole fractions of the mixture;
            dimension at least KK, the total species count.
ICKWRK(*)  - Integer workspace array; dimension at least LENICK.
RCKWRK(*)  - Real workspace array; dimension at least LENRCK.

```

```

OUTPUT
SBML       - Real scalar, mean entropy.
            cgs units, ergs/(mole*K)

```

```

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 pressure,
temperature(s) and mass fractions; see Eq. (43).

```

INPUT
P          - Real scalar, pressure.
            cgs units, dynes/cm**2
T(*)       - Real array, temperature(s); dimension is determined by
            the application program to be the total number of
            species temperatures, nominally 1.
            cgs units, K
Y(*)       - Real array, mass fractions of the mixture;
            dimension at least KK, the total species count.
ICKWRK(*)  - Integer workspace array; dimension at least LENICK.
RCKWRK(*)  - Real workspace array; dimension at least LENRCK.

```

```

OUTPUT
SBMS       - Real scalar, mean entropy.
            cgs units, ergs/(gm*K)

```

```

CKSMH      CKSMH      CKSMH      CKSMH      CKSMH      CKSMH      CKSMH
*****
*****
*****

```

SUBROUTINE CKSMH (T, ICKWRK, RCKWRK, SMH)*
Returns the array of entropies minus enthalpies for species.
It is normally not called directly by the user.

INPUT
T(*) - Real array, temepratures;
 cgs units, K
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
SMH(*) - Real array, entropy minus enthalpy for species,
 SMH(K) = S(K)/R - H(K)/RT;
 dimension at least KK, the total species count.

```

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

```

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

INPUT
T(*) - Real array, temperature(s); dimension is determined by
 the application program to be the total number of
 species temperatures, nominally 1.
 cgs units, K
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
SML(*) - Real array, standard state entropies for species;
 dimension at least KK, the total species count.
 cgs units, ergs/(mole*K)

```

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(*) - Real array, temperature(s); dimension is determined by the application program to be the total number of species temperatures, nominally 1.
cgs units, K
- ICKWRK(*) - Integer workspace array; dimension at least LENICK.
- RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

- SMS(*) - Real array, standard state entropies for species; dimension at least KK, the total species count.
cgs units, ergs/(gm*K)

```

CKSNUM      CKSNUM      CKSNUM      CKSNUM      CKSNUM      CKSNUM      CKSNUM
*****
*****
*****

```

```

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

```

Search a character string, LINE, for (1) a character substring which may also appear in an array of character substrings KRAY, and (2) some number of character substrings representing numbers. In the case of (1), if the character substring appears in KRAY, KNUM is its index position.

In the case of (2), the character substrings are converted to NVAL real numbers and stored in RVAL, until NEXP are converted. This allows format-free input of combined alpha-numeric data. For example, 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      - Character string; length depends on calling routine.
NEXP      - Integer scalar, number of values to be found in LINE.
           If NEXP < 0, then IABS(NEXP) values are expected, but
           it is not an error condition if less values are found.
LOUT      - Integer scalar, formatted output file unit.
KRAY(*)   - Character string array.
NN        - Integer scalar, total number of character strings
           in KRAY.

```

OUTPUT

```

KNUM      - Integer scalar, index of character string in KRAY
           which corresponds to the first substring in LINE.
NVAL      - Integer scalar, count of real values found in LINE.
RVAL(*)   - Real array, real values found in LINE; dimension at least
           NEXP.
KERR      - Logical, syntax or dimensioning error flag;
           corresponding string not found, or total of
           values found is not the number of values expected,
           will result in KERR = .TRUE.

```

```

CKSOR      CKSOR      CKSOR      CKSOR      CKSOR      CKSOR      CKSOR
*****
*****
*****

```

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

INPUT
T(*) - Real array, temperature(s); dimension is determined by the application program to be the total number of species temperatures, nominally 1.
cgs units, K
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
SOR(*) - Real array, nondimensional entropies for species; dimension at least KK, the total species count.

```

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

```

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

INPUT
LINE - Character string; length determined by calling routine.
LOU - Integer scalar, formatted output file unit.
NDIM - Integer scalar, dimension of a character string array.

OUTPUT
SUB(*) - Character string array, the character substrings of LINE; dimension SUB at least NDIM.
NFOUND - Integer scalar, count of substrings found in LINE.
KERR - Logical, error flag; dimensioning errors will result in KERR = .TRUE.

```

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

```

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

INPUT
CCKWRK(*) - Character string workspace array; dimension at least LENCK.
LOU - Integer scalar, formatted output file unit.

OUTPUT
ENAME(*) - Character string array, element names; dimension at least MM, the total element count.
KERR - Logical, character length error flag.

```

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 - Integer scalar, reaction index.
LOUT - Integer scalar, formatted output file unit.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.
CCKWRK(*) - Character string workspace array;
dimension at least LENCCK.

OUTPUT
ISTR - Character string, description of reaction I.
LT - Integer scalar, number of non-blank characters in ISTR.
KERR - Logical, character length error flag.

```

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

```

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

INPUT
CCKWRK(*) - Character string workspace array;
dimension at least LENRCK.
LOUT - Integer scalar, formatted output file unit.

OUTPUT
KNAME(*) - Character string array, species names;
dimension at least KK, the total species count.
KERR - Logical, character length error flag.

```

CKTHB      CKTHB      CKTHB      CKTHB      CKTHB      CKTHB      CKTHB
*****
*****
*****

```

SUBROUTINE CKTHB (KDIM, ICKWRK, RCKWRK, AKI)
Returns matrix of enhanced third body coefficients; see Eq. (63).

INPUT
KDIM - Integer scalar, first dimension of the matrix AKI;
 KDIM must be at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
AKI(*,*) - Real matrix, enhanced third body efficiencies of the
 species in the reactions;
 dimension at least KK for first, the total species count,
 and at least II for the second, the total reaction count.
 AKI(K,I) is the enhanced efficiency of species K in
 reaction I.

```

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(*) - Real array, temperature(s); dimension is determined by
 the application program to be the total number of
 species temperatures, nominally 1.
 cgs units, K
X(*) - Real array, mole fractions of the mixture;
 dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
UBML - Real scalar, mean internal energy.
 cgs units, ergs/mole

```

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(*)        - Real array, temperature(s); dimension is determined by
              the application program to be the total number of
              species temperatures, nominally 1.
              cgs units, K
Y(*)        - Real array, mass fractions of the mixture;
              dimension at least KK, the total species count.
ICKWRK(*)   - Integer workspace array; dimension at least LENICK.
RCKWRK(*)   - Real    workspace array; dimension at least LENRCK.

OUTPUT
UBMS        - Real scalar, mean internal energy.
              cgs units, ergs/gm

```

```

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(*)        - Real array, temperature(s); dimension is determined by
              the application program to be the total number of
              species temperatures, nominally 1.
              cgs units, K
ICKWRK(*)   - Integer workspace array; dimension at least LENICK.
RCKWRK(*)   - Real    workspace array; dimension at least LENRCK.

OUTPUT
UML(*)      - Real array, internal energies for species;
              dimension at least KK, the total species count.
              cgs units, ergs/mole

```

```

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(*) - Real array, temperature(s); dimension is determined by the application program to be the total number of species temperatures, nominally 1.
cgs units, K
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
UMS(*) - Real array, internal energies for species; dimension at least KK, the total species count.
cgs units, ergs/gm

```

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

```

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

INPUT
T(*) - Real array, temperature(s); dimension is determined by the application program to be the total number of species temperatures, nominally 1.
cgs units, K
C(*) - Real array, concentrations of the species; dimension at least KK, the total species count.
cgs units, mole/cm**3
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
WDOT(*) - Real array, chemical production rates of the species; dimension at least KK, the total species count.
cgs units, moles/(cm**3*sec)

```

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(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
WL(*) - Real array, radiation wavelengths for reactions;
dimension at least II, total reaction count.
cgs units, angstrom.
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;

```

CKWT      CKWT      CKWT      CKWT      CKWT      CKWT      CKWT
*****
*****
*****

```

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

INPUT
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
WT(*) - Real array, molecular weights of the species;
dimension at least KK, the total species count.
cgs units, gm/mole

```

CKWXP      CKWXP      CKWXP      CKWXP      CKWXP      CKWXP      CKWXP
*****
*****
*****

```

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

INPUT

P - Real scalar, pressure.
cgs units, dynes/cm**2

T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K

X(*) - Real array, mole fractions of the mixture;
dimension at least KK, the total species count.

ICKWRK(*) - Integer workspace array; dimension at least LENICK.

RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

WDOT(*) - Real array, chemical production rates of the species;
dimension at least KK, the total species count.
cgs units, moles/(cm**3*sec)

```

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

```

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

INPUT

RHO - Real scalar, mass density.
cgs units, gm/cm**3

T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K

X(*) - Real array, mole fractions of the mixture;
dimension at least KK, the total species count.

ICKWRK(*) - Integer workspace array; dimension at least LENICK.

RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT

WDOT(*) - Real array, chemical production rates of the species;
dimension at least KK, the total species count.
cgs units, moles/(cm**3*sec)

```

CKWYP      CKWYP      CKWYP      CKWYP      CKWYP      CKWYP      CKWYP
*****
*****
*****

```

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

INPUT
P - Real scalar, pressure.
cgs units, dynes/cm**2
T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
Y(*) - Real array, mass fractions of the mixture;
dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
WDOT(*) - Real array, chemical production rates of the species;
dimension at least KK, the total species count.
cgs units, moles/(cm**3*sec)

```

CKWYPK     CKWYPK     CKWYPK     CKWYPK     CKWYPK     CKWYPK     CKWYPK
*****
*****
*****

```

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

INPUT
P - Real scalar, pressure.
cgs units, dynes/cm**2
T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
Y(*) - Real array, mass fractions of the mixture;
dimension at least KK, the total species count.
RKFT(*) - Real array, forward reaction rates for reactions;
dimension at least II, the total reaction count.
cgs units, depends on the reaction
RKRT(*) - Real array, reverse reaction rates for reactions;
dimension at least II, the total reaction count.
cgs units, depends on the reaction
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
WDOT(*) - Real array, chemical production rates of the species;
dimension at least KK, the total species count.
cgs units, moles/(cm**3*sec)

```

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

```

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

```

INPUT
RHO      - Real scalar, mass density.
          cgs units, gm/cm**3
T(*)     - Real array, temperature;
          cgs units, K
Y(*)     - Real array, mass fractions of the mixture;
          dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
WDOT(*)  - Real array, chemical production rates of the species;
          dimension at least KK, the total species count.
          cgs units, moles/(cm**3*sec)

```

```

CKXNUM    CKXNUM    CKXNUM    CKXNUM    CKXNUM    CKXNUM    CKXNUM
*****
*****
*****

```

SUBROUTINE CKXNUM (LINE, NEXP, LOUT, NVAL, RVAL, KERR)
Searches a character string, LINE, for NEXP space-delimited substrings representing numbers, until NVAL real values are converted and stored in the array, RVAL.

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     - Character string, length established by calling program.
NEXP     - Integer scalar, number of real values to be found in LINE;
          If NEXP < 0 then IABS(NEXP) values are expected, but
          it is not an error condition if fewer values are found.
LOUT     - Integer scalar, output unit for printed diagnostics.

OUTPUT
NVAL     - Integer scalar, count of real values found in LINE.
RVAL     - Real array, values converted from characters in LINE;
          dimension at least NEXP.
KERR     - Logical, syntax or dimensioning error flag.

```

```

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

```

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

```

INPUT
P          - Real scalar, pressure.
            cgs units, dynes/cm**2
T(*)      - Real array, temperature(s); dimension is determined by
            the application program to be the total number of
            species temperatures, nominally 1.
            cgs units, K
X(*)      - Real array, mole fractions of the mixture;
            dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
C(*)      - Real array, concentrations of the species;
            dimension at least KK, the total species count.
            cgs units, mole/cm**3

```

```

CKXTCR      CKXTCR      CKXTCR      CKXTCR      CKXTCR      CKXTCR      CKXTCR
*****
*****
*****

```

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

```

INPUT
RHO       - Real scalar, mass density.
            cgs units, gm/cm**3
T(*)      - Real array, temperature(s); dimension is determined by
            the application program to be the total number of
            species temperatures, nominally 1.
            cgs units, K
X(*)      - Real array, mole fractions of the mixture;
            dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
C(*)      - Real array, concentrations of the species;
            dimension at least KK, the total species count.
            cgs units, mole/cm**3

```

```

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

```

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

INPUT
X(*) - Real array, mole fractions of the mixture;
dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
Y(*) - Real array, mass fractions of the mixture;
dimension at least KK, the total species count.

```

CKYTCP      CKYTCP      CKYTCP      CKYTCP      CKYTCP      CKYTCP      CKYTCP
*****
*****
*****

```

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

INPUT
P - Real scalar, pressure.
cgs units, dynes/cm**2
T(*) - Real array, temperature(s); dimension is determined by
the application program to be the total number of
species temperatures, nominally 1.
cgs units, K
Y(*) - Real array, mass fractions of the mixture;
dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real workspace array; dimension at least LENRCK.

OUTPUT
C(*) - Real array, concentrations of the species;
dimension at least KK, the total species count.
cgs units, mole/cm**3

```

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

```

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

```

INPUT
RHO      - Real scalar, mass density.
          cgs units, gm/cm**3
T(*)     - Real array, temperature(s); dimension is determined by
          the application program to be the total number of
          species temperatures, nominally 1.
          cgs units, K
Y(*)     - Real array, mass fractions of the mixture;
          dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real      workspace array; dimension at least LENRCK.

OUTPUT
C(*)     - Real array, concentrations of the species;
          dimension at least KK, the total species count.
          cgs units, mole/cm**3

```

```

CKYTX      CKYTX      CKYTX      CKYTX      CKYTX      CKYTX      CKYTX
*****
*****
*****

```

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

```

INPUT
Y(*)     - Real array, mass fractions of the mixture;
          dimension at least KK, the total species count.
ICKWRK(*) - Integer workspace array; dimension at least LENICK.
RCKWRK(*) - Real      workspace array; dimension at least LENRCK.

OUTPUT
X(*)     - Real array, mole fractions of the mixture;
          dimension at least KK, the total species count.

```

```

PKINDX     PKINDX     PKINDX     PKINDX     PKINDX     PKINDX     PKINDX
*****
*****
*****

```

SUBROUTINE PKINDX (ICKWRK, KELECT, KKION)
Returns plasma indices for the particular reaction mechanism.

```

INPUT
ICKWRK(*) - Integer workspace array; dimension at least LENICK.

OUTPUT
KELECT    - Integer scalar, species array index for the electron.
KKION     - Integer scalar, total ion count.

```

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

$$\frac{dT}{dt} = -\frac{1}{\rho \bar{c}_p} \sum_{k=1}^K h_k \dot{\omega}_k W_k,$$
$$\frac{dY_k}{dt} = \frac{\dot{\omega}_k W_k}{\rho}, \quad k = 1, \dots, K.$$

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 VODE.¹⁷ We find this code to be highly reliable for the solution of a wide range of stiff initial-value problems.

The FORTRAN code for the 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 VODE to obtain the solution at uniform print intervals. The governing equation formulation is found in SUBROUTINE FUN, which is called by VODE.

The sections below present a sample shell script containing the UNIX command procedure for the sample problem, CHEMKIN Interpreter input and output, the input FORTRAN code, called CONP, and output for the CONP execution. The last section describes how to use VODE.

1. UNIX Shell Script for Running the Sample Problem

```
#!/bin/sh
# to execute:  sh comp.sh logname &

sh 1> ${1}.log 2>&1 << ENDSH

set -x

#cd /scr/$LOGNAME          #go to user's scratch directory
#mkdir "${1}$$"           #make subdirectory /myrun##
#cd "${1}$$"              #go to /myrun##

cat << EOF > makefile

include chemmake.h

OBJS = comp.o cklib.o vode.o math.o
INPS = therm.dat chem.inp comp.inp
OUTS = chem.out chem.bin comp.out
EXES = chem.exe comp.exe

chem.exe: ckinterp.o
        $(LINK) chem.exe ckinterp.o

comp.exe: $(OBJS)
        $(LINK) comp.exe $(OBJS)
EOF

cat << EOF > chem.inp
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
  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
EOF
```

```
cat << EOF > comp.inp
1 1000
H2 1
O2 3
N2 .1
END
3.0E-4 3.0E-5
EOF

touch makefile; make chem.exe comp.exe
chem.exe <chem.inp >chem.out
comp.exe < comp.inp > comp.out

