Fall 1995

Reaction kinetics of methanol and MTBE oxidation and pyrolysis

Wen-Chiun Ing

New Jersey Institute of Technology

Follow this and additional works at: https://digitalcommons.njit.edu/dissertations

Part of the Chemical Engineering Commons

Recommended Citation
https://digitalcommons.njit.edu/dissertations/1029

This Dissertation is brought to you for free and open access by the Theses and Dissertations at Digital Commons @ NJIT. It has been accepted for inclusion in Dissertations by an authorized administrator of Digital Commons @ NJIT. For more information, please contact digitalcommons@njit.edu.
Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be “used for any purpose other than private study, scholarship, or research.” If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that user may be liable for copyright infringement.

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation.

Printing note: If you do not wish to print this page, then select “Pages from: first page # to: last page #” on the print dialog screen.
The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6” x 9” black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700  800/521-0600
PLEASE NOTE

Page(s) not included with original material and unavailable from author or university. Filmed as received.
ABSTRACT

REACTION KINETICS OF METHANOL AND MTBE OXIDATION AND PYROLYSIS

by
Wen-chiu Ing

This study presents experimental data on the decomposition of methanol in several different reaction environments - fuel lean to stoichiometric at a temperature range of 873 and 1073 K and a pressure range of 1 and 5 atm. Methane fuel is also added in several of the systems studied in order to provide experimental data to understand the methanol addition effect on the methane oxidation.

Computer codes: ThermCal, ThermSrt and ThermCvt have been developed for the thermal property calculations of stable molecules by the Benson group additivity method and of radicals by the NJIT hydrogen bond increment method.

Pressure dependent rate coefficients have been expressed using Chebyshev polynomials adopted for complex chemical activated reaction systems in this study, as well as unimolecular decomposition reactions. This method has also been tested and shows significant improvement over two convention methods, Troe's and SRI. The Levenberg-Marquardt algorithm has been incorporated with the QRRK code, CHEMACT, for the fitting of Chebyshev polynomials.
A pressure dependent mechanism which consists 147 species and 448 elementary reactions, based on thermochemical kinetic principals has been developed and calibrated by the experimental data. The reaction mechanisms (models) include pathways for formation of higher molecular weight products, such as the formation of methyl ethers. This accurate model based on principles of thermochemical kinetics and statistical mechanics will not only provide fundamental understanding, but can be used to suggest directions toward process optimization for experimental testing.

The CHEMKIN interpreter has been modified in this study to take the flexible matrix size of Chebyshev polynomials and to generate the appropriate link file for further processing. Subroutines involving kinetic rate coefficient calculation and array size initialization are also modified to incorporate the Chebyshev polynomials expression. Existing CHEMKIN drivers can simply link with the modified CHEMKIN library to take temperature and pressure dependent Chebyshev polynomials without modification. A couple of drivers have been linked with the new CHEMKIN library and tested. These include a shock tube driver, a constant temperature and pressure driver, and an adiabatic constant pressure driver.

The mechanism is validated with methanol oxidation and pyrolysis experimental data and serves as a basis to build upon during the subsequent efforts on higher molecular weight oxygenated hydrocarbon (MTBE in this study). The methanol addition shows dramatic acceleration effect on the methane oxidation experimentally and predicted by the model.
REACTION KINETICS OF METHANOL AND MTBE OXIDATION AND PYROLYSIS

by
Wen-chiun Ing

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Department of Chemical Engineering,
Chemistry, and Environmental Science

January 1996
APPROVAL PAGE

REACTION KINETICS OF METHANOL AND MTBE OXIDATION AND PYROLYSIS

Wen-chiun Ing

Dr. Joseph W. Bozzelli, Dissertation Advisor  
Distinguished Professor of Chemistry, NJIT

Dr. Robert B. Barat, Committee Member  
Associate Professor of Chemical Engineering, NJIT

Dr. Dana E. Knox, Committee Member  
Associate Professor of Chemical Engineering, NJIT

Dr. Lev N. Krasnoperov, Committee Member  
Associate Professor of Chemistry, NJIT

Dr. Michael Booty, Committee Member  
Associate Professor of Mathematics, NJIT
BIOGRAPHICAL SKETCH

Author: Wen-chiun Ing

Degree: Doctor of Philosophy in Chemical Engineering

Date: January 1996

Undergraduate and Graduate Education:

- Doctor of Philosophy in Chemical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1996

- Master of Science in Environmental Science,
  New Jersey Institute of Technology, Newark, NJ, 1990

- Diploma in Chemical Engineering,
  National Taipei Institute of Technology, Taipei, Taiwan, R.O.C., 1983

Major: Chemical Engineering

Presentations and Publications:


Ing, Wen-chiun; Chen, Chiung-Ju; Bozzelli, Joseph W. "Pyrolysis and Oxidation of MTBE (Methyl Tert-Butyl Ether) and Methanol: Experimental and Detailed Reaction Model", Combustion Reaction Engineering section, AICHE annual meeting, San Francisco, California, USA, Nov 13-18, 1994.


Krasnoperov, Lev N.; Ing, Wen-chiun; Dean, Anthony M.; Bozzelli, Joseph W. "Study on the CH$_2$OH + O$_2$ and CH$_2$O + HO$_2$ Reaction system: Potential Energy


This dissertation is dedicated to
my wife, 蘇秋慧
ACKNOWLEDGMENT

I wish to express my appreciation to Prof. Joseph W. Bozzelli, my advisor, not only for his professional advice but also his encouragement, patience, and kindness. I am deeply indebted to him for the opportunities which he made available to me.

I would also like to thank my dissertation committee members, Dr. Dana E. Knox, Dr. Robert B. Barat, Dr. Lev N. Krasnoperov and Dr. Michael Booty for their helpful corrections and productive comments.

It is my pleasure to thank Dr. Yo Ping Wu, Dr. Wen Pin Ho, who shared their experiences with me and helped me with GC/MS analysis. In addition, I like to thank my coworkers at NJIT, Dr. Hong-Ming Chiang, Xian Zhong and Chiang-Ju Chen, for having dealt with me as a colleague, which has made my time at NJIT much more enjoyable and productive.

For love and inspiration I shall be eternally grateful to my wife, 蘇秋慧 and my parents, 應信岳, 鄭春. Without their constant support and encouragement, I truly believe all of this would not have been possible.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Objectives of This Study</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Literature Review</td>
<td>5</td>
</tr>
<tr>
<td>1.3.1 Methanol</td>
<td>5</td>
</tr>
<tr>
<td>1.3.2 MTBE (Methyl Terbutyl Ether)</td>
<td>10</td>
</tr>
<tr>
<td>2 EXPERIMENTAL</td>
<td>13</td>
</tr>
<tr>
<td>2.1 Experimental Conditions</td>
<td>13</td>
</tr>
<tr>
<td>2.1.1 Methanol Pyrolysis</td>
<td>13</td>
</tr>
<tr>
<td>2.1.2 Methanol Oxidation</td>
<td>14</td>
</tr>
<tr>
<td>2.1.3 Methanol/Methane Oxidation</td>
<td>14</td>
</tr>
<tr>
<td>2.2 Experimental Apparatus</td>
<td>15</td>
</tr>
<tr>
<td>2.3 Temperature Control and Measurement</td>
<td>17</td>
</tr>
<tr>
<td>2.4 Qualitative and Quantitative Analysis</td>
<td>17</td>
</tr>
<tr>
<td>2.5 Summary of Experimental Results</td>
<td>18</td>
</tr>
<tr>
<td>2.5.1 Methanol Pyrolysis</td>
<td>18</td>
</tr>
<tr>
<td>2.5.2 Methanol Oxidation</td>
<td>19</td>
</tr>
<tr>
<td>2.5.3 Methane / Methanol Oxidation</td>
<td>20</td>
</tr>
<tr>
<td>Chapter</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>3.1 Thermodynamic Properties</td>
<td>22</td>
</tr>
<tr>
<td>3.2 Determination of Rate Coefficients</td>
<td>22</td>
</tr>
<tr>
<td>3.2.1 Abstraction Reactions</td>
<td>22</td>
</tr>
<tr>
<td>3.2.2 Addition Reactions</td>
<td>23</td>
</tr>
<tr>
<td>3.2.3 Elimination Reactions (Beta Scission)</td>
<td>23</td>
</tr>
<tr>
<td>3.2.4 Dissociation Reactions (Simple Unimolecular)</td>
<td>24</td>
</tr>
<tr>
<td>3.2.5 Combination Reactions</td>
<td>24</td>
</tr>
<tr>
<td>3.3 Quantum RRK Treatment</td>
<td>25</td>
</tr>
<tr>
<td>3.4 Expression of Rate Coefficients: Chebyshev Polynomials</td>
<td>28</td>
</tr>
<tr>
<td>3.4.1 Chebyshev Approximants</td>
<td>29</td>
</tr>
<tr>
<td>3.4.2 Comparison of Modified Troe and SRI Methods and Chebyshev Polynomials</td>
<td>30</td>
</tr>
<tr>
<td>3.5 Computer Codes Used / Developed for Kinetic Modeling</td>
<td>33</td>
</tr>
<tr>
<td>3.5.1 ThermCal (developed in this study)</td>
<td>33</td>
</tr>
<tr>
<td>3.5.2 ThermCvt / ThermSrt (developed in this study)</td>
<td>34</td>
</tr>
<tr>
<td>3.5.3 RadiCalc (used in this study)</td>
<td>34</td>
</tr>
<tr>
<td>3.5.4 ThermFit (used in this study)</td>
<td>35</td>
</tr>
<tr>
<td>3.5.5 CHEMECT (modified in this study for fitting of Chebyshev Polynomials)</td>
<td>35</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

(Continued)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5.6 CHEMKIN Interpreter and Library (modified in this study)</td>
<td>35</td>
</tr>
<tr>
<td>4 CH$_3$OH + O$_2$ AND CH$_2$O + HO$_2$ REACTION SYSTEMS</td>
<td>37</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>37</td>
</tr>
<tr>
<td>4.2 Calculations and Input Parameters</td>
<td>39</td>
</tr>
<tr>
<td>4.3 Results and Discussion</td>
<td>44</td>
</tr>
<tr>
<td>5 METHANOL PYROLYSIS AND OXIDATION</td>
<td>46</td>
</tr>
<tr>
<td>5.1 Methanol Pyrolysis</td>
<td>46</td>
</tr>
<tr>
<td>5.1.1 Experimental Results</td>
<td>46</td>
</tr>
<tr>
<td>5.1.2 Kinetic Modeling</td>
<td>47</td>
</tr>
<tr>
<td>5.2 Methanol Oxidation</td>
<td>49</td>
</tr>
<tr>
<td>5.2.1 Experimental Results</td>
<td>49</td>
</tr>
<tr>
<td>5.2.2 Kinetic Modeling</td>
<td>50</td>
</tr>
<tr>
<td>5.3 Summary</td>
<td>52</td>
</tr>
<tr>
<td>6 EFFECTS OF METHANOL ADDITION TO METHANE OXIDATION</td>
<td>54</td>
</tr>
<tr>
<td>6.1 Experimental Results</td>
<td>54</td>
</tr>
<tr>
<td>6.2 QRRK Analysis of CH$_3$ + O$_2$ Reactions</td>
<td>55</td>
</tr>
<tr>
<td>6.3 Comparison of Methane Oxidation versus Methanol Oxidation</td>
<td>56</td>
</tr>
<tr>
<td>6.2.1 Initiation</td>
<td>56</td>
</tr>
<tr>
<td>6.2.2 Propagation</td>
<td>56</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

(Continued)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.3 Oxidation</td>
<td>57</td>
</tr>
<tr>
<td>6.4 Acceleration of Methane Oxidation by Added Methanol</td>
<td>57</td>
</tr>
<tr>
<td>7 MTBE (METHYL TERT-BUTYL ETHER) OXIDATION</td>
<td>59</td>
</tr>
<tr>
<td>7.1 Pyrolysis of MTBE</td>
<td>59</td>
</tr>
<tr>
<td>7.1.1 Experimental Results</td>
<td>59</td>
</tr>
<tr>
<td>7.1.2 Unimolecular Dissociation of C₃COC</td>
<td>59</td>
</tr>
<tr>
<td>7.1.3 Comparison of Modeling and Experimental Results</td>
<td>61</td>
</tr>
<tr>
<td>7.2 Oxidation of MTBE</td>
<td>62</td>
</tr>
<tr>
<td>7.2.1 Kinetic Modeling</td>
<td>62</td>
</tr>
<tr>
<td>7.2.2 QRRK Analysis</td>
<td>64</td>
</tr>
<tr>
<td>Unimolecular Dissociation of C₃</td>
<td>64</td>
</tr>
<tr>
<td>Unimolecular Dissociation of C₃-COC</td>
<td>65</td>
</tr>
<tr>
<td>Unimolecular Dissociation of C₂COC</td>
<td>65</td>
</tr>
<tr>
<td>Unimolecular Dissociation of C₂C-O-C</td>
<td>66</td>
</tr>
<tr>
<td>Unimolecular Dissociation of C₂C-O</td>
<td>66</td>
</tr>
<tr>
<td>Unimolecular Dissociation of C₂-C-OC</td>
<td>66</td>
</tr>
<tr>
<td>7.2.3 Mechanism Validation</td>
<td>67</td>
</tr>
<tr>
<td>8 CONCLUSIONS</td>
<td>69</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS
(Continued)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPENDIX A TABLES</td>
<td>...................................................... 71</td>
</tr>
<tr>
<td>APPENDIX B FIGURES</td>
<td>.................................................. 111</td>
</tr>
<tr>
<td>APPENDIX C QRRK INPUT PARAMETERS</td>
<td>........................................ 225</td>
</tr>
<tr>
<td>C.1 CH(_3)OH → Products</td>
<td>.................................................. 225</td>
</tr>
<tr>
<td>C.2 CH(_3) + O(_2) → [CH(_3)OO•]* → Products</td>
<td>.................................................. 227</td>
</tr>
<tr>
<td>C.3 CH(_2)OH + O(_2) ↔ [H(_2)(OH)COO•]* → Products</td>
<td>.................................................. 228</td>
</tr>
<tr>
<td>C.4 CH(_2)O + HO(_2) ↔ [CH(_2)O...HOO•]* → Products</td>
<td>.................................................. 231</td>
</tr>
<tr>
<td>C.5 CH(_2)OH + HO(_2) ↔ [CQH(_2)OH•]* → Products</td>
<td>.................................................. 233</td>
</tr>
<tr>
<td>C.6 CH(_2)O → Products</td>
<td>.................................................. 234</td>
</tr>
<tr>
<td>C.7 CH(_3)O + CH(_3) ↔ [CH(_3)OCH(_3)]* → Products</td>
<td>.................................................. 235</td>
</tr>
<tr>
<td>C.8 CH(_2)O + CH(_3) ↔ [CH(_3)OCH(_2)]* → Products</td>
<td>.................................................. 236</td>
</tr>
<tr>
<td>C.9 CH(_2)OH + CH(_3)O ↔ [CH(_3)OCH(_2)OH]* → Products</td>
<td>.................................................. 237</td>
</tr>
<tr>
<td>C.10 CH(_3)O + CH(_2)O ↔ [CH(_3)OCH(_2)O•]* → Products</td>
<td>.................................................. 238</td>
</tr>
<tr>
<td>C.11 COC• + CH(_2)O ↔ [COCOC•]* → Products</td>
<td>.................................................. 239</td>
</tr>
<tr>
<td>C.12 COCO• + CH(_3) → ↔ [COCOC]* → Products</td>
<td>.................................................. 240</td>
</tr>
<tr>
<td>C.13 COC• + CH(_3)O → ↔ [COCOC]* → Products</td>
<td>.................................................. 241</td>
</tr>
</tbody>
</table>