ENDSH
```

2. Sample Input to the 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
  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 the Interpreter for the Sample Input

CHEMKIN-III GAS-PHASE MECHANISM INTERPRETER:
 DOUBLE PRECISION Vers. 5.0 March 1, 1996
 Copyright 1995, Sandia Corporation.
 The U.S. Government retains a limited license in this software.

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

SPECIES CONSIDERED	S G		MOLECULAR WEIGHT	TEMPERATURE		ELEMENT COUNT		
	E	E		LOW	HIGH	H	O	N
1. H2	G	0	2.01594	300	5000	2	0	0
2. H	G	0	1.00797	300	5000	1	0	0
3. O2	G	0	31.99880	300	5000	0	2	0
4. O	G	0	15.99940	300	5000	0	1	0
5. OH	G	0	17.00737	300	5000	1	1	0
6. HO2	G	0	33.00677	300	5000	1	2	0
7. H2O2	G	0	34.01474	300	5000	2	2	0
8. H2O	G	0	18.01534	300	5000	2	1	0
9. N	G	0	14.00670	300	5000	0	0	1
10. N2	G	0	28.01340	300	5000	0	0	2
11. NO	G	0	30.00610	300	5000	0	1	1

REACTIONS CONSIDERED	(k = A T**b exp(-E/RT))		
	A	b	E
1. H2+O2=2OH	1.70E+13	0.0	47780.0
2. OH+H2=H2O+H	1.17E+09	1.3	3626.0
3. O+OH=O2+H	4.00E+14	-0.5	0.0
4. O+H2=OH+H	5.06E+04	2.7	6290.0
5. H+O2+M=HO2+M	3.61E+17	-0.7	0.0
H2O	Enhanced by	1.860E+01	
H2	Enhanced by	2.860E+00	
N2	Enhanced by	1.260E+00	
6. OH+HO2=H2O+O2	7.50E+12	0.0	0.0
7. H+HO2=2OH	1.40E+14	0.0	1073.0
8. O+HO2=O2+OH	1.40E+13	0.0	1073.0
9. 2OH=O+H2O	6.00E+08	1.3	0.0
10. H+H+M=H2+M	1.00E+18	-1.0	0.0
H2O	Enhanced by	0.000E+00	
H2	Enhanced by	0.000E+00	
11. H+H+H2=H2+H2	9.20E+16	-0.6	0.0
12. H+H+H2O=H2+H2O	6.00E+19	-1.2	0.0
13. H+OH+M=H2O+M	1.60E+22	-2.0	0.0
H2O	Enhanced by	5.000E+00	
14. H+O+M=OH+M	6.20E+16	-0.6	0.0
H2O	Enhanced by	5.000E+00	

15.	$O+O+M=O_2+M$	1.89E+13	0.0	-1788.0
16.	$H+HO_2=H_2+O_2$	1.25E+13	0.0	0.0
17.	$HO_2+HO_2=H_2O_2+O_2$	2.00E+12	0.0	0.0
18.	$H_2O_2+M=OH+OH+M$	1.30E+17	0.0	45500.0
19.	$H_2O_2+H=HO_2+H_2$	1.60E+12	0.0	3800.0
20.	$H_2O_2+OH=H_2O+HO_2$	1.00E+13	0.0	1800.0
21.	$O+N_2=NO+N$	1.40E+14	0.0	75800.0
22.	$N+O_2=NO+O$	6.40E+09	1.0	6280.0
23.	$OH+N=NO+H$	4.00E+13	0.0	0.0

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

NO ERRORS FOUND ON INPUT,
 BINARY Vers. 1.0 CHEMKIN linkfile chem.bin written.

WORKING SPACE REQUIREMENTS ARE

INTEGER:	770
REAL:	538
CHARACTER:	14

4. Sample User's FORTRAN Code: CONP

```
PROGRAM CONP
C
C   Integration of adiabatic, constant pressure kinetics problems
C
C   VERSION 1.2:
C   1. Implement new VODE solver
C   VERSION 1.3:
C   1. Add IFLAG to CKLEN, CKINIT argument list.
C   VERSION 1.4:
C   1. Change character index "(:" to "(1:"
C****precision > double
      IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER(I-N)
C****END precision > double
C****precision > single
      IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
C****END precision > single
C
      PARAMETER (LENIWK=4000, LENRWK=4000, LENCWK=500, NK=5, NLMAX=55,
1          LIN=5, LOUT=6, LINKCK=25, KMAX=50, ITOL=1, IOPT=0,
2          RTOL=1.0E-6, ITASK=1, ATOL=1.0E-15)
C
      DIMENSION IWORK(LENIWK), RWORK(LENRWK), X(KMAX), Z(KMAX)
      CHARACTER*16 CWORK(LENCWK), KSYM(KMAX), PRVERS, PRDATE, PREC
      CHARACTER LINE*80
      LOGICAL KERR, IERR
      EXTERNAL FUN
      COMMON /ICONS/ KK, NP, NWT, NH, NWDOT
C
      DATA KERR/.FALSE./, X/KMAX*0.0/, KSYM/KMAX*' '//,
1          PRVERS/'1.0'//, PRDATE/'March 1, 1996'//
C
C****precision > double
      PREC = 'DOUBLE'
C****END precision > double
C****precision > single
      PREC = 'SINGLE'
C****END precision > single
C
      WRITE (LOUT, '( /A, /1XA, A, A, A, /A, /A, //)')
1' CONP: CHEMKIN-III Constant Pressure Kinetics Code,',
2  PREC(1:ILASCH(PREC)), ' PRECISION VERS. ',
3  PRVERS(1:ILASCH(PRVERS)+1), PRDATE,
4  ' Copyright 1995, Sandia Corporation.',
5' The U.S. Government retains a limited license in this software.'
C
C   Initialize CHEMKIN
C****gas linkfile > binary
      OPEN (LINKCK, FORM='UNFORMATTED', STATUS='UNKNOWN',
1          FILE='chem.bin')
C****END gas linkfile > binary
C****gas linkfile > ascii
      OPEN (LINKCK, FORM='FORMATTED', STATUS='UNKNOWN',
C          1          FILE='chem.dat')
C****END gas linkfile > ascii
C
      CALL CKLEN (LINKCK, LOUT, LENI, LENR, LENC, IFLAG)
      IF (IFLAG .GT. 0) STOP
      CALL CKINIT (LENIWK, LENRWK, LENCWK, LINKCK, LOUT, IWORK,
1          RWORK, CWORK, IFLAG)
      IF (IFLAG .GT. 0) STOP
      CLOSE (LINKCK)
      CALL CKINDX (IWORK, RWORK, MM, KK, II, NFIT)
C
      NEQ = KK + 1
      LRW = 22 + 9*NEQ + 2*NEQ**2
      NVODE = LENR + 1
      NP = NVODE + LRW
      NWT = NP + 1
      NH = NWT + KK
      NWDOT = NH + KK
```

```

      NTOT = NWDOT+ KK - 1
C
      LIW = 30 + NEQ
      IVODE = LENI + 1
      ITOT = IVODE + LIW - 1
C
      IF (KK.GT.KMAX .OR. LENRWK.LT.NTOT .OR. LENIWK.LT.ITOT) THEN
        IF (KK .GT. KMAX) WRITE (LOUT, *)
1      ' Error...KMAX too small...must be at least ',KK
        IF (LENRWK .LT. NTOT) WRITE (LOUT, *)
1      ' Error...LENRWK too small...must be at least', NTOT
        IF (LENIWK .LT. ITOT) WRITE (LOUT, *)
1      ' Error...LENIWK too small...must be at least', ITOT
      STOP
      ENDIF
C
      CALL CKSYMS (CWORK, LOUT, KSYM, IERR)
      IF (IERR) KERR = .TRUE.
      CALL CKWT (IWORK, RWORK, RWORK(NWT))
      CALL CKRP (IWORK, RWORK, RU, RUC, PATM)
C
      Pressure and temperature
C
      WRITE (LOUT, '(//A//A)')
1' ADIABATIC FIXED PRESSURE PROBLEM,'
2' INPUT PRESSURE(ATM) AND TEMPERATURE(K):'
      READ (LIN, *) PA, T
      WRITE (LOUT,7105) PA, T
      RWORK(NP) = PA*PATM
C
      Initial non-zero moles
C
40 CONTINUE
      LINE = ' '
      WRITE (LOUT, '(//A)') ' INPUT MOLES OF NEXT SPECIES '
      READ (LIN, '(A)', END=45) LINE
      WRITE (LOUT, '(1X,A)') LINE
      ILEN = INDEX (LINE, '!')
      IF (ILEN .EQ. 1) GO TO 40
C
      ILEN = ILEN - 1
      IF (ILEN .LE. 0) ILEN = LEN(LINE)
      IF (INDEX(LINE(1:ILEN), 'END') .EQ. 0) THEN
        IF (LINE(1:ILEN) .NE. ' ') THEN
          CALL CKSNUM (LINE(1:ILEN), 1, LOUT, KSYM, KK, KNUM,
1          NVAL, VAL, IERR)
          IF (IERR) THEN
            WRITE (LOUT,*) ' Error reading moles...'
            KERR = .TRUE.
          ELSE
            X(KNUM) = VAL
          ENDIF
        ENDIF
      ENDIF
      GO TO 40
      ENDIF
C
45 CONTINUE
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
      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, IWORK, RWORK, Z(2))
C
C    Integration control parameters for VODE
C
        TT2 = TT1
        MF = 22
        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)(1:10), K=1,MIN(NK,KK))
        NLINES = 1
C
        DO 200 K1 = NK+1, KK, NK
            WRITE (LOUT, 7110) (KSYM(K)(1:10), 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), IWORK, RWORK, 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
C*****precision > single
C    CALL SVODE
C*****END precision > single
C*****precision > double
C    CALL DVODE
C*****END precision > double
*          (FUN, NEQ, Z, TT1, TT2, ITOL, RTOL, ATOL, ITASK,
1          ISTATE, IOPT, RWORK(NVODE), LRW, IWORK(IVODE),
2          LIW, JAC, MF, RWORK, IWORK)
C
        IF (ISTATE .LE. -2) THEN
            IF (ISTATE .EQ. -1) THEN
                ISTATE = 2
                GO TO 350
            ELSE
                WRITE (LOUT,*) ' ISTATE=',ISTATE
                STOP
            ENDIF
        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, RPAR, IPAR)
C
C*****precision > double
      IMPLICIT DOUBLE PRECISION(A-H,O-Z), INTEGER(I-N)
C*****END precision > double
C*****precision > single
C      IMPLICIT REAL (A-H,O-Z), INTEGER(I-N)
C*****END precision > single
C
      COMMON /ICONS/ KK, NP, NWT, NH, NWDOT
      DIMENSION Z(*), ZP(*), RPAR(*), IPAR(*)
C
C      Variables in Z are:  Z(1)   = T
C                        Z(K+1) = Y(K)
C
C      Call CHEMKIN subroutines
C
      CALL CKRHOY (RPAR(NP), Z(1), Z(2), IPAR, RPAR, RHO)
      CALL CKCPBS (Z(1), Z(2), IPAR, RPAR, CPB)
      CALL CKWYP (RPAR(NP), Z(1), Z(2), IPAR, RPAR, RPAR(NWDOT))
      CALL CKHMS (Z(1), IPAR, RPAR, RPAR(NH))
C
C      Form governing equation
C
      SUM = 0.0
      DO 100 K = 1, KK
          H = RPAR(NH + K - 1)
          WDOT = RPAR(NWDOT + K - 1)
          WT = RPAR(NWT + K - 1)
          ZP(K+1) = WDOT * WT / RHO
          SUM = SUM + H * WDOT * WT
100 CONTINUE
      ZP(1) = -SUM / (RHO*CPB)
C
      RETURN
      END

```

5. Input to the Sample FORTRAN Code, CONP

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

6. Output form the Sample FORTRAN Code, CONP

CONP: CHEMKIN-III Constant Pressure Kinetics Code,
DOUBLE PRECISION VERS. 1.0 March 1, 1996
Copyright 1995, Sandia Corporation.
The U.S. Government retains a limited license in this software.

CKLIB: CHEMKIN-III GAS-PHASE CHEMICAL KINETICS LIBRARY,
DOUBLE PRECISION Vers. 5.0 March 1, 1996
Copyright 1995, Sandia Corporation.
The U.S. Government retains a limited license in this software.

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

1

T (SEC)	TMP (K)	H2 HO2 NO	H H2O2	O2 H2O	O N	OH N2
0.000E+00	0.100E+04	0.244E+00 0.000E+00 0.000E+00	0.000E+00 0.000E+00	0.732E+00 0.000E+00	0.000E+00 0.000E+00	0.000E+00 0.244E-01
0.300E-04	0.100E+04	0.244E+00 0.129E-04 0.375E-19	0.817E-05 0.103E-07	0.732E+00 0.259E-04	0.425E-05 0.181E-20	0.144E-05 0.244E-01
0.600E-04	0.196E+04	0.890E-02 0.174E-03 0.167E-07	0.169E-01 0.355E-04	0.625E+00 0.224E+00	0.570E-01 0.229E-09	0.411E-01 0.262E-01
0.900E-04	0.235E+04	0.367E-02 0.845E-04 0.163E-05	0.331E-02 0.445E-05	0.658E+00 0.246E+00	0.235E-01 0.193E-08	0.392E-01 0.271E-01
0.120E-03	0.243E+04	0.258E-02 0.693E-04 0.438E-05	0.185E-02 0.254E-05	0.665E+00 0.251E+00	0.165E-01 0.229E-08	0.352E-01 0.272E-01
0.150E-03	0.246E+04	0.216E-02 0.641E-04 0.730E-05	0.139E-02 0.197E-05	0.669E+00 0.254E+00	0.138E-01 0.236E-08	0.330E-01 0.273E-01
0.180E-03	0.248E+04	0.197E-02 0.619E-04 0.102E-04	0.120E-02 0.173E-05	0.670E+00 0.255E+00	0.125E-01 0.237E-08	0.319E-01 0.273E-01
0.210E-03	0.248E+04	0.188E-02 0.609E-04 0.131E-04	0.111E-02 0.162E-05	0.671E+00 0.255E+00	0.119E-01 0.238E-08	0.313E-01 0.273E-01
0.240E-03	0.249E+04	0.183E-02 0.604E-04 0.160E-04	0.106E-02 0.157E-05	0.671E+00 0.256E+00	0.116E-01 0.239E-08	0.310E-01 0.273E-01
0.270E-03	0.249E+04	0.181E-02 0.602E-04 0.188E-04	0.104E-02 0.154E-05	0.672E+00 0.256E+00	0.115E-01 0.240E-08	0.308E-01 0.273E-01
0.300E-03	0.249E+04	0.179E-02 0.600E-04 0.217E-04	0.103E-02 0.152E-05	0.672E+00 0.256E+00	0.114E-01 0.241E-08	0.307E-01 0.273E-01

7. Summary of VODE Math Library Usage

```
      SUBROUTINE DVODE (F, NEQ, Y, T, TOUT, ITOL, RTOL, ATOL, ITASK,  
1          ISTATE, IOPT, RWORK, LRW, IWORK, LIW, JAC, MF,  
2          RPAR, IPAR)  
      EXTERNAL F, JAC  
      DOUBLE PRECISION Y, T, TOUT, RTOL, ATOL, RWORK, RPAR  
      INTEGER NEQ, ITOL, ITASK, ISTATE, IOPT, LRW, IWORK, LIW,  
1          MF, IPAR  
      DIMENSION Y(*), RTOL(*), ATOL(*), RWORK(LRW), IWORK(LIW),  
1          RPAR(*), IPAR(*)
```

```
C-----  
C DVODE.. Variable-coefficient Ordinary Differential Equation solver,  
C with fixed-leading coefficient implementation.  
C This version is in double precision.  
C  
C DVODE solves the initial value problem for stiff or nonstiff  
C systems of first order ODEs,  
C  $dy/dt = f(t,y)$  , or, in component form,  
C  $dy(i)/dt = f(i) = f(i,t,y(1),y(2),\dots,y(NEQ))$  ( $i = 1,\dots,NEQ$ ).  
C DVODE is a package based on the EPISODE and EPISODEB packages, and  
C on the ODEPACK user interface standard, with minor modifications.  
C-----  
C Revision History (YYMMDD)  
C 890615 Date Written  
C 890922 Added interrupt/restart ability, minor changes throughout.  
C 910228 Minor revisions in line format, prologue, etc.  
C 920227 Modifications by D. Pang:  
C (1) Applied subgennam to get generic intrinsic names.  
C (2) Changed intrinsic names to generic in comments.  
C (3) Added *DECK lines before each routine.  
C 920721 Names of routines and labeled Common blocks changed, so as  
C to be unique in combined single/double precision code (ACH).  
C 920722 Minor revisions to prologue (ACH).  
C 920831 Conversion to double precision done (ACH).  
C-----  
C References..  
C  
C 1. P. N. Brown, G. D. Byrne, and A. C. Hindmarsh, "VODE: A Variable  
C Coefficient ODE Solver," SIAM J. Sci. Stat. Comput., 10 (1989),  
C pp. 1038-1051. Also, LLNL Report UCRL-98412, June 1988.  
C 2. G. D. Byrne and A. C. Hindmarsh, "A Polyalgorithm for the  
C Numerical Solution of Ordinary Differential Equations,"  
C ACM Trans. Math. Software, 1 (1975), pp. 71-96.  
C 3. A. C. Hindmarsh and G. D. Byrne, "EPISODE: An Effective Package  
C for the Integration of Systems of Ordinary Differential  
C Equations," LLNL Report UCID-30112, Rev. 1, April 1977.  
C 4. G. D. Byrne and A. C. Hindmarsh, "EPISODEB: An Experimental  
C Package for the Integration of Systems of Ordinary Differential  
C Equations with Banded Jacobians," LLNL Report UCID-30132, April  
C 1976.  
C 5. A. 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 6. K. R. Jackson and R. Sacks-Davis, "An Alternative Implementation  
C of Variable Step-Size Multistep Formulas for Stiff ODEs," ACM  
C Trans. Math. Software, 6 (1980), pp. 295-318.  
C-----
```