xii
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPENDIX D FORTRAN PROGRAMS</td>
<td>242</td>
</tr>
<tr>
<td>D.1 THERMCAL</td>
<td>242</td>
</tr>
<tr>
<td>D.2 THERMSRT</td>
<td>267</td>
</tr>
<tr>
<td>D.3 THERMCVT</td>
<td>286</td>
</tr>
<tr>
<td>D.4 INTP4</td>
<td>305</td>
</tr>
<tr>
<td>D.5 CKLIB4</td>
<td>355</td>
</tr>
<tr>
<td>D.6 MECHCVT</td>
<td>363</td>
</tr>
<tr>
<td>D.7 CHEBYSHEV POLY FITTINGS</td>
<td>416</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>437</td>
</tr>
</tbody>
</table>
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Comparisons of Average/Maximum Absolute Relative Errors in the Rate Coefficients</td>
<td>33</td>
</tr>
<tr>
<td>4.1 QRRK Input Parameters for ( \text{CH}_2\text{OH} + \text{O}_2 \leftrightarrow [\text{H}_2(\text{OH})\text{COO}^\ddagger] \rightarrow \text{Products} )</td>
<td>40</td>
</tr>
<tr>
<td>4.2 QRRK Input Parameters for ( \text{H}_2\text{C}=\text{O} + \text{HO}_2 \leftrightarrow [\text{H}_2\text{CO}..\text{HOO}^\ddagger] \rightarrow \text{Products} )</td>
<td>42</td>
</tr>
<tr>
<td>5.1 Experimental conditions of methanol pyrolysis</td>
<td>46</td>
</tr>
<tr>
<td>5.2 Summary of important reactions for methanol pyrolysis and oxidation</td>
<td>53</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1 Experimental Apparatus</td>
<td>112</td>
</tr>
<tr>
<td>B.2 Calibration curve of mass flow controller 1 for argon with inlet pressure of 200 psig</td>
<td>113</td>
</tr>
<tr>
<td>B.3 Calibration curve of mass flow controller 2 for Argon with inlet pressure of 200 psig</td>
<td>114</td>
</tr>
<tr>
<td>B.4 Calibration curve of mass flow controller 3 for methane with inlet pressure of 200 psig</td>
<td>115</td>
</tr>
<tr>
<td>B.5 Calibration curve of mass flow controller 4 for oxygen with inlet pressure of 200 psig</td>
<td>116</td>
</tr>
<tr>
<td>B.6 Reactor Temperature Profiles at 1.3 gm Ar / min</td>
<td>117</td>
</tr>
<tr>
<td>B.7 Reactor Temperature Profiles at 1 atm</td>
<td>118</td>
</tr>
<tr>
<td>B.8 Experimental result of methanol pyrolysis: $T = 1073, K; p = 1, atm; X_{\text{CH}_3\text{OH}} = 0.0395$</td>
<td>119</td>
</tr>
<tr>
<td>B.9 Experimental result of methanol pyrolysis: $T = 1073, K; p = 3, atm; X_{\text{CH}_3\text{OH}} = 0.01317$</td>
<td>120</td>
</tr>
<tr>
<td>B.10 Experimental result of methanol pyrolysis: $T = 1073, K; p = 5, atm; X_{\text{CH}_3\text{OH}} = 0.0079$</td>
<td>121</td>
</tr>
<tr>
<td>B.11 Experimental result of methanol pyrolysis: $T = 1073, K; p = 8, atm; X_{\text{CH}_3\text{OH}} = 0.00494$</td>
<td>122</td>
</tr>
<tr>
<td>B.12 Experimental result of methanol pyrolysis: $T = 1073, K; p = 8, atm; X_{\text{CH}_3\text{OH}} = 0.0025$</td>
<td>123</td>
</tr>
<tr>
<td>B.13 Experimental result of methanol pyrolysis: $T = 1073, K; p = 10, atm; X_{\text{CH}_3\text{OH}} = 0.00395$</td>
<td>124</td>
</tr>
<tr>
<td>B.14 Experimental result of methanol oxidation: $\varphi = 1.0; T = 873, K; p = 3, atm; X_{\text{CH}_3\text{OH}} = 0.0078$</td>
<td>125</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>B.15</td>
<td>Experimental result of methanol oxidation: $\phi = 0.75; T = 873 \text{ K}; p = 3 \text{ atm}; X_{O,CH_3OH} = 0.0078$</td>
</tr>
<tr>
<td>B.16</td>
<td>Experimental result of methanol oxidation: $\phi = 1.0; T = 873 \text{ K}; p = 5 \text{ atm}; X_{O,CH_3OH} = 0.0078$</td>
</tr>
<tr>
<td>B.17</td>
<td>Experimental result of methanol oxidation: $\phi = 0.75; T = 873 \text{ K}; p = 5 \text{ atm}; X_{O,CH_3OH} = 0.0078$</td>
</tr>
<tr>
<td>B.18</td>
<td>Experimental result of methanol oxidation: $\phi = 1.0; T = 923 \text{ K}; p = 1 \text{ atm}; X_{O,CH_3OH} = 0.0078$</td>
</tr>
<tr>
<td>B.19</td>
<td>Experimental result of methanol oxidation: $\phi = 0.75; T = 923 \text{ K}; p = 1 \text{ atm}; X_{O,CH_3OH} = 0.0078$</td>
</tr>
<tr>
<td>B.20</td>
<td>Experimental result of methanol oxidation: $\phi = 1.0; T = 923 \text{ K}; p = 3 \text{ atm}; X_{O,CH_3OH} = 0.0078$</td>
</tr>
<tr>
<td>B.21</td>
<td>Experimental result of methanol oxidation: $\phi = 0.75; T = 923 \text{ K}; p = 3 \text{ atm}; X_{O,CH_3OH} = 0.0078$</td>
</tr>
<tr>
<td>B.22</td>
<td>Experimental result of methanol oxidation: $\phi = 1.0; T = 923 \text{ K}; p = 5 \text{ atm}; X_{O,CH_3OH} = 0.0078$</td>
</tr>
<tr>
<td>B.23</td>
<td>Experimental result of methanol oxidation: $\phi = 0.75; T = 923 \text{ K}; p = 5 \text{ atm}; X_{O,CH_3OH} = 0.0078$</td>
</tr>
<tr>
<td>B.24</td>
<td>Experimental result of methanol oxidation: $\phi = 1.0; T = 973 \text{ K}; p = 1 \text{ atm}; X_{O,CH_3OH} = 0.0078$</td>
</tr>
<tr>
<td>B.25</td>
<td>Experimental result of methanol oxidation: $\phi = 0.75; T = 973 \text{ K}; p = 1 \text{ atm}; X_{O,CH_3OH} = 0.0078$</td>
</tr>
<tr>
<td>B.26</td>
<td>Experimental result of methanol oxidation: $\phi = 1.0; T = 973 \text{ K}; p = 3 \text{ atm}; X_{O,CH_3OH} = 0.0078$</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>B.27</td>
<td>138</td>
</tr>
<tr>
<td>B.28</td>
<td>139</td>
</tr>
<tr>
<td>B.29</td>
<td>140</td>
</tr>
<tr>
<td>B.30</td>
<td>141</td>
</tr>
<tr>
<td>B.31</td>
<td>142</td>
</tr>
<tr>
<td>B.32</td>
<td>143</td>
</tr>
<tr>
<td>B.33</td>
<td>144</td>
</tr>
<tr>
<td>B.34</td>
<td>145</td>
</tr>
<tr>
<td>B.35</td>
<td>146</td>
</tr>
<tr>
<td>B.36</td>
<td>147</td>
</tr>
<tr>
<td>B.37</td>
<td>148</td>
</tr>
<tr>
<td>B.38</td>
<td>149</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>B.39</td>
<td>150</td>
</tr>
<tr>
<td>B.40</td>
<td>151</td>
</tr>
<tr>
<td>B.41</td>
<td>152</td>
</tr>
<tr>
<td>B.42</td>
<td>153</td>
</tr>
<tr>
<td>B.43</td>
<td>154</td>
</tr>
<tr>
<td>B.44</td>
<td>155</td>
</tr>
<tr>
<td>B.45</td>
<td>156</td>
</tr>
<tr>
<td>B.46</td>
<td>157</td>
</tr>
<tr>
<td>B.47</td>
<td>158</td>
</tr>
<tr>
<td>B.48</td>
<td>159</td>
</tr>
<tr>
<td>B.49</td>
<td>160</td>
</tr>
<tr>
<td>B.50</td>
<td>161</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES
(Continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.51 Experimental result of methanol oxidation: $\varphi = 0.75$; $t = 0.6$ s; $p = 3$ atm; $X_{0,\text{CH}_3\text{OH}} = 0.0078$</td>
<td>162</td>
</tr>
<tr>
<td>B.52 Experimental result of methanol oxidation: $\varphi = 1.0$; $t = 0.8$ s; $p = 3$ atm; $X_{0,\text{CH}_3\text{OH}} = 0.0078$</td>
<td>163</td>
</tr>
<tr>
<td>B.53 Experimental result of methanol oxidation: $\varphi = 0.75$; $t = 0.8$ s; $p = 3$ atm; $X_{0,\text{CH}_3\text{OH}} = 0.0078$</td>
<td>164</td>
</tr>
<tr>
<td>B.54 Experimental result of methanol oxidation: $\varphi = 1.0$; $t = 0.6$ s; $p = 5$ atm; $X_{0,\text{CH}_3\text{OH}} = 0.0078$</td>
<td>165</td>
</tr>
<tr>
<td>B.55 Experimental result of methanol oxidation: $\varphi = 0.75$; $t = 0.6$ s; $p = 5$ atm; $X_{0,\text{CH}_3\text{OH}} = 0.0078$</td>
<td>166</td>
</tr>
<tr>
<td>B.56 Experimental result of methanol oxidation: $\varphi = 1.0$; $t = 0.8$ s; $p = 5$ atm; $X_{0,\text{CH}_3\text{OH}} = 0.0078$</td>
<td>167</td>
</tr>
<tr>
<td>B.57 Experimental result of methanol oxidation: $\varphi = 0.75$; $t = 0.8$ s; $p = 5$ atm; $X_{0,\text{CH}_3\text{OH}} = 0.0078$</td>
<td>168</td>
</tr>
<tr>
<td>B.58 Experimental result of methanol oxidation: $\varphi = 1.0$; $t = 1.0$ s; $p = 5$ atm; $X_{0,\text{CH}_3\text{OH}} = 0.0078$</td>
<td>169</td>
</tr>
<tr>
<td>B.59 Experimental result of methanol oxidation: $\varphi = 0.75$; $t = 1.0$ s; $p = 5$ atm; $X_{0,\text{CH}_3\text{OH}} = 0.0078$</td>
<td>170</td>
</tr>
<tr>
<td>B.60 Experimental result of methanol oxidation: $\varphi = 1.0$; $t = 1.5$ s; $p = 5$ atm; $X_{0,\text{CH}_3\text{OH}} = 0.0078$</td>
<td>171</td>
</tr>
<tr>
<td>B.61 Experimental result of methanol oxidation: $\varphi = 0.75$; $t = 1.5$ s; $p = 5$ atm; $X_{0,\text{CH}_3\text{OH}} = 0.0078$</td>
<td>172</td>
</tr>
<tr>
<td>B.62 Experimental result of methane / methanol oxidation: $\varphi = 1.0$; $T = 873$ K; $p = 5$ atm; $X_{0,\text{CH}<em>4} = 0.0078$, $X</em>{0,\text{CH}_3\text{OH}} = 0.0078$</td>
<td>173</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>B.63 Experimental result of methane/methanol oxidation: ( \varphi = 1.0; T = 873 ) K; ( p = 5 ) atm; ( X_{o,CH_4} = 0.01014; X_{o,CH_3OH} = 0.00546 )</td>
<td>174</td>
</tr>
<tr>
<td>B.64 Experimental result of methane/methanol oxidation: ( \varphi = 1.0; T = 873 ) K; ( p = 5 ) atm; ( X_{o,CH_4} = 0.0117; X_{o,CH_3OH} = 0.0039 )</td>
<td>175</td>
</tr>
<tr>
<td>B.65 Experimental result of methane/methanol oxidation: ( \varphi = 1.0; T = 873 ) K; ( p = 5 ) atm; ( X_{o,CH_4} = 0.01326; X_{o,CH_3OH} = 0.00234 )</td>
<td>176</td>
</tr>
<tr>
<td>B.66 Experimental result of methane/methanol oxidation: ( \varphi = 1.0; T = 873 ) K; ( p = 5 ) atm; ( X_{o,CH_4} = 0.01404; X_{o,CH_3OH} = 0.00156 )</td>
<td>177</td>
</tr>
<tr>
<td>B.67 Experimental result of methane/methanol oxidation: ( \varphi = 1.0; T = 873 ) K; ( p = 5 ) atm; ( X_{o,CH_4} = 0.0039; X_{o,CH_3OH} = 0.0039 )</td>
<td>178</td>
</tr>
<tr>
<td>B.68 Experimental result of methane/methanol oxidation: ( \varphi = 1.0; T = 873 ) K; ( p = 5 ) atm; ( X_{o,CH_4} = 0.00273; X_{o,CH_3OH} = 0.00507 )</td>
<td>179</td>
</tr>
<tr>
<td>B.69 Experimental result of methane/methanol oxidation: ( \varphi = 1.0; T = 873 ) K; ( p = 5 ) atm; ( X_{o,CH_4} = 0.00195; X_{o,CH_3OH} = 0.00585 )</td>
<td>180</td>
</tr>
<tr>
<td>B.70 Experimental result of methane/methanol oxidation: ( \varphi = 1.0; T = 1023 ) K; ( p = 1 ) atm; ( X_{o,CH_4} = 0.0078; X_{o,CH_3OH} = 0.0078 )</td>
<td>181</td>
</tr>
<tr>
<td>B.71 Experimental result of methane/methanol oxidation: ( \varphi = 1.0; T = 1073 ) K; ( p = 1 ) atm; ( X_{o,CH_4} = 0.0156; X_{o,CH_3OH} = 0.0 )</td>
<td>182</td>
</tr>
<tr>
<td>B.72 Experimental result of methane/methanol oxidation: ( \varphi = 1.0; T = 1073 ) K; ( p = 1 ) atm; ( X_{o,CH_4} = 0.0078; X_{o,CH_3OH} = 0.0078 )</td>
<td>183</td>
</tr>
<tr>
<td>B.73 Experimental result of methane/methanol oxidation: ( \varphi = 1.0; T = 1073 ) K; ( p = 1 ) atm; ( X_{o,CH_4} = 0.0; X_{o,CH_3OH} = 0.0156 )</td>
<td>184</td>
</tr>
<tr>
<td>B.74 Rate coefficients for ( CH_3 \rightarrow CH_3 + CH_3 )</td>
<td>185</td>
</tr>
<tr>
<td>B.75 Rate coefficients for (a) ( CH_2OH + O_2 \rightarrow CH_2O + HO_2 ) (b) ( CH_3OH + O_2 \rightarrow HOCH(OO)H_2 )</td>
<td>186</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>B.76 Relative errors for CH$_2$OH + O$_2$ $\rightarrow$ HOC(OH)H$_2$ (a) Chebyshev 7x3 (b) Chebyshev 9x5</td>
<td>187</td>
</tr>
<tr>
<td>B.77 Relative errors for CH$_2$OH + O$_2$ $\rightarrow$ CH$_2$O + HO$_2$ (a) Chebyshev 7x3 (b) Chebyshev 9x5</td>
<td>188</td>
</tr>
<tr>
<td>B.78 Relative errors for CH$_2$OH + O$_2$ $\rightarrow$ HOC(OH)H$_2$ (a) Chebyshev 7x3 (b) Chebyshev 9x5</td>
<td>189</td>
</tr>
<tr>
<td>B.79 Potential Energy Diagram of CH$_2$OH + O$_2$ and H$_2$C=O + HO$_2$ Reaction Systems</td>
<td>190</td>
</tr>
<tr>
<td>B.80 Results of QRRK Analysis for CH$_2$OH + O$_2$ $\leftrightarrow$ [H$_2$(OH)COO•]* $\rightarrow$ Products at 300K</td>
<td>191</td>
</tr>
<tr>
<td>B.81 Results of QRRK Analysis for CH$_2$OH + O$_2$ $\leftrightarrow$ [H$_2$(OH)COO•]* $\rightarrow$ Products at 900K</td>
<td>192</td>
</tr>
<tr>
<td>B.82 Results of QRRK Analysis for CH$_2$OH + O$_2$ $\leftrightarrow$ [H$_2$(OH)COO•]* $\rightarrow$ Products at 1800K</td>
<td>193</td>
</tr>
<tr>
<td>B.83 Results of QRRK Analysis for CH$_2$OH + O$_2$ $\leftrightarrow$ [H$_2$(OH)COO•]* $\rightarrow$ Products at 1 atm</td>
<td>194</td>
</tr>
<tr>
<td>B.84 Results of QRRK Analysis for CH$_2$OH + O$_2$ $\leftrightarrow$ [H$_2$(OH)COO•]* $\rightarrow$ Products at 10 atm</td>
<td>195</td>
</tr>
<tr>
<td>B.85 Comparison of QRRK Analysis and Experimental Data for CH$_2$OH + O$_2$ $\rightarrow$ H$_2$C=O + HO$_2$</td>
<td>196</td>
</tr>
<tr>
<td>B.86 Results of QRRK Analysis for H$_2$C=O + HO$_2$ $\leftrightarrow$ [H$_2$CO.HOO•]* $\rightarrow$ Products at 300 K</td>
<td>197</td>
</tr>
<tr>
<td>B.87 Results of QRRK Analysis for H$_2$C=O + HO$_2$ $\leftrightarrow$ [H$_2$CO..HOO•]* $\rightarrow$ Products at 900 K</td>
<td>198</td>
</tr>
</tbody>
</table>
LIST OF FIGURES
(Continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.88 Results of QRRK Analysis for $\text{H}_2\text{C}=\text{O} + \text{HO}_2 \leftrightarrow [\text{H}_2\text{CO}..\text{HOO}\bullet]^*$ → Products at 1800 K</td>
<td>199</td>
</tr>
<tr>
<td>B.89 Results of QRRK Analysis for $\text{H}_2\text{C}=\text{O} + \text{HO}_2 \leftrightarrow [\text{H}_2\text{CO}..\text{HOO}\bullet]^*$ → Products at 1 atm</td>
<td>200</td>
</tr>
<tr>
<td>B.90 Comparison of QRRK Analysis and Experimental Data for $\text{H}_2\text{C}=\text{O} + \text{HO}_2 \leftrightarrow \text{H}_2(\text{OH})\text{COO}$•</td>
<td>201</td>
</tr>
<tr>
<td>B.91 Experimental Results of Methanol Pyrolysis at 873 K</td>
<td>202</td>
</tr>
<tr>
<td>B.92 Potential Energy Diagram for $\text{CH}_3\text{OH} \rightarrow \text{Products}$</td>
<td>203</td>
</tr>
<tr>
<td>B.93 Comparison of Model and Experimental Results for Methanol Pyrolysis at 1073 K, 1 atm, $X_0 = 3.95%$</td>
<td>204</td>
</tr>
<tr>
<td>B.94 Comparison of Model and Experimental Results for Methanol Pyrolysis at 1073 K, 3 atm, $X_0 = 1.317%$</td>
<td>205</td>
</tr>
<tr>
<td>B.95 Comparison of Model and Experimental Results for Methanol Oxidation at 873 K, 5 atm, $X_0 = 0.78%, \phi = 1.0$</td>
<td>206</td>
</tr>
<tr>
<td>B.96 Comparison of Model and Experimental Results for Methanol Oxidation at 873 K, 5 atm, $X_0 = 0.78%, \phi = 0.75$</td>
<td>207</td>
</tr>
<tr>
<td>B.97 Comparison of Model and Experimental Results for Methanol Oxidation at Residence Time = 0.8 second, 5 atm, $X_0 = 0.78%, \phi = 1.0$</td>
<td>208</td>
</tr>
<tr>
<td>B.98 Comparison of Model and Experimental Results for Methanol Oxidation at Residence Time = 1.0 second, 5 atm, $X_0 = 0.78%, \phi = 1.0$</td>
<td>209</td>
</tr>
<tr>
<td>B.99 Comparison of Model and Experimental Results for Methanol Oxidation at Residence Time = 1.5 second, 5 atm, $X_0 = 0.78%, \phi = 1.0$</td>
<td>210</td>
</tr>
<tr>
<td>B.100 Comparison of Model and Experimental Results for Methanol Oxidation at 923 K, 3 atm, $X_0 = 0.78%, \phi = 1.0$</td>
<td>211</td>
</tr>
</tbody>
</table>
**LIST OF FIGURES**

(Continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.101 Potential Energy Diagram for $\text{CH}_3 + \text{O}_2 \leftrightarrow [\text{CH}_3\text{OO}]^* \rightarrow \text{Products}$</td>
<td>212</td>
</tr>
<tr>
<td>B.102 Comparisons of QRRK Analysis Results and Experimental Data for $\text{CH}_3 + \text{O}_2 \rightarrow \text{CH}_3\text{OO}$ (a) Bath Gas = Ar (b) Bath Gas = N$_2$</td>
<td>213</td>
</tr>
<tr>
<td>B.103 Comparisons of QRRK Analysis Results and Experimental Data for (a) $\text{CH}_3 + \text{O}_2 \rightarrow \text{CH}_3\text{O} + \text{O}$ (b) $\text{CH}_3 + \text{O}_2 \rightarrow \text{CH}_2\text{O} + \text{OH}$</td>
<td>214</td>
</tr>
<tr>
<td>B.104 Results of QRRK Analysis for $\text{CH}_3 + \text{O}_2 \rightarrow [\text{CH}_3\text{OO}]^* \rightarrow \text{Products}$</td>
<td>215</td>
</tr>
<tr>
<td>B.105 Comparison of Model and Experimental Results for Methane/Methanol Oxidation at 873 K, 1 atm, $X_{\text{CH}<em>4} = 0.78%$, $X</em>{\text{CH}_3\text{OH}} = 0.78%$, $\phi = 1.0$</td>
<td>216</td>
</tr>
<tr>
<td>B.106 Comparison of Model and Experimental Results for Methane/Methanol Oxidation at 873 K, 1 atm, $X_{\text{CH}<em>4} = 0.39%$, $X</em>{\text{CH}_3\text{OH}} = 0.39%$, $\phi = 1.0$</td>
<td>217</td>
</tr>
<tr>
<td>B.107 Comparison of Methane Decay of Model and Experimental Results for Methane/Methanol Oxidation at 873 K, 1 atm, $X_{\text{CH}_3\text{OO}} = 1.56$, $\phi = 1.0$</td>
<td>218</td>
</tr>
<tr>
<td>B.108 Comparison of Methanol Decay of Model and Experimental Results for Methanol and Methane/Methanol Oxidation at 873 K, 5 atm, $\phi = 1.0$</td>
<td>219</td>
</tr>
<tr>
<td>B.109 Comparison of Methanol Decay of Model and Experimental Results for Methanol and Methane/Methanol Oxidation at 873 K, $X_{\text{CH}_3\text{OO}} = 0.78%$, 5 atm, $\phi = 1.0$</td>
<td>220</td>
</tr>
<tr>
<td>B.110 MTBE Pyrolysis, $C_0 = 0.5%$, $-\ln(C/C_0)$ versus Retention Time</td>
<td>221</td>
</tr>
<tr>
<td>B.111 Potential Energy Diagram for MTBE $\rightarrow$ Products</td>
<td>222</td>
</tr>
<tr>
<td>B.112 Comparisons of Experimental and Model Results for MTBE Pyrolysis, $C_0 = 0.5%$</td>
<td>223</td>
</tr>
<tr>
<td>B.113 Model Comparison with Norton's Experimental Data for MTBE Oxidation at 1024 K, 1 atm, $\phi = 0.96$ (Model Result 10 msec offset)</td>
<td>224</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 Overview

Alcohols, such as methanol and ethanol, are under consideration for widespread use as alternative motor fuels (1). Ethers, such as methyl tert-butyl ether (MTBE), are already in limited use as anti-knock components in gasolines (2).

A key factor which will determine the utility of oxygenated hydrocarbons (OHC's) as dedicated fuels, blend components, or additives is their environmental impact, especially on urban air pollutants such as ozone and peroxyacetyl nitrates (PAN's). Methanol is of interest as an additive and an alternative to conventional transportation fuels, because of potential reductions in pollutant emissions. Specifically, methanol is indicated to aid in reduction of unburned hydrocarbons, which enhance ozone production and photo-chemical smog formation, but enhanced emissions of formaldehyde from methanol oxidation might actually contribute to urban smog (3).

The usefulness of OHC's will also depend on their engine performance characteristics. For example, laminar burning velocities of iso-octane / methanol blends in air are lower than the velocity of either component alone (4).

Methanol is easily produced from petroleum and non-petroleum resources such as coal, wood etc. It is also an important early intermediate formed in the combustion of the high-octane fuel additive MTBE (methyl tert-butyl ether) (5).
During the 1990s, methyl tert-butyl ether (MTBE) will be one of the fastest growing chemicals in the world. Between 1980, the first year in which the U. S. International Trade Commission began reporting separate data on MTBE, and 1984, production has grown at an annual compound rate of 20% (6). U. S. consumption of MTBE has been reported to be growing at a rate of 40% per year in 1986 (7). About forty plants are running and thirty more are under construction or planned (8) including three plants which will be built in Canada with a total capacity of 1.5 million metric tons (9-10).

If all projects planned in 1987 are built, MTBE production will reach about 20 billion lb/yr which is about the double of the production in 1987 (11). MTBE demand for methanol will triple to 1.2 billion gal, up from 400 million gal in 1989 and cause the global demand for methanol to grow more than 18% (12).

There are several factors causing the demand for MTBE: the lead phasedown in gasoline in both the U. S. and in Europe, the failure of other nonleaded oxygenates to significantly improve octane ratings and therefore to serve as replacements for tetra ethyl lead in gasoline, and the restrictions of U. S. Environmental Protection Agency on gasoline RVP (Reid Vapor Pressure) in order to decrease tropospheric ozone concentrations (13).

All the refiners in United States and most of those in Europe have accepted MTBE as an octane enhancer. It blends like a hydrocarbon and does not have the handling problems that alcohols do. It can also solve the octane problems of refiners without requiring large additional capital investment. Many processes have been
developed to reduce the cost of MTBE production (14-19), such as: a new process using sulfuric acid catalyst (14), an adsorption process reducing control problems (15), adding a MTBE unit ahead of alkylation in refinery processing (16), a bifunctional catalyst process (17)... etc.(18-19)

1.2 Objectives of this Study

The goals of this study are:

- Development and advancement of the understanding of oxidation processes on oxygenated fuels under atmospheric and high pressure conditions.
- Identification of important reactions and key species in the high temperature chemistry of OHC's.
- Acquisition of experimental data from lab experiments and from the archival literature on oxidation of the oxygenated fuels, i.e. methanol and MTBE mixed with hydrocarbons.
- Development of a pressure and temperature dependent kinetic model, based on thermochemical kinetic principles and calibrated by experimental data, which will allow computer experiments to suggest trends for future experimental testing and preferred fuel compositions that reduce hydrocarbon emissions while hopefully maintaining or improving engine performance.
- Development of computer tools for the above objectives. These incorporate a pressure dependent mechanism into kinetic modeling and are used in conjunction with a modified Sandia National Lab CHEMKIN general kinetic library. The modification is included in this study.
This study presents experimental data on the decomposition of methanol in several different reaction environments - fuel lean to stoichiometric at a temperature range from 873 to 1073 K and a pressure range from 1 to 5 atm. Methane fuel is also added in several of the systems studied in order to provide experimental data on the use of methanol as an additive in motor vehicle fuels. This data will be a basis, along with that in the literature, for validation of a model of methanol oxidation and pyrolysis which can be used for evaluation and simulation of methanol combustion under atmospheric conditions as well as in compression engines or turbines, where pressures are very different from atmospheric.

Methanol was studied as a base case. During oxidation, all higher OHC's can be considered to pass through the same or similar intermediates as methanol. Methanol is also a key intermediate in the oxidation of MTBE. Some experimental data on methanol oxidation already exists, however, modeling based on fundamental principles is lacking. For example, pressure dependencies of reaction rate constants, which are critical to describe oxidation in internal combustion engines correctly, were not considered. The proposed study was therefore begun with limited available experimental data on methanol pyrolysis and oxidation. A fundamental elementary reaction model has therefore been generated to describe data from both the literature and our own experimental data. These studies on methanol will provide experimental and modeling bases upon which to build during subsequent research on OHC's with higher molecular weight than MTBE.
An important issue here is the impact of OHC's on the formation of higher molecular weight hydrocarbons and soot in diesel engines. This is likely to occur in fuel rich, pyrolytic zones. Addition of an appropriate OHC to the fuel has the potential to limit molecular weight growth while not adversely affecting engine performance (20-21). Ethane is chosen as a model compound fuel. Its pyrolytic chemistry is presented as a function of OHC additive.