```

C Authors..
C
C           Peter N. Brown and Alan C. Hindmarsh
C           Computing and Mathematics Research Division, L-316
C           Lawrence Livermore National Laboratory
C           Livermore, CA 94550
C and
C           George D. Byrne
C           Exxon Research and Engineering Co.
C           Clinton Township
C           Route 22 East
C           Annandale, NJ 08801
C-----
C Summary of usage.
C
C Communication between the user and the DVODE 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
C           SUBROUTINE F (NEQ, T, Y, YDOT, RPAR, IPAR)
C           DOUBLE PRECISION T, Y, YDOT, RPAR
C           DIMENSION Y(NEQ), YDOT(NEQ)
C
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 (21, 22, 24, 25), and DVODE requires the Jacobian
C matrix in some form. In these cases (MF .gt. 0), DVODE will use a
C saved copy of the Jacobian matrix. If this is undesirable because of
C storage limitations, set MF to the corresponding negative value
C (-21, -22, -24, -25). (See full description of MF below.)
C The Jacobian matrix is regarded either as full (MF = 21 or 22),
C or banded (MF = 24 or 25). In the banded case, DVODE requires two
C half-bandwidth parameters ML and MU. These are, respectively, the
C widths of the lower and upper parts of the band, excluding the main
C diagonal. Thus the band consists of the locations (i,j) with
C i-ML .le. j .le. i+MU, and the full 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, DVODE 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
C           SUBROUTINE JAC (NEQ, T, Y, ML, MU, PD, NROWPD, RPAR, IPAR)
C           DOUBLE PRECISION T, Y, PD, RPAR
C           DIMENSION Y(NEQ), PD(NROWPD,NEQ)
C
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 DVODE 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 DVODE. On the first call to DVODE, 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 input used.
C RWORK   = Real work array of length at least..
C           20 + 16*NEQ           for MF = 10,
C           22 + 9*NEQ + 2*NEQ**2 for MF = 21 or 22,
C           22 + 11*NEQ + (3*ML + 2*MU)*NEQ for MF = 24 or 25.
C LRW     = Declared length of RWORK (in user's DIMENSION statement).
C IWORK   = Integer work array of length at least..
C           30           for MF = 10,
C           30 + 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 RPAR,IPAR = user-defined real and integer arrays passed to F and JAC.
C Note that the main program must declare arrays Y, RWORK, IWORK,
C and possibly ATOL, RPAR, and IPAR.
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 DVODE 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 input.)
C     -5 means repeated convergence failures. (Perhaps bad
C         Jacobian 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 DVODE again.  No other parameters need be reset.
C
```

REFERENCES

1. Kee, R. J., Dixon-Lewis, G., Warnatz, J., Coltrin, M. E. and Miller, J. A.: "A Fortran Computer Code Package for the Evaluation of Gas-Phase Multicomponent Transport Properties," Sandia National Laboratories Report SAND86-8246 (1986).
2. Coltrin, M. E., Kee, R. J. and Rupley, F. M.: *International Journal of Chemical Kinetics*, **23** 1111 (1991).
3. Coltrin, M. E., Kee, R. J., Rupley, F. M. and Meeks, E.: "SURFACE CHEMKIN-III: A Fortran Package for Analyzing Heterogeneous Chemical Kinetics at a Solid-Surface - Gas-Phase Interface," Sandia National Laboratories Report SAND96-8217 (1996).
4. Kee, R. J. and Miller, J. A.: "A Structured Approach to the Computational Modeling of Chemical Kinetics and Molecular Transport in Flowing Systems," Sandia National Laboratories Report SAND86-8841 (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.: "CHEMKIN-II: A Fortran Chemical Kinetics Package for the Analysis of Gas-Phase Chemical Kinetics," Sandia National Laboratories Report SAND89-8009 (1990).
7. Kee, R. J., Rupley, F. M. and Miller, J. A.: "The CHEMKIN Thermodynamic Data Base," Sandia National Laboratories Report SAND87-8215B (1990).
8. 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 Report SP-273 (1971).
9. Grcar, J. F.: "The Change Tool for Change Programs and Scripts," Sandia National Laboratories Report SAND92-8225 (1992).
10. Mitchner, M. and Charles H. Kruger, J.: Partially Ionized Gases, John Wiley & Sons, New York (1973).
11. Lindemann, F.: *Trans. Faraday Soc.*, **17** 598 (1922).
12. Gilbert, R. G., Luther, K. and Troe, J.: *Ber. Bunsenges. Phys. Chem.*, **87** 169 (1983).
13. Stewart, P. H., Larson, C. W. and Golden, D.: *Combustion and Flame*, **75** 25 (1989).
14. Wagner, A. F. and Wardlaw, D. M.: *Journal of Physical Chemistry*, **92** 2462 (1988).
15. Stewart, P. H., Larson, C. W. and Golden, D. M.: *Combustion and Flame*, **75** 25 (1989).
16. Janev, R. K., Langer, W. D., K. Evans, J. and D. E. Post, J.: Elementary Processes in Hydrogen-Helium Plasmas, Springer-Verlag, New York (1987).
17. Brown, P. N., Byrne, G. D. and Hindmarsh, A. C.: *SIAM J. Sci. Stat. Comput.*, **10** 1038 (1989).

APPENDIX A. STORAGE ALLOCATION FOR THE WORK ARRAYS

Work arrays ICKWRK, RCKWRK, and CCKWRK contain information about the elements, species and reactions in the mechanism; they also contain some work space needed for internal manipulations. A user wishing to modify a subroutine or to write new routines will probably want to use the work arrays directly. The pointers described below are starting addresses for information stored in the work arrays, and are found in the labeled common block COMMON /CKSTRT/, declared by the use of the include file ckstrt.h.

```
COMMON /CKSTRT/
Integer constants
1  NMM,  NKK,  NII,  MXSP, MXTB, MXTP,  NCP,  NCP1, NCP2, NCP2T,
2  NPAR, NLAR, NFAR, NLAN, NFAL, NREV, NTHB, NRLT, NWL,  NEIM,
3  NJAN, NJAR, NFT1, NF1R, NEXC, NMOM, NXSM, NTDE, NRNU, NORD,
4  MXORD,KEL,  NKKI,
Integer pointers to character string arrays in CCKWRK
5  IcMM, IcKK,
Integer pointers to integer arrays in ICKWRK
6  IcNC, IcPH, IcCH, IcNT, IcNU, IcNK, IcNS, IcNR, IcLT, IcRL,
7  IcRV, IcWL, IcFL, IcFO, IcFT, IckF, IcTB, IckN, IckT, IceI,
8  IcET, IcJN, IcF1, IcEX, IcMO, IcMK, IcXS, IcXI, IcXK, IcTD,
9  IctK, IcRNU,IcORD,IcKOR,IcKI, IckTF,Ick1, Ick2,
Integer pointers to real variables and arrays in RCKWRK
*  NcAW, NcWT, NcTT, NcAA, NcCO, NcRV, NcLT, NcRL, NcFL, NcKT,
1  NcWL, NcJN, NcF1, NcEX, NcRU, NcRC, NcPA, NcKF, NcKR, NcRNU,
2  NcKOR,NcK1, NcK2, NcK3, NcK4, NcI1, NcI2, NcI3, NcI4
INTEGER CONSTANTS:
NMM,    Total count, elements in problem.
NKK,    Total count, species in problem.
NII,    Total count, reactions in problem.
MXSP,   Maximum number of species (reactants plus products) allowed
        in any reaction; unless changed in the interpreter, MXSP=12.
MXTB,   Maximum number of enhanced third-bodies allowed in 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 species;
        unless changed, NCP2T = (MXTP-1)*NCP2 = 14.
NPAR,   Number of parameters required in the rate expression for
        reactions; in the current formulation NPAR=3, however,
        a 4th parameter is used for purposes of scaling.
NLAR,   Number of parameters required for Landau-Teller reactions;
        NLAR=4.
NFAR,   Number of parameters allowed for pressure-dependent
        reactions; NFAR=8.
NLAN,   Total count, Landau-Teller reactions.
NFAL,   Total count, pressure-dependent reactions.
NREV,   Total count, reactions with reverse parameters.
NTHB,   Total count, reactions with third-bodies.
NRLT,   Total count, Landau-Teller reactions with reverse parameters.
NWL,    Total count, reactions with radiation wavelength enhancement.
NEIM,   Total count, electron-impact reactions.
NJAN,   Total count, Janev-Langer,Evans,Post reactions.
```

NJAR, Number of parameters required for an NJAN reaction.
 NFT1, Total count, reactions using fit#1.
 NF1R, Number of parameters required for an NFT1 reaction.
 NEXC, Total count, excitation-only reactions.
 NMOM, Total count, electron momentum-transfer reactions.
 NXSM, Total count, ion momentum-transfer reactions.
 NTDE, Total count, non-thermal-equilibrium reactions.
 NRNU, Total count, real stoichiometry reactions.
 NORD, Total count, changed-order reactions.
 MXORD, Maximum number of order changes allowed for above.
 KEL, Species index of the electron species if present.
 NKKI, Total count, ion species in the mechanism.

STARTING ADDRESSES FOR THE CHARACTER WORK SPACE, CCKWRK.

IcMM, CCKWRK(I = IcMM) starts an array of element names;
 CCKWRK(I + M - 1) is the name of element M.
 IcKK, CCKWRK(I = IcKK) starts an array of species names;
 CCKWRK(I + K - 1) is the name of species K.

STARTING ADDRESSES FOR THE INTEGER WORK SPACE, ICKWRK.

IcNC, ICKWRK(I = IcNC) starts a matrix of elemental composition
 for the species;
 ICKWRK(I + (K-1)*NMM + M - 1) is the quantity of element M
 in species K.
 IcPH, ICKWRK(I = IcPH) starts an array of physical phases for the
 species;
 ICKWRK(I + K - 1) = -1, species K is solid
 = 0, species K is gaseous
 = +1, species K is liquid
 IcCH, ICKWRK(I = IcCH) starts an array of electronic charges for
 the species;
 ICKWRK(I + K - 1) = -2, species K has two excess electrons.
 IcNT, ICKWRK(I = IcNT) starts an array of the total number of
 temperatures dividing the ranges of thermodynamic fits of
 the species;
 ICKWRK(I + K - 1) is the number of dividing temperatures
 for thermodynamic fits for species K.
 IcNU, ICKWRK(I = IcNU) starts a matrix of stoichiometric coefficients
 for the MXSP species in the reactions;
 ICKWRK(I + (N-1)*MXSP + L - 1) is the coefficient of the Lth
 participant species in the Nth reaction.
 IcNK, ICKWRK(I = IcNK) starts a matrix of indices for the MXSP
 species in the reactions;
 ICKWRK(I + (N-1)*MXSP + L - 1) is the species index for the
 Lth participant species in the Nth reaction.
 IcNS, ICKWRK(I = IcNS) starts an array of the total counts of
 participant species for the reactions,
 and indicates the reversibility of the reactions;
 ICKWRK(I + N - 1) = +L, reaction N is reversible and has
 L participant species (reactants+products)
 = -L, reaction N is irreversible and has
 L participant species (reactants+products)
 IcNR, ICKWRK(I = IcNR) starts an array of the total count of
 reactants only for the reactions;
 ICKWRK(I + N - 1) is the total reactant count for reaction N.
 IcLT, ICKWRK(I = IcLT) starts an array of reaction indices for
 Landau-Teller reactions;
 ICKWRK(I + N - 1) is the reaction index of the Nth LT reaction.
 IcRL, ICKWRK(I = IcRL) starts an array of reaction indices for
 Landau-Teller reactions with explicit reverse parameters;
 ICKWRK(I + N - 1) is the reaction index of the Nth reaction
 with reverse Landau-Teller parameters.

IcRV, ICKWRK(I = IcRV) starts an array of reaction indices for those with explicit reverse Arrhenius coefficients;
 ICKWRK(I + N - 1) is the reaction index of the Nth reaction with reverse coefficients.

IcWL, ICKWRK(I = IcWL) starts an array of reaction indices for those with radiation wavelength enhancement;
 ICKWRK(I + N - 1) is the reaction index of the Nth reaction with wavelength enhancement.

IcFL, ICKWRK(I = IcFL) starts an array of reaction indices for those with pressure-dependent formulations;
 ICKWRK(I + N - 1) is the reaction index of the Nth pressure-dependent reaction.

IcFO, ICKWRK(I = IcFO) starts an array of formulation types for pressure-dependent reactions;
 ICKWRK(I + N - 1) is the type of the Nth pressure-dependent 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

IcFT, ICKWRK(I = IcFT) starts an array of option types for pressure-dependent reactions;
 ICKWRK(I + N - 1) is an option for the Nth pressure-dependent reaction,
 0 for unimolecular fall-off,
 1 for chemically activated.

IcKF, ICKWRK(I = IcKF) starts an array of third-body species flags for pressure-dependent reactions;
 ICKWRK(I + N - 1) is the third-body species flag for the Nth pressure-dependent reaction,
 0, the concentration of the third-body is the sum of the concentrations of all species in the problem
 K, the concentration of the third-body is the concentration of species K.

IcTB, ICKWRK(I = IcTB) starts an array of reaction indices for those with enhanced third-bodies;
 ICKWRK(I + N - 1) is the reaction index of the Nth third-body reaction.

IcKN, ICKWRK(I = IcKN) starts an array of enhanced species counts for 3rd-body reactions;
 ICKWRK(I + N - 1) is the total enhanced species count for the Nth third-body reaction.

IcKT, ICKWRK(I = IcTB) starts a matrix of species indices for enhanced 3rd bodies in third-body reactions;
 ICKWRK(I + (N-1)*MXTB + L - 1) is the species index of the Lth enhanced species in the Nth third-body reaction.

IcEI, ICKWRK(I = IcEI) starts an array of reaction indices for electron-impact reactions;
 ICKWRK(I + N - 1) is the reaction index of the Nth electron-impact reaction.

IcET, ICKWRK(I = IcET) starts an array of temperature-dependence flags for electron-impact reactions;
 ICKWRK(I + N - 1) is a pointer to the temperature in an array which is used to compute the reaction's rate.

IcJN, ICKWRK(I = IcJN) starts an array of reaction indices for Janev-Langer reactions;
 ICKWRK(I + N - 1) is the reaction index of the Nth Janev-Langer reaction.

IcF1, ICKWRK(I = IcF1) starts an array of reaction indices for fit-type reactions;
 ICKWRK(I + N - 1) is the reaction index of the Nth fit-type reaction.

IcEX, ICKWRK(I = IcEX) starts an array of reaction indices for excitation-only reactions;
 ICKWRK(I + N - 1) is the reaction index of the Nth excitation-only reaction.