1.3 Literature Review

1.3.1 Methanol

The oxidation and pyrolysis of gas phase methanol has been studied widely over the past half century by several different methods: diffusion flame, shock tube and static or flow reactors. Study on the pyrolysis and oxidation of methanol by diffusion flames was reported by Smith and Gordon (22) in 1956. They used a quartz probe sampling technique to extract stable species from the flame regions and the samples were analyzed with a consolidated analytical mass spectrometer. The temperature measured in the flames varied from about 200 °C at the wick to about 1400 °C at the tip and edge of the flame. From the analysis of these products, they indicated that the mechanism of burning involves pyrolysis of the alcohol followed by oxidation of the pyrolysis products. The pyrolysis of methanol was free radicals induced by the small percentage of O2 which diffuses in near the base of the burner, and H or OH radicals which come from the reaction of O2 an H2.

Earlier studies were reported by Fletcher (23) in 1934 and by Someno (24) in 1942. Fletcher studied the pyrolysis of methanol in a static system at pressure less than
atmospheric and temperatures between 899 and 1003 K. He proposed that the pyrolysis occurs in two stages; the first stage gives formaldehyde and hydrogen, and in the second stage, formaldehyde is pyrolyzed to CO and H2.

Someno studied the thermal decomposition and slow combustion of monohydric alcohols. He worked in quartz vessels at sub-atmospheric pressure and temperatures up to 923 K. The identification of products was made spectrographically, by IR and visible spectroscopy, and Ketones, aldehydes, ketenes and unsaturated hydrocarbons were found in the products. Formaldehyde was found in the products only under conditions of slow oxidation.

A shock tube study on the ignition of methanol with oxygen was made by Cooke and co-workers (25). The experiment was performed using stoichiometric mixtures of methanol and oxygen diluted with 95% argon at shock generated temperatures from 1573 to 1873 K, pressure range from 200 to 300 Torr. They found that methanol was less reactive than ethanol. In high temperature conditions, the reactions of $\text{CH}_3\text{OH} \rightarrow \text{CH}_3 + \text{OH}$ and $\text{CH}_3\text{OH} \rightarrow \text{CH}_2\text{OH} + \text{H}$ were reported to evolve with fission of the O-H bond less likely to occur.

Bowman studied the high temperature oxidation of methanol behind reflected shock waves in the temperature range from 1545 to 2180 K (26). The reaction of methanol-oxygen mixtures diluted with argon was initiated by reflected shock waves. The concentrations of O, OH, H$_2$O and CO were measured by various spectroscopic techniques. Following shock-heating, oxidation appears to proceed via two distinct phases: an induction period, in which the concentrations of radical species and water
increase rapidly with little change in temperature; followed by exothermic reaction with the concentration of radical species and water slowly approaching equilibrium values. He proposed a small reaction mechanism and indicated that the significant reactions include thermal decomposition of methanol, attack of radicals O, OH and H on methanol and thermal decomposition of the important radical intermediate CH₂OH.

Aronowitz et al. (27) reported methanol oxidation results (CH₃OH/O₂/N₂) in an adiabatic, turbulent flow reactor, having plug flow characteristics which allowed for chemical sampling with high spatial resolution. Experiments were performed at atmospheric pressure, in the temperature and equivalence ratio ranges from 680 to 880 °C and 0.03 to 3.16, respectively. Major products included CO, CO₂ and H₂O, with small amounts of H₂ and CH₂O. Trace quantities of hydrocarbon products were observed at more fuel rich equivalence ratios. They presented a two-step overall or global model to describe the oxidation of methanol to carbon dioxide. They specifically emphasized the importance of HO₂ chemistry in the oxidation of methanol over this temperature region at atmospheric pressure, while Westbrook et al. (28) had previously indicated that hydroperoxyl radicals can be important in high temperature chemistry.

A study of methanol combustion in laminar flames was made by Vandooren and Van Tiggelen (29) in 1981. A detailed mechanism and rate constants of elementary steps for lean methanol flames burning at 40 Torr were investigated by using molecular beam sampling coupled with mass spectrometric analysis. They reported that “about 70% of the fuel molecules were consumed by reaction with hydroxyl radical and 30% by reaction with hydrogen atom”. The main initial product was CH₂OH which then
reacts mostly with molecular oxygen. The occurrence of electronically excited CH$_2$O was detected, and attributed to highly exothermic reactions between CH$_2$OH and O, H or OH radicals. They indicated that the formaldehyde conversion occurred via reactions with radicals, mainly OH, or by unimolecular (low pressure limit) decomposition processes.

Important elementary reactions of CH$_2$OH with molecular and atomic oxygen in the methanol oxidation system were studied by Grotheer et al (30) using a direct discharge flow reactor over temperature range from 25 to 400 °C at pressures around 1 mbar. Radical profiles were monitored by a low-energy electron impact ionization mass spectrometer. The rate coefficient was measured for the CH$_2$OH + O reaction. A strong non-Arrhenius behavior for CH$_2$OH + O$_2$ reaction was reported.

Recent studies on methanol oxidation and pyrolysis have been reported by Norton and Dryer (31-32). In 1989, they performed methanol oxidation experiments using a turbulent flow reactor at equivalence ratios in the range from 0.6 to 1.6 and initial temperatures from 1025 to 1090 K at atmospheric pressure. They indicate that the existing and widely-used mechanism (33) for methanol oxidation is in error and should be updated.

Norton and Dryer present a kinetic mechanism for methanol pyrolysis which matches multiple sets of experimental data from static, flow, and shock tube reactors, covering temperatures of 973 to 1993 K and pressures of 0.3 to 1 atmosphere. They indicated that the fuel decomposition reaction CH$_3$OH → CH$_2$OH + H, previously been included only in mechanisms for high temperature conditions, also has a significant
effect at low temperatures through radical combination reaction. They reported that
the reaction $\text{CH}_3\text{O} + \text{CO} \rightarrow \text{CH}_3 + \text{CO}_2$ rather than $\text{CH}_3\text{OH} + \text{H} \rightarrow \text{CH}_3 + \text{H}_2\text{O}$, was
the major source of $\text{CH}_3$ at low temperatures and the reverse of $\text{CH}_3 + \text{OH} \rightarrow \text{CH}_2\text{OH} + \text{H}$ was important to $\text{CH}_3$ production at high temperatures.

One of the principle problems with the Norton Dryer mechanism is that it does
not consider pressure effects of the important unimolecular and bimolecular reactions.
For example, the pressure dependence of the decomposition steps below are not fully
included or analyzed for different systems:

$\text{CH}_3\text{OH} \rightarrow \text{CH}_3 + \text{OH}$

$\text{CH}_3\text{OH} \rightarrow ^1\text{CH}_2 + \text{H}_2\text{O}$

$\text{CH}_3\text{OH} \rightarrow ^1\text{HCOH} + \text{H}_2$

$\text{CH}_3\text{OH} \rightarrow \text{CH}_2\text{OH} + \text{H}$.

More importantly, the correct path way analysis for :

$\text{CH}_2\text{OH} + \text{O}_2 \rightarrow [\text{OOC}\text{H}_2\text{OH}]# \rightarrow .\text{OCH}_2\text{OH} + \text{O}$

$\rightarrow [\text{HOOC}\text{HOH}]# \rightarrow \text{HCO}_2 + \text{OH}$

$\rightarrow [\text{CH}_2\text{O}\ldots\text{HOO}]# \rightarrow \text{CH}_2\text{O} + \text{HO}_2$

$\rightarrow [\text{HOOC}\text{H}_2\text{O}]# \rightarrow \text{CH}_2\text{O} + \text{HO}_2$

$\rightarrow \text{CH}_2\text{OH} + \text{O}_2$ (Reverse - No Rxn)

$\rightarrow .\text{OOC}\text{H}_2\text{OH}$ (Stabilization)

as well as reactions of $\text{CH}_3\text{O} + \text{O}_2$ to form $[\text{CH}_3\text{OOO}]#$ and its further reactions were
not included.
We note that the Norton-Dryer mechanism does not properly incorporate the negative temperature dependence, pressure effects and correct reaction pathways which account for the observations by Grotheer.

These pressure dependent reactions are very important in the formation of CO and CO₂. In particular, it is important to evaluate properly the CO₂ formation path from HCO₂ formation in the reaction of CH₂OH + O₂ listed above. This reaction proceeds through a chemically activated intermediate, which can be stabilized, dissociate back to reactants (no reaction) or react via isomerization to either HOOC.HOH or HOOCH₂O. isomers; both of which will then immediately dissociate to lower energy products.

Further items missing from present mechanisms include pathways for the formation of higher molecular weight products, such as the formation of methyl ethers, which we observe in this study and report for the first time.

1.3.2 MTBE (Methyl Tert-butyl Ether)

T. J. Wallington (34) conducted a flash photolysis resonance fluorescence study on the kinetics of the hydroxyl radical (OH) reaction with MTBE over the temperature range from 240 to 440 K at total pressures between 25 and 50 Torr. The Arrhenius expression for the overall rate constant was determined as follows:

\[ \text{C₃COC} + \text{OH} \rightarrow \text{Products} \]

\[ k_1 = (5.1 \pm 1.6) \times 10^{12} \exp[-(155 \pm 100)/T] \text{ cm}^3/\text{mole sec} \]

We note that this rate constant includes abstraction of both the methyl ether hydrogens and the primary methyl tert-butyl hydrogens.
A reaction study was made earlier by Cox and Goldstone, but only at room temperature and it was a relative-rate investigation, yielding an overall rate constant at 298 K (35).

There have been three previous experimental kinetic studies of rate constant determinations on the thermal decomposition kinetics of MTBE up to temperatures of 1160 K, and they conclude that the reaction is a four-center molecular elimination,

$$(\text{CH}_3)_2\text{COCH}_3 \rightarrow (\text{CH}_3)_2\text{C}=\text{CH}_2 + \text{CH}_3\text{OH}.$$  

Daly and Wentrup (36) report that the decomposition reaction of MTBE is unimolecular, and they determine a rate constant, of $10^{14.38} \exp(-30970/T) \text{ s}^{-1}$ over the temperature range from 706 to 768 K. Choo et al. (37) report a value of $k = 10^{13.9} \exp(-29700/T) \text{ s}^{-1}$ in a VLPP (Very Low-Pressure Pyrolysis) reactor from 890 to 1160 K, while Brocard and Baronnet (38) obtained $k = 10^{14.0} \exp(-29960/T) \text{ s}^{-1}$ in a Pyrex reactor at 725 to 761 K. The latter workers also find that the addition of propene or toluene does not modify the rate of formation of the major products, methanol, and they conclude that a four-center unimolecular elimination path accounts for the homogeneous thermal decomposition of MTBE over the temperature and pressure ranges of their study.

Other studies on the high temperature oxidation of MTBE in an argon diluent have been reported by Dunphy and Simmie (39) in reflected shock waves over the temperature range from 1024 to 1850 K and at a pressures of 3.5 bar. The mixture compositions varied widely, with equivalence ratios varying from 0.25 for a fuel-lean, through 1.0 for a stoichiometric, to 2.5 for a fuel-rich mix. Measurements of the
ignition delay times, characterized by chemiluminescence and pressure rise, can be concluded that the high temperature oxidation of MTBE is in essence that of methanol + iso-butene.

Norton and Dryer (40) presented the experimental results for flow reactor oxidation of MTBE at equivalence ratio 0.96, initial temperature of 1024 K and atmospheric pressure. Gas samples extracted at fifteen positions along the reactor duct centerline were quenched in the hot-water-cooled probe and stored at 343 K for later gas chromatographic analysis. The purity of the MTBE was 97%.

The main products they observe are isobutene and methanol. At 1024 K, 50% MTBE is converted in a time of about 5 msec, and 90% of conversion is achieved at about 40 msec. Isobutene and methanol are generated at up to 75% and 40% of initial MTBE concentration, and decay slowly compared to the MTBE decomposition rate. An increasing amount of carbon monoxide is produced with MTBE conversion. Methane and propene are observed at about 10% of MTBE initial concentration. Other, lower concentration, products (less than 10% of MTBE initial concentration) such as CO₂, C2, C3, and C5, are also produced.

Norton and Dryer describe the result of MTBE decay by the unimolecular elimination reaction only, which occurs through a four-center activated complex. They attribute the final products observed to faster MTBE oxidation rather than to either isobutene or methanol oxidation.
CHAPTER 2

EXPERIMENTAL

The oxidation and pyrolysis of methanol with and without methane is analyzed by varying temperature, pressure, residence time and reactant ratios. A temperature range from 873 to 1073 K, pressure range from 1 to 5 atm and residence time range from 0.1 to 2.0 seconds is studied in two reaction ratios (stoichiometric and fuel lean) using a high temperature and high pressure tubular reactor for methanol oxidation. Methanol pyrolysis is studied over a pressure range of 1 to 10 atm at 1073 K. Oxidation of Methanol and methane mixtures is studied with an overall stoichiometric reactant ratio at selected temperatures and pressures. Methane oxidation under stoichiometric conditions, with pressure range from 1 to 5 atm and temperatures from 1023 to 1073 K is studied for comparison with methanol and methanol/methane oxidation and for further validation of kinetic models.

2.1 Experimental Conditions

2.1.1 Methanol Pyrolysis

Methanol pyrolysis is studied over a pressure range from 1 to 10 atm at 1073 K under isothermal reactor conditions. The reactant concentrations at different pressures are:
2.1.2 Methanol Oxidation

Methanol oxidation experiments are performed with two reactant ratio sets which represent stoichiometric and fuel lean conditions over a temperature range from 873 to 1073 K and a pressure range from 1 to 5 atm. The reactant concentration ratios are:

- **Fuel Lean** $\phi = 0.75$  
  $\text{CH}_3\text{OH} : \text{Ar} = 0.0078 : 0.0156 : 0.9766$

- **Stoichiometric** $\phi = 1.0$  
  $\text{CH}_3\text{OH} : \text{O}_2 : \text{Ar} = 0.0078 : 0.0117 : 0.9805$

2.1.3 Methanol/Methane Oxidation

Methanol/methane oxidation is carried out at overall stoichiometric conditions at selected temperature and pressure in order to understand the effects of methanol addition on methane oxidation and the effects of methane on methanol oxidation. The total concentration and ratio of methanol and methane at the indicated temperature and pressure are:

<table>
<thead>
<tr>
<th>Pressure (atm)</th>
<th>CH$_3$OH : Ar</th>
<th>CH$_3$OH : O$_2$ : Ar</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 atm</td>
<td>CH$_3$OH : Ar = 0.0395 : 0.9605</td>
<td></td>
</tr>
<tr>
<td>3 atm</td>
<td>CH$_3$OH : Ar = 0.01317 : 0.98683</td>
<td></td>
</tr>
<tr>
<td>5 atm</td>
<td>CH$_3$OH : Ar = 0.0079 : 0.9921</td>
<td></td>
</tr>
<tr>
<td>8 atm</td>
<td>CH$_3$OH : Ar = 0.00494 : 0.99506</td>
<td></td>
</tr>
<tr>
<td>8 atm</td>
<td>CH$_3$OH : Ar = 0.0025 : 0.9975</td>
<td></td>
</tr>
<tr>
<td>10 atm</td>
<td>CH$_3$OH : Ar = 0.00395 : 0.99605</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>Pressure</td>
<td>$X_{o, CH_4+CH_3OH}$</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>----------------------</td>
</tr>
<tr>
<td>873 K</td>
<td>5 atm</td>
<td>0.0156</td>
</tr>
<tr>
<td>873 K</td>
<td>5 atm</td>
<td>0.0156</td>
</tr>
<tr>
<td>873 K</td>
<td>5 atm</td>
<td>0.0156</td>
</tr>
<tr>
<td>873 K</td>
<td>5 atm</td>
<td>0.0156</td>
</tr>
<tr>
<td>873 K</td>
<td>5 atm</td>
<td>0.0156</td>
</tr>
<tr>
<td>873 K</td>
<td>5 atm</td>
<td>0.0156</td>
</tr>
<tr>
<td>873 K</td>
<td>5 atm</td>
<td>0.0078</td>
</tr>
<tr>
<td>873 K</td>
<td>5 atm</td>
<td>0.0078</td>
</tr>
<tr>
<td>873 K</td>
<td>5 atm</td>
<td>0.0078</td>
</tr>
<tr>
<td>873 K</td>
<td>5 atm</td>
<td>0.0078</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Pressure</th>
<th>$X_{o, CH_4+CH_3OH}$</th>
<th>CH$_3$OH : CH$_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1023 K</td>
<td>1 atm</td>
<td>0.0156</td>
<td>1.0 : 1.0</td>
</tr>
<tr>
<td>1073 K</td>
<td>1 atm</td>
<td>0.0156</td>
<td>2.0 : 0.0</td>
</tr>
<tr>
<td>1073 K</td>
<td>1 atm</td>
<td>0.0156</td>
<td>1.0 : 1.0</td>
</tr>
<tr>
<td>1073 K</td>
<td>1 atm</td>
<td>0.0156</td>
<td>0.0 : 2.0</td>
</tr>
</tbody>
</table>

2.2 Experimental Apparatus

A diagram of the experimental apparatus is shown in Figure B.1. Reactants are reagent grade supplied by Aldrich Co. Methane and argon gases are filtered for O$_2$, H$_2$O, and hydrocarbon impurities before entering the reactor system. The carrier gas (argon) is passed through a saturation bubbler which contained liquid methanol held at 273 K.
using an ice bath. A second argon flow stream (after the bubbler) is used as make-up gas in order to achieve the desired reactant ratio. Methane and oxygen are then added to the CH$_3$OH/Ar flow as required. Those four gas streams (argon, make-up argon, oxygen and methane) are controlled by Union Carbide LINDE® Model FM-4550 mass flowmeter-flowcontroller with four mass flow control modules. Four channels are calibrated to indicate 0-100% range of the desired gases. The calibration curves are plotted as Figures B.2 to B.5 of Appendix B.

A quartz tube of 6 mm ID and 12 mm OD is employed as the reactor, which is housed within 75 cm length of three-zone Chemshell 1.25" ID electric tube furnace equipped with three independent Omega Model CN-310 digital temperature controllers. A Neon Controls BPS 26G2501, 200 psi back pressure regulator is used to maintain the desired pressure within the reactor.

The mixed reactants (feed mixture) are preheated to about 373 K to prevent condensation and to improve reactor temperature control. The reactants can either flow through the reactor or flow directly to a GC sampling valve via a bypass line. The bypass is used to determine the initial concentration of reactants without going through the high temperature reactor.

Gas samples are drawn through the sampling line by means of a mechanical vacuum pump with a constant flow rate of 30 cm$^3$/min. A HP-5890 Series II gas chromatograph with two flame ionization detectors is used on-line for analysis. The bulk of the effluent is passed through a sodium bicarbonate (NaHCO$_3$) flask for neutralization before being released to a fume hood.
2.3 Temperature Control and Measurement

Temperature profiles are obtained at each flow using a type K thermocouple probe moved axially within the 75 cm length reactor. Thermocouple error caused by furnace wall radiation is minimized by using a grounded junction sheath and with a representative flow of inert. The darkened outside surface of the quartz tube reactor also served as a second radiation shield. Tight temperature control resulted in temperature profiles isothermal to within ± 5 K over 80 - 85% of the furnace length for each temperature. Steep temperature gradients of 500 K in 5 cm occur at the inlet and outlet of the reactor.

Uncertainty in absolute temperature measurements is estimated to be ± 1% (i.e. ± 8-12 K) but relative temperatures are measured to within ± 5 K. The temperature profiles are shown as Figures B.6 and B.7 with and without reactions present.

2.4 Qualitative and Quantitative Analysis

A HP-5890 Series II gas chromatograph with two flame ionization detectors is used on-line to determine the concentration of reactants and products. A ten-port VALCO gas sampling valve is employed to introduce the gas samples into the GC columns. Gas samples are passed through two sampling loops, 1.0 cm$^3$ and 0.25 cm$^3$, at a constant flow rate of 30 cm$^3$ per minute, and then injected into a packed column and a capillary column, respectively.

Two columns, one packed and one capillary, are used to perform separations. The 6' x 1/8" feet stainless steel column is packed with 50% 80/100 Poropak T and 50% 80/100 Poropak Q for the separation of CO, CO$_2$ and light hydrocarbons. In order
to increase the accuracy of quantitative analysis, a catalytic converter connected in series after the packed column with 5% of 80/100 ruthenium on alumina catalyst is used to convert the CO and CO$_2$ to methane after separation. The 90 m $\times$ 0.53 mm Hewlett Packard fused silica capillary column is used for heavier hydrocarbon and oxy-hydrocarbon separation. The chromatogram peaks are analyzed with two HP 3396A integrators.

Calibration for obtaining appropriate molar response factors and retention times of relevant compounds is performed by injecting known concentrations of standard gases and known quantities of liquid samples. The average retention times and relative response factors are shown in Tables A.1 and A.2.

Product identifications are also verified by HP 5899A GC/Mass Spectrometry, with a HP 90 m $\times$ 0.53 mm fused silica capillary columns (same as the one used in the on-line GC system), on batch samples of reactor gas drawn, from the reactor exit into evacuated 25 cm$^3$ stainless steel sample cylinders for later analysis.

2.5 Summary of Experimental Results

2.5.1 Methanol Pyrolysis

Experimental results on methanol pyrolysis at 1073 K over a pressure range from 1 to 10 atm are displayed in Figures B.8 to B.13. The decay rate of methanol increases with increasing pressure, however, the differences between 5, 8 and 10 atmospheres are small.

Methanol decays and forms the intermediate stable product formaldehyde, which then decomposes to carbon monoxide. Normalized concentrations ($C/C_o$) of
methane formation are about the same at different pressures. Ethylene, which results from methyl radical combination through ethane, is observed at ppm levels with 3.95 mole % initial methanol concentration at 1 atm total pressure.

2.5.2 Methanol Oxidation

Results of methanol oxidation are shown in Figures B.14 to B.61, which plot the mole fraction as a function of reaction time and temperature, at temperatures ranging from of 873 and 1073 K, 3 pressures (1, 3, 5 atm) and two equivalence ratios of 1.0 and 0.75.