IcMO, ICKWRK(I = IcMO) starts an array of reaction indices for those with electron momentum-transfer;
 ICKWRK(I + N - 1) is the reaction index of the Nth electron momentum-transfer reaction.

IcMK, ICKWRK(I = IcMK) starts an array of species indices for an electron's collision partner in the electron momentum-transfer reactions;
 ICKWRK(I + N - 1) is the species index of the collision partner in the Nth electron momentum-transfer reaction.

IcXS, ICKWRK(I = IcXS) starts an array of reaction indices for those with ion momentum-transfer cross-section;
 ICKWRK(I + N - 1) is the reaction index of the Nth ion momentum-transfer cross-section reaction.

IcXI, ICKWRK(I = IcXI) starts an array of species indices for the ion collision partner in ion momentum-transfer reactions;
 ICKWRK(I + N - 1) is the species index of the ion collision partner in the Nth ion momentum-transfer reaction.

IcXK, ICKWRK(I = IcXK) starts an array of species indices for the non-ion collision partner in ion momentum-transfer reactions;
 ICKWRK(I + N - 1) is the species index of the non-ion collision partner for the Nth.

IcTD, ICKWRK(I = IcTD) starts an array of reaction indices for those with non-thermal-equilibrium temperature-dependence;
 ICKWRK(I + N - 1) is the reaction index of the Nth non-thermal-equilibrium reaction.

IcTK, ICKWRK(I = IcTK) starts an array of temperature-dependent species indices for the non-thermal-equilibrium reactions;
 ICKWRK(I + N - 1) is the index of the species which determines the temperature for the Nth non-thermal-equilibrium reaction.

IcRNU, ICKWRK(I = IcRNU) starts an array of reaction indices for those with real stoichiometry;
 ICKWRK(I + N - 1) is the reaction index of the Nth reaction with real stoichiometry.

IcORD, ICKWRK(I = IcORD) starts an array of reaction indices for those with changed-order species;
 ICKWRK(I + N - 1) is the reaction index of the Nth reaction with changes of order.

IcKOR, ICKWRK(I = IcKOR) starts a matrix of species indices for the changed-order reactions;
 K = ICKWRK(I + (N-1)*MXORD + L - 1) is the species number of the Lth change-of-order for the Nth changed-order reaction;
 > 0, K participates in the reverse direction (product),
 < 0, K participates in the forward direction (reactant).

IcKI, ICKWRK(I = IcKI) starts an array of species indices for those which are ions;
 ICKWRK(I + N - 1) is the species index for the Nth ion.

IcKTF, ICKWRK(I = IcKTF) starts an array of temperature array pointers for the species;
 ICKWRK(I + K - 1) is the pointer to the temperature array for species K.

IcK1, ICKWRK(I = IcK1) starts scratch storage space of length NKK.
 IcK2, ditto

STARTING ADDRESSES FOR THE REAL WORK SPACE, RCKWRK.

NcAW, RCKWRK(I = NcAW) starts an array of atomic weights (gm/mole);
 RCKWRK(I + M - 1) is the atomic weight of element M.

NcWT, RCKWRK(I = NcWT) starts an array of molecule weights (gm/mole);
 RCKWRK(I + K - 1) is the molecular weight of species K.

NcTT, RCKWRK(I = NcTT) starts an array of temperatures (Kelvin) used to fit thermodynamic properties for the species;
 RCKWRK(I + (K-1)*MXTP + N - 1) is the Nth temperature for species K.

NcAA, RCKWRK(I = NcAA) starts a three-dimensional array of polynomial coefficients for thermodynamic properties of the species; RCKWRK(I + (L-1)*NCP2 + (K-1)*NCP2T + N - 1) is the Nth polynomial coefficient A(N,L,K) for species K, in the Lth temperature range.

NcCO, RCKWRK(I = NcCO) starts a matrix of Arrhenius parameters for reactions; RCKWRK(I + (N-1)*(NPAR+1) + L - 1) is the Lth parameter of reaction N, where
L=1 is the pre-exponential factor (mole-cm-sec-K),
L=2 is the temperature exponent,
L=3 is the activation energy (Kelvins), and
L=4 is used as a scalar for sensitivity analysis.

NcRV, RCKWRK(I = NcRV) starts a matrix of reverse Arrhenius parameters for reactions which give them explicitly; RCKWRK(I + (N-1)*(NPAR+1) + L - 1) is the Lth reverse parameter for the Nth reaction with reverse parameters declared, where
L=1 is the pre-exponential factor (mole-cm-sec-K),
L=2 is the temperature exponent,
L=3 is the activation energy (Kelvins),
L=4 is used as a scalar for sensitivity analysis.
The reaction index is ICKWRK(IcRV + N - 1).

NcLT, RCKWRK(I = NcLT) starts a matrix of parameters for the Landau-Teller reactions; RCKWRK(I + (N-1)*NLAR + L - 1) is the Lth Landau-Teller parameter for the Nth Landau-Teller reaction, where
L=1 is B(I) (Eq. 78) (Kelvins**1/3), and
L=2 is C(I) (Eq. 78) (Kelvins**2/3).
The reaction index is ICKWRK(IcLT + N - 1).

NcRL, RCKWRK(I = NcRL) starts a matrix of explicitly-given reverse Landau-Teller parameters; RCKWRK(I + (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. 78) (Kelvins**1/3), and
L=2 is C(I) (Eq. 78) (Kelvins**2/3).
The reaction index is ICKWRK(IcRL + N - 1).

NcFL, RCKWRK(I = NcFL) starts a matrix of parameters for the pressure-dependent reactions; RCKWRK(I + (N-1)*NFAR + L - 1) is the Lth parameter for the Nth pressure-dependent reaction, where the low pressure limits 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. 73 parameter a,
L=5 is the Eq. 73 parameter T*** (Kelvins),
L=6 is the Eq. 73 parameter T* (Kelvins), and
L=7 is the Eq. 73 parameter T** (Kelvins).
SRI: L=4 is the Eq. 74 parameter a,
L=5 is the Eq. 74 parameter b (Kelvins),
L=6 is the Eq. 74 parameter c (kelvins),
L=7 is the Eq. 74 parameter d, and
L=8 is the Eq. 74 parameter e.
The reaction index is ICKWRK(IcFL+N-1) and the type of formulation is ICKWRK(IcFO+N-1).

NcKT, RCKWRK(I = NcKT) starts a matrix of enhancement factors for third-body reactions; RCKWRK(I + (N-1)*MXTB + L - 1) is an enhancement factor for the Lth enhanced species in the Nth third-body reaction; the reaction index is ICKWRK(IcTB+N-1), and the Lth enhanced species index is ICKWRK(IcKT+(N-1)*MXTB+L-1).

NcWL, RCKWRK(I = NcWL) starts an array of wavelengths for wavelength-enhanced reactions;
RCKWRK(I + N - 1) is the wavelength enhancement (angstrom) for the Nth wavelength-enhanced reaction;
the reaction index is ICKWRK(IcWL+N-1).

NcJN, RCKWRK(I = NcJN) starts a matrix of parameters for the Janev-Langer reactions;
RCKWRK(I + (N-1)*NJAR + L - 1) is Lth parameter for the Nth Janev-Langer reaction, where
L=1 is
L=2 is
The reaction index is ICKWRK(IcJN+N-1).

NcF1, RCKWRK(I = NcF1) starts an array of parameters for fit#1 reactions;
RCKWRK(I + (N-1)*NF1R + L - 1) is the Lth parameter for the Nth fit-type reaction, where
L=1 is
L=2 is
The reaction index is ICKWRK(IcF1+N-1).

NcEX, RCKWRK(I = NcEX) starts an array of energy losses for excitation-only reactions;
RCKWRK(I + N - 1) is the excitation energy loss per event for the Nth excitation only reaction.
The reaction index is ICKWRK(IcEX+N-1).

NcRU, RCKWRK(I = NcRU) is the universal gas constant (ergs/mole-K).
NcRC, RCKWRK(I = NcRC) is the universal gas constant (cal/mole-K).
NcPA, RCKWRK(I = NcPA) is the pressure of one standard atmosphere (dynes/cm**2).

NcKF, RCKWRK(I = NcKF) starts an array of the temperature-dependent forward rate components for reactions.

NcKR, RCKWRK(I = NcKR) starts an array of the temperature-dependent reverse rate components for reactions.

NcRNU, RCKWRK(I = NcRNU) starts a matrix of stoichiometric coefficients for reactions with real stoichiometry;
RCKWRK(I + (N-1)*MXSP + L - 1) is the coefficient for the Lth species in the Nth real stoichiometry reaction.
The reaction index is ICKWRK(IcRNU+N-1).
The species index is ICKWRK(IcNUNK+(N-1)*MXSP+L-1).

NcKOR, RCKWRK(I = NcKOR) starts a matrix of order values for changed-order species reactions;
RCKWRK(I + (N-1)*MXORD + L - 1) is the order for the Lth changed-order species in the Nth change-order reaction.
The reaction index is ICKWRK(IcKOR+N-1).
The change-order species index is
ICKWRK(IcKOR+(N-1)*MXORD+L-1).

NcK1, RCKWRK(I = NcK1) starts species scratch workspace.
NcK2, RCKWRK(I = NcK2) starts species scratch workspace.
NcK3, RCKWRK(I = NcK3) starts species scratch workspace.