Changes in methanol conversion, intermediates and products under conditions of excess oxygen (equivalence ratio = 0.75) are small relative to reaction at stoichiometric conditions for all temperatures and pressures of this study. Methanol starts to decay at 923 K and 100% conversion is observed at 1073 K, at 1 atm and reaction time of 0.15 second. At 3 atm and a reaction time of 0.6 second, methanol decay begins at 873 K and is 95% converted at 923 K. At 5 atm and a reaction time of 0.6 second, methanol conversion is 5% at 873 K and 100% at 923 K.

At increased temperatures, formaldehyde levels decrease and this intermediate only exists for a short residence time (ca. 0.2 seconds). CO increases rapidly with the conversion of methanol. The formation of CO₂ starts to occur when significant amounts of CO produced. At higher temperature, oxidation of CO occurs rapidly to the final product CO₂.

Methane is observed as a minor product. Hydrocarbon molecular weight growth species are not detected (or are below instrument detection limits).
2.5.3 Methane / Methanol Oxidation

Experimental results on the oxidation of methane / methanol mixtures are shown in Figures B.62 to B.73. Methanol is observed to have a significant acceleration effect on methane oxidation. A uniform trend is observed over the temperature and pressure ranges studied by changing the initial composition of methane / methanol mixtures. This acceleration by methanol on methane oxidation will be discussed and compared with modeling results in Chapter 6.
CHAPTER 3

KINETIC MODELING

A mechanism which consists 147 species and 448 elementary reactions, based on thermochemical kinetic principles has been developed and calibrated by the experimental data.

Specific emphasis has been placed on understanding and properly treating the pressure dependent reactions and identifying important reactions and key species in the high temperature chemistry so the model can be applied to both atmospheric and internal engine oxidation / combustion. The reaction mechanisms (models) include pathways for formation of higher molecular weight products, such as the formation of methyl ethers, which are observed in the experiment and reported here for the first time. An accurate model based on principles of thermochemical kinetics and statistical mechanics will not only provide fundamental understanding, but will suggest directions toward process optimization for experimental testing. It will also have a higher probability of successful application outside the range of calibration.

The reaction mechanism is based upon principles of thermochemical kinetics including Transition State Theory (TST) and accurate molecular thermodynamic properties. The mechanism consists of elementary reactions, with each reaction based on literature evaluation or, if it is estimated, on thermochemical and kinetic principles.
3.1 Thermodynamic Properties

The addition reactions and subsequent unimolecular isomerization or dissociation reactions are analyzed by construction of potential energy diagrams of the systems based on existing experimental data, theoretical data and on group additivity estimation techniques.

Thermodynamic parameters - $\Delta H_{298}$, $S_{298}$ and $C_p(300)$ to $C_p(\infty)$ for species in the reaction schemes are listed in Table 1 along with appropriate references. Enthalpies of radicals are from evaluated literature on C-H bond energies and $\Delta H_f$ of the stable molecule which corresponds to the radical with a H atom at the radical site. Entropies and $C_p(T)$ values are from use of Hydrogen Bond Increment (HBI) (41). The HBI group technique is based on known thermodynamic properties of the parent molecule and calculated changes that occur upon formation of a radical via loss of a H atom. The HBI group incorporates evaluated carbon hydrogen (C-H) bond energies, for $\Delta H_{298}$ of the respective radical, and changes that result from loss or changes in vibrational frequencies, internal rotation symmetry, and spin degeneracy when a hydrogen atom is removed from the specific carbon site. HBI groups, are described fully in Ref. (42, 43).

3.2 Determination of Rate Coefficients

3.2.1 Abstraction Reactions

Abstraction reaction rate constants are not pressure dependent and usually taken from evaluated literature when available. If estimation is required, a generic reaction is utilized as a model and adjusted for steric effects. Evans Polanyi analysis is used on the reaction in the exothermic direction to estimate the energy of activation ($E_a$) for rate constant. An Evans Polanyi plot, $E_a$ versus $\Delta H_{\text{exn}}$, allows use of a known $\Delta H_{\text{exn}}$ to obtain
$E_a$ for these reactions. The abstraction reaction in an endothermic reaction must incorporate the $\Delta H_{\text{on}}$ or thermodynamics will be violated.

### 3.2.2 Addition Reactions

Addition reactions are treated with the quantum RRK formalism for $k(E)$ and modified strong collision theory of Gilbert et al. for falloff. This will be described in the following section. The reaction involves addition of an atom or radical to an unsaturated species. Addition reactions typically form an energized adduct with ca. 20 to 50 kcal/mol of energy above the ground state. This energy is often sufficient to allow the adduct to react to other reaction products (lower energy) before stabilization occurs.

### 3.2.3 Elimination Reactions (Beta Scission)

These reactions are the reverse of addition reactions and utilize the quantum RRK formalism for $k(E)$ and modified strong collision theory of Gilbert et al. for falloff and are treated in two ways. In the first way, a unimolecular quantum RRK formalism for $k(E)$ and modified strong collision theory of Gilbert et al. for falloff is employed where reverse high-pressure reaction (addition) parameters are determined. The corresponding high-pressure unimolecular beta scission rate constants using microscopic reversibility are then calculated. These high pressure unimolecular elimination parameters are then input to the QRRK formalism to calculate the rate constants at the appropriate pressure. An alternate method is to use the reverse rate constants from the QRRK - modified strong collision reaction calculations for the corresponding addition reaction.
3.2.4 Dissociation Reactions (Simple Unimolecular)

Simple unimolecular (dissociation) rate constants are determined by two methods similar to beta scission reactions. The reverse high-pressure reaction (combination) parameters are determined. The corresponding high-pressure unimolecular dissociation rate constants are then calculated using microscopic reversibility. These high pressure unimolecular dissociation parameters are then input to the QRRK formalism to calculate the rate constants at the appropriate pressure and temperature. The reverse rate constant from the QRRK combination reaction calculation can also be used for the respective reaction.

3.2.5 Combination Reactions

These reactions involve the combination of two radicals or an atom with a radical. The energy of the adduct formed before stabilization is equal to the bond energy of the new bond(s) formed and this is typically on the order of 80 to 120 kcal/mol. This is often sufficient under combustion conditions for an adduct, with this initial energy above its ground state energy, to react to lower energy products before stabilization occurs. The high-pressure limit rate constant for combination is obtained from literature or estimated from known generic combination reactions. The QRRK chemical activation formalism is then used to calculate the rate constants at the appropriate pressure and temperature to all the recognized available channels.

Reaction to other new product channels as well as isomerization, stabilization and reverse reaction are included in this calculation. This an important aspect of the reaction analysis for both combination as well as addition reactions that other modelers do not usually incorporate. This leads to a more correct treatment of fall-off and
pressure dependence for these non-elementary reaction systems. Rate constants inclusion in for the model are obtained which incorporate the calculated pressure dependency and therefore make the model more fundamentally correct.

3.3 Quantum RRK Treatment

Branching ratios of the adduct formed from combination, addition or insertion reactions to various product channels are calculated using a quantum version of RRK theory (QRRK) to evaluate energy dependent rate constants, \( k(E) \), of the adduct to the various channels. QRRK analysis, as initially presented by Dean (44, 45, 46), combined with the "modified strong collision approach" of Gilbert et al. (47) is used to compute rate constants for both chemical activation and unimolecular reactions over a range of temperature and pressure.

For the chemical activation reactions, stabilization is also included in the calculations.

A significant number of modifications have been made since the initially descriptions of the quantum RRK and fall-off calculations are published (44, 45, 46). These modifications include (46):

- The use of a manifold of 3 vibration frequencies and respective degeneracies, plus incorporation of energies from 1 external rotation mode for the calculation of the ratio of the density of states to the partition coefficient, \( \rho(E)/Q(T) \).
- The 3 vibrational frequencies and degeneracies are also used in calculation of \( k(E) \) and of \( F(E) \).
- The \( FE \) factor is now explicitly calculated for use in determining the collision efficiency \( \beta_c \) (47), in place of the previously assigned 1.15 value.
• $\beta_e$ is now calculated by:

$$\beta_e = \left[ \frac{\alpha_e}{\alpha_e + F_E \cdot k \cdot T} \right]^{\frac{3}{2}} / \Delta$$

from Gilbert et. al Eqn. 4.7 (47),

$$\Delta = \Delta_1 - \frac{(F_E \cdot k \cdot T)}{(\alpha_e + F_E \cdot k \cdot T) \cdot \Delta_2}.$$ 

Where $\Delta_1$ and $\Delta_2$ are temperature-dependent integrals involving the density of states, and $\alpha_e$ is the average energy of down-collisions.

• The Lennard-Jones collision frequency $Z_{lJ}$ is now calculated by $Z_{lJ} = Z \Omega^{(2,2)}$ integral. (48, 49, 50), where $\Omega$ is obtained from fit of Reid et al. (50)

The QRRK analysis with the modified strong collision approach and constant FE for fall-off has been used to analyze a variety of chemical activation reaction systems, Westmoreland et al. (51, 52, 53), Dean et al. (52, 54, 55, 56, 57), Bozzelli et al. (54-57, 58, 59, 60) It is shown to yield reasonable results in these applications, and provides a mechanism by which the effects of temperature and pressure can be both evaluated and included in the kinetics.

Limitations affected by the assumptions in the QRRK and fall off calculations are often overshadowed by uncertainties in high pressure limit rate constants and thermodynamic properties for species and transition state structures in the chemical systems.

Input data requirements for the QRRK calculations:

• Pre-exponential factors (Arrhenius A factors) in the high pressure limits, which are obtained from the literature using the methods of Benson (61) and of Dean (44).
• Activation energies come from endothermicity of reaction $\Delta U_{\text{rxn}}$ and from analogy to similar reactions with known energetics.

• The three vibration frequencies and their associated degeneracies are computed from fits to heat capacity data, as described by Ritter. These have been shown by Ritter to accurately reproduce molecular heat capacities, $C_p(T)$, and by Bozzelli et al. to yield accurate density of states $\rho(E)$ to partition coefficient $(Q)$ ratios. Frequency sets for the adducts are listed in Table 1.

• Lennard-Jones parameters ($\sigma$, $\epsilon/k$) are obtained from tabulations and from a calculation method based on molar volumes and compressibility.

• Arrhenius $A$ factors for bimolecular combination of radicals at the high pressure limit are obtained from literature, and from trends in homologous series of these type reactions.

It is important to have accurate input data for the calculations, without this aspect the accuracy or assumptions in the calculations or chemical activation rate constants are of less value. Several of the input parameters for these calculations, e.g., thermodynamic properties and kinetic parameters, need to be estimated. This is done in a consistent and uniform manner with reference to literature experiment and / or theoretical calculations in all cases. The accumulation and referencing of these data comprise a major component of this work. The input parameters and references are of
utmost importance; they are assembled as indicated in the tables associated with each calculation in Appendix C.

3.4 Expression of Rate Coefficients: Chebyshev Polynomials

Detailed reaction modeling of gas-phase kinetics is a powerful tool in the studies of combustion and atmospheric chemistry. Rate coefficients for dissociation, recombination and chemically-activated reactions can be estimated by statistical theories such as QRRK (44, 51), RRKM (65, 66, 67) and SACM (68, 69, 70, 71). Temperature and pressure-dependent rate coefficients of elementary reactions can be evaluated using one of these theories. However, calculation of rate coefficients is computationally demanding. Empirical or semiempirical approximate formulas, therefore, are generally practiced.