NcK4, RCKWRK(I = NcK4) starts species scratch workspace.
NcI1, RCKWRK(I = NcI1) starts reaction scratch workspace.
NcI2, RCKWRK(I = NcI2) starts reaction scratch workspace.
NcI3, RCKWRK(I = NcI3) starts reaction scratch workspace.
NcI4, RCKWRK(I = NcI4) starts reaction scratch workspace.

STORING DATA INTO THE ARRAYS is usually accomplished by a CALL SKINIT, which reads a linkfile generated by the gas-phase mechanism interpreter;
the linkfile consists of the following records:
Linkfile information:
1. FILVER
2. PRVERS
3. PREC
4. KERR
5. LENI, LENR, LENC
Parameters and constants:

6. MAXSP, MAXTB, MAXTP, NTHCF, NIPAR, NITAR,
NIFAR, NJA, MXORD, NF1

7. MM, KK, II, NREV, NFAL, NTHB, NLAN, NRLT, NWL,
NCHRG, NEIM, NJAN, NFT1, NEXC, NMOM, NXSM,
NTDE, NRNU, NORD, KEL, KKION

8. CKMIN
Element data:

9. (CCKWRK(IcMM + M - 1), M = 1, MM) element names
10. (RCKWRK(NcAW + M - 1), M = 1, MM) atomic weights

Species data:

11. (CCKWRK(IcKK + K - 1), K = 1, KK) species names
12. (RCKWRK(NcWT + K - 1), K = 1, KK) molecular weights
13. ((ICKWRK(IcNC + (K-1)*MM + M - 1),
M = 1, MM), K = 1, KK) composition
14. (ICKWRK(IcCH + K - 1), K = 1, KK) electronic charge
15. (ICKWRK(IcNT + K - 1), K = 1, KK) #fit temperatures
16. (ICKWRK(IcPH + K - 1), K = 1, KK) physical phase
17. ((RCKWRK(NcTT + (K-1)*MAXTP + L - 1),
L = 1, MAXTP), K = 1, KK) fit temperatures
18. (((RCKWRK(NcAA + (L-1)*NCP2 + (K-1)*NCP2T + N - 1),
N = 1, NCP2), L = 1, (MAXTP-1)), K = 1, KK) thermodynamics

Ion data (if NKKI > 0):

19. NKKI
20. (ICKWRK(IcKI + K - 1), K = 1, NKKI) ion species indices

Reaction data (if II > 0):

21. (ICKWRK(IcNS + I - 1), I = 1, II) species counts
22. (ICKWRK(IcNR + I - 1), I = 1, II) reactant counts
23. ((ICKWRK(IcNU + (I-1)*MAXSP + N - 1),
ICKWRK(IcNK + (I-1)*MAXSP + N - 1),
N = 1, MAXSP), I = 1, II) stoichiometric coeff'nts
species indices
24. ((RCKWRK(NcCO + (I-1)*(NPAR+1) + N - 1),
N = 1, NPAR), I = 1, II) Arrhenius coefficients

Explicit reverse parameter reaction data (if NREV > 0):

25. NREV
26. (ICKWRK(IcRV + N - 1), N = 1, NREV) reaction indices
27. ((RCKWRK(NcRV + (N-1)*(NPAR+1) + L - 1),
L = 1, NPAR), N = 1, NREV) reverse coefficients

Pressure-dependent reaction data (if NFAL > 0):

28. NFAL, NFAR
29. (ICKWRK(IcFL+N-1),
ICKWRK(IcFO+N-1),
ICKWRK(IcFT+N-1),
ICKWRK(IcKF+N-1), N = 1, NFAL) reaction indices
option type
flow type
3rd-body species index
30. ((RCKWRK(NcFL + (N-1)*NFAR + L - 1),
L = 1, NFAR), N = 1, NFAL) option parameters

Third-body reaction data (if NTHB > 0):

31. NTHB
32. (ICKWRK(IcTB + N - 1),
ICKWRK(IcKN + N - 1), N = 1, NTHB) reaction indices
3rd body count
33. ((ICKWRK(IcKT + (N-1)*MAXTB + L - 1),
L = 1, MAXTB), N = 1, NTHB) 3rd body species indices
34. ((RCKWRK(NcKT + (N-1)*MAXTB + L - 1),
L = 1, MAXTB), N = 1, NTHB) enhancement factors

Landau-Teller reaction data (if NLAN > 0):

35. NLAN, NLAR
36. (ICKWRK(IcLT + N - 1), N = 1, NLAN) reaction indices
37. ((RCKWRK(NcLT + (N-1)*NLAR + L - 1),
L = 1, NLAR), N = 1, NLAN) L-T parameters

Landau-Teller reverse reaction data (if NRLT > 0):

38. NRLT
39. (ICKWRK(IcRL + N - 1), N = 1, NRL) reaction indices
40. ((RCKWRK(NcRL + (N-1)*NLAR + L - 1),
L = 1, NLAR), N = 1, NRLT) reverse L-T parameters

Wavelength enhancement reaction data (if NWL > 0):

41. NWL
42. (ICKWRK(IcWL + N - 1), N = 1, NWL) reaction indices
43. (RCKWRK(NcWL + N - 1), N = 1, NWL) enhancement factors
Electron-impact reaction data (if NEIM > 0):
44. NEIM
45. (ICKWRK(IcEI + N - 1), N = 1, NEIM) reaction indices
46. (ICKWRK(IcET + N - 1), N = 1, NEIM) electron energy
Janev-Langer reaction data (if NJAN > 0):
47. NJAN, NJAR
48. (ICKWRK(IcJN + N - 1), N = 1, NJAN) reaction indices
49. ((RCKWRK(NcJN + (N-1)*NJAR + L - 1), J-L parameters
L = 1, NJAR), N = 1, NJAN)
Fit #1 reaction data (if NFT1 > 0):
50. NFT1, NF1R
51. (ICKWRK(IcF1 + N - 1), N = 1, NFT1) reaction indices
52. ((RCKWRK(NcF1 + (N-1)*NF1R + L - 1), fit#1 parameters
N = 1, NF1R), N = 1, NFT1)
Excitation-only reaction data (if NEXC > 0):
53. NEXC
54. (ICKWRK(IcEX + N - 1), N = 1, NEXC) reaction indices
55. (RCKWRK(NcEX + N - 1), N = 1, NEXC)
Electron momentum-transfer collision reaction data (if NMOM > 0):
56. NMOM
57. (ICKWRK(IcMO + N - 1), N = 1, NMOM) reaction indices
58. (ICKWRK(IcMK + N - 1), N = 1, NMOM) partner species indices
Ion momentum-transfer cross-section reaction data (if NXSM > 0):
59. NXSM
60. (ICKWRK(IcXS + N - 1), N = 1, NXSM) reaction indices
61. (ICKWRK(IcXI + N - 1), N = 1, NXSM) ion species indices
62. (ICKWRK(IcXK + N - 1), N = 1, NXSM) partner species indices
Non-thermal-equilibrium reaction data (if NTDE > 0):
63. NTDE
64. (ICKWRK(IcTD + N - 1), N = 1, NTDE) reaction indices
65. (ICKWRK(IcTK + N - 1), N = 1, NTDE) species indices
Real stoichiometry reaction data (if NRNU > 0):
66. NRNU
67. (ICKWRK(IcRNU + N - 1), N = 1, NRNU) reaction indices
68. ((RCKWRK(NCRNU + (N-1)*MAXSP + L - 1), stoichiometric coeff'nts
L = 1, MAXSP), N = 1, NRNU)
Changed-order reaction data (if NORD > 0):
69. NORD
70. (ICKWRK(IcORD + N - 1), N = 1, NORD) reaction indices
71. ((ICKWRK(IcKOR + (N-1)*MXORD + L - 1), change-order species
L = 1, MXORD), N = 1, NORD)
72. ((RCKWRK(NcKOR + (N-1)*MXORD + L - 1), change-order values
L = 1, MXORD), N = 1, NORD)

INITIAL DISTRIBUTION
UNLIMITED RELEASE

0367 R. J. Buss, 1812
0601 J. Tsao, 1126
0601 M. E. Bartram, 1126
0601 W. G. Breiland, 1126
0601 M. E. Coltrin, 1126
0601 J. R. Creighton, 1126
0601 P. Ho, 1126
0601 H. K. Moffat, 1126
0601 G. H. Hays, 1128
Attn: G. A. Hebner, 1128
P. A. Miller, 1128
M. E. Riley, 1128
0826 W. Hermina, 9111
0826 D. K. Gartling, 9111
0826 S. N. Kempka, 9111
0826 P. R. Schunk, 9111
0827 R. T. McGrath, 9114
0827 T. J. Bartel, 9114
0827 J. E. Brockmann, 9114
0827 R. B. Campbell, 9114
0827 S. J. Choi, 9114
0827 A. S. Geller, 9114
0827 M. L. Hudson, 9114
0827 J. Johannes, 9114
0827 D. J. Rader, 9114
0827 A. J. Russo, 9114
0827 R. Veerasingam, 9114
0827 C. C. Wong, 9114
0834 A. C. Ratzel, 9112
0834 M. R. Baer, 9112
0834 J. R. Torczynski, 9112
0841 P. J. Hommert, 9100
Attn: R. D. Skocypec, 9102
J. H. Biffle, 9103
Elaine D. Gorham, 9104
S. E. Gianoulakis, 9113
W. H. Rutledge, 9115
C. W. Peterson, 9116
1078 C. W. Gwyn, 1302
1078 J. D. McBrayer, 1302
1079 A. D. Romig, 1300
Attn: R. S. Blewer, 1305
G. V. Herrera, 1308
P. Esherick, 1311
L. M. Cecchi, 1326
1111 J. N. Shadid, 9221

1427 P. L. Mattern, 1100
 Attn: S. T. Picraux, 1112
 J. Nelson, 1113
 T. A. Michalske, 1114
 G. A. Samara, 1152
 E. B. Stechel, 1153

9001 T. Hunter, 8000
 Attn: J. B. Wright, 2200
 A. West, 8200
 R. C. Wayne, 8400
 P. N. Smith, 8500
 L. A. Hiles, 8800

9042 C. M. Hartwig, 8345
 9042 G. H. Evans, 8345
 9042 J. F. Grcar, 8345
 9042 S. K. Griffiths, 8345
 9042 W. G. Houf, 8345
 9042 R. J. Kee, 8303 (10)
 9042 R. S. Larson, 8345
 9042 A. E. Lutz, 8345
 9042 E. Meeks, 8345 (10)
 9042 R. H. Nilson, 8345
 9042 F. M. Rupley, 8345 (50)
 9042 J. W. Shon, 8345
 9042 P. A. Spence, 8345
 9042 A. Ting, 8345
 9052 D. R. Hardesty, 8361
 9052 M. D. Allendorf, 8361
 9052 L. L. Baxter, 8361
 9053 S. R. Vosen, 8362
 9053 D. W. Hahn, 8366
 9054 W. J. McLean
 Attn: C. W. Robinson, 8301
 W. Bauer, 8302
 R. W. Carling, 8362
 R. J. Gallagher, 8366

9055 F. P. Tully, 8353
 9055 J. L. Durant, 8353
 9055 J. A. Miller, 8353
 9057 L. A. Rahn, 8351
 9057 J. H. Chen, 8351
 9057 T. Echecki, 8351
 9057 H. N. Najm, 8351
 9057 P. H. Paul, 8351
 9141 S. Subramanian, 8800
 9161 W. G. Wolfer, 8717
 9162 D. A. Buchenauer, 8716

9405 T. M. Dyer, 8700
Attn: M. W. Perra, 8711
M. I. Baskes, 8712
J. C. F. Wang, 8713
G. J. Thomas, 8715
K. L. Wilson, 8716
G. A. Benedetti, 8741
M. R. Birnbaum, 8742
P. E. Nielan, 8743
W. A. Kawahara, 8746
D. B. Nelson, 8783

9214 C. F. Melius, 8117
9409 R. H. Stulen, 8250
9021 Technical Communications Department, 8535, for OSTI (10)
9021 Technical Communications Department, 8535/Technical Library, MS8099, 13414
8099 Technical Library 13414 (4)
9018 Central Technical Files, 8523-2 (3)