Approximations of fall-off surfaces of simple unimolecular reactions using empirical expressions has been extensively studied such as Troe's $F_{\text{cent}}$ method and method of Stewart and coworkers (SRI method hereafter) (47, 72). Less attention has been taken for complex chemically-activated systems. Recently, Kazakov et al. (73) gave a parametrization formula called generalized-mean-limits formula (GML) based on Gardiner's (74) treatment. They report good agreement for "isothermal" fitting over pressure domains. Venkatesh et al. (75) directly approximate the rate coefficients via Chebyshev polynomials and report good agreement over broad temperature and pressure range for a number of different types of reaction systems. In light of the accuracy using Chebyshev polynomials for complex chemical activated reaction
systems, this method has been tested, employed and compared with two convention methods, Troe's and SRI, in this study.

3.4.1 Chebyshev Approximants

Chebyshev series (76, 77) in the inverse temperature and logarithm of pressure are used as the approximation of the logarithm of the rate coefficients. Temperature and pressure are constrained by predefined minimum and maximum values,

\[ T_{\text{min}} \leq T \leq T_{\text{max}}, \]
\[ P_{\text{min}} \leq P \leq P_{\text{max}}. \]

The Chebyshev polynomial of degree \((i - 1)\) is given by

\[ \varphi_i(x) = \cos ((i-1) \arccos(x)); \quad i = 1, 2, \ldots, \]

where \(x\) is the variable of interest defined to be in the close interval \([-1, +1]\). The transformations to map the temperature and pressure domain onto the unit square are

\[ \frac{2T^{-1} - T_{\text{min}} - T_{\text{max}}}{T_{\text{max}} - T_{\text{min}}} \rightarrow \bar{T}, \]

and

\[ \frac{2\log P - \log P_{\text{min}} - P_{\text{max}}}{\log P_{\text{max}} - \log P_{\text{min}}} \rightarrow \bar{P}. \]

\(k(T, P)\) is then mapped to \(k(\bar{T}, \bar{P})\),

\[ k(T, P) \rightarrow k(\bar{T}, \bar{P}). \]

The logarithm of the rate coefficient is thus approximated as

\[ \log k(\bar{T}, \bar{P}) = \sum_{i=1}^{M} \sum_{j=1}^{M} a_y \varphi_i(\bar{T}) \varphi_j(\bar{P}), \]
where $N$ and $M$ denote temperature and pressure axis respectively and are predefined for different orders of accuracy.

The data points are taken as $d \times d$ Gauss-Chebyshev grid (76, 77) which is given by

$$\tilde{T}_i = \cos \left[ \frac{2i-1}{2d} \pi \right], \quad (-1 \leq \tilde{T}_i \leq 1),$$

$$\tilde{P}_i = \cos \left[ \frac{2i-1}{2d} \pi \right], \quad (-1 \leq \tilde{P}_i \leq 1),$$

where $1 \leq i \leq d$.

A $50 \times 50$ Gauss-Chebyshev grid is taken to fit $N$ by $M$ Chebyshev polynomials using Levenberg-Marquardt algorithm in the next section for the reaction systems of interest.

### 3.4.2 Comparison of Modified Troe and SRI Methods and Chebyshev Polynomials

The comparison of average and maximum errors of reactions tested for modified Troe, modified SRI, $7 \times 3$ and $9 \times 5$ Chebyshev approximants are shown as Table 3.1. The modified Troe and SRI methods are taken as Venkatesh et al. (75). First, the low- and high-pressure limit rate coefficients are enhanced and expressed as

$$k_0(T) = \left( a_0 + \frac{a_1}{\sqrt{T}} + \frac{a_2}{T} + \frac{a_3}{T^{1.5}} + \frac{a_4}{T^2} \right) \exp \left( -\frac{E_0}{RT} \right)$$

and

$$k_\infty(T) = \left( b_0 + \frac{b_1}{\sqrt{T}} + \frac{b_2}{T} + \frac{b_3}{T^{1.5}} + \frac{b_4}{T^2} \right) \exp \left( -\frac{E_\infty}{RT} \right),$$
where $a_0, a_1, a_2, a_3, E_0, b_0, b_1, b_2, b_3,$ and $E_{\infty}$ are adjustable parameters. Another enhancement is to extend the original single well treatment by explicitly adding the number of wells for different product channels. These modified methods are tested and showed better performance for multi-well chemical-activated reactions than the original Troe and SRI methods.

$C_2H_6$ decomposed to form $C_2H_5 + H$ and $CH_3 + CH_3$ is a typical unimolecular dissociation reaction system. The rate coefficients of later channel are displayed as Figure B.74 which shows typical high- and low-pressure limit and fall-off curves of the unimolecular reaction through the isothermal slices. From Table 3.1, the average errors of all of methods are less than 10% for $C_2H_6$ decomposition and maximum errors are less than 32%. The performances of modified Troe, modified SRI and Chebyshev polynomials of 7 by 3 are similar. The Chebyshev polynomials of 9 by 5 give excellent agreement - less than 2% average errors and 5% maximum errors.

The results of single well chemical-activation reactions of $CH_3 + O_2$ are similar to the $C_2H_6$ reactions. Chebyshev polynomials of 9 by 5 still yields best performance - less than 1% average errors and only 4% maximum errors.

For multi-wells chemical activation reactions of $CH_3OH + O_2$, the results are dramatically different with unimolecular dissociation and single well chemical activation reactions. Rate coefficients as a function of temperature and pressure for first two channels are plotted as Figure B.75 (a) and (b). It is obviously hard to fit results of rate coefficients for multi channels with traditional single well semi-empirical expressions, even extended version of them. The modified Troe method yields about 150%
maximum errors for first the two channels and the modified SRI method yields about 700% maximum errors. The results of two well reactions of $CH_2OH + O_2 \leftrightarrow CH_2O + HO_2$ are the worst than the other one well reactions. The average and maximum errors of Chebyshev 7 by 3 polynomials are about 10% which are still in an acceptable range for kinetic modeling. Chebyshev 9 by 5 polynomials can give excellent performance if more accurate expression is required. The price here is more parameters (45, about double the number for 7 by 3) and much longer CPU time for the fitting (more than four times of CPU time for 7 by 3 fitting). In some complicated systems, the price of CPU time is high for the speed of workstations or midrange mainframes. Therefore, a balanced choice is $7 \times 3$ Chebyshev polynomials. This is also adopted in the kinetic modeling of this study.

The absolute relative errors of these three channels for $7 \times 3$ and $9 \times 5$ Chebyshev polynomials are displayed as Figure B.76, B.77 and B.78. The dramatic improvement using $9 \times 5$ parameters can be seen from the different scale of error axis.
Table 3.1 Comparisons of Average/Maximum Absolute Relative Errors in the Rate Coefficients

<table>
<thead>
<tr>
<th>Reaction</th>
<th>modified Troe</th>
<th>modified SRI</th>
<th>Chebyshev Approximants</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>7 x 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9 x 5</td>
</tr>
<tr>
<td>C_2H_6 &lt;\rightarrow C_2CH_3 + H</td>
<td>0.05 / 0.15</td>
<td>0.04 / 0.09</td>
<td>0.08 / 0.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.02 / 0.05</td>
</tr>
<tr>
<td>C_2H_6 &lt;\rightarrow CH_3 + CH_3</td>
<td>0.10 / 0.29</td>
<td>0.10 / 0.32</td>
<td>0.08 / 0.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.01 / 0.02</td>
</tr>
<tr>
<td>CH_3 + O_2 &lt;\rightarrow CH_3(OO)</td>
<td>0.08 / 0.19</td>
<td>0.05 / 0.19</td>
<td>0.07 / 0.21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.01 / 0.04</td>
</tr>
<tr>
<td>CH_3 + O_2 &lt;\rightarrow CH_2O + OH</td>
<td>0.05 / 0.33</td>
<td>0.04 / 0.24</td>
<td>0.08 / 0.30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.01 / 0.02</td>
</tr>
<tr>
<td>CH_3OH + O_2 &lt;\rightarrow HOC(O)H_2</td>
<td>0.68 / 1.53</td>
<td>0.79 / 6.87</td>
<td>0.10 / 0.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.02 / 0.08</td>
</tr>
<tr>
<td>CH_3OH + O_2 &lt;\rightarrow CH_3O + HO_2</td>
<td>0.62 / 1.45</td>
<td>0.88 / 7.11</td>
<td>0.15 / 0.52</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.02 / 0.06</td>
</tr>
<tr>
<td>CH_3OH + O_2 &lt;\rightarrow HOC(O)H_2 + O</td>
<td>0.39 / 0.80</td>
<td>0.44 / 3.81</td>
<td>0.07 / 0.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.02 / 0.05</td>
</tr>
</tbody>
</table>

3.5 Computer Codes Used / Developed for Kinetic Modeling

3.5.1 ThermCal (developed in this study)

The thermodynamic properties related to this reaction system are evaluated from the literature. When no literature data are available, the values are estimated using ThermCal computer code which is developed in this study and based on THERM (78) computer code.

ThermCal can be used to calculate thermodynamic property data for gas phase radicals and molecules using Benson' Group Additivity method (79). All group contributions considered for a species are recorded and thermodynamic properties are generated in NASA polynomial format (for compatibility with CHEMKIN (80)) in addition to a listing which are more convenient for thermodynamic, kinetic and equilibrium calculation.
ThermCal is a “batch” process by contrast with the “interactive” THERM computer code. The advantages of ThermCal are use of much less memory, much less storage for equivalent document file, and the batch process which makes it portable to other computer platforms. An updated NASA format compatible thermodynamic property file can be easily obtained by a recalculation of the input file by running ThermCal. This, for example, is required whenever the thermodynamic groups been updated. A thermodynamic data base up to C6 for the species with C/H/O elements is developed at NJIT and used for modeling the kinetic scheme of elementary reactions input to the program.

3.5.2 ThermCvt / ThermSrt (developed in this study)

ThermCvt is programmed to convert therm document file (*.doc) to ThermCal type input file. ThermCvt serves to avoid from human key-in error and save a great deal of time. ThermSrt is a sorting program to sort the ThermCal input file by the order specified in therm.cfg configuration file.

3.5.3 RadiCalc (used in this study)

RadiCalc (81) is a computer code that estimate the entropies and heat capacities of radical species and transition structures. The calculation is based upon the properties of these parent molecules. It calculates the effect of loss of vibrational modes (including inversion) and/or changes in barriers to internal rotation and moments of inertia. Databases are compiled from literature data on generic vibrational frequencies, moments of inertia and changes in the barrier to rotation.
3.5.4 ThermFit (used in this study)

ThermFit (82, 83) is a computer code that determine geometric mean frequency as well as a three frequency set. It accepts input in the form of heat capacities versus temperature to 1000 K in addition to the number of vibrational modes and the number of internal rotors in the molecule. This code fits the heat capacity data in the above range to a five parameter harmonic oscillator model and extends the temperature range to 5000 K. An additional method of estimating the vibrational frequencies of a radical is to use MOPAC.

3.5.5 CHEMACT (modified in this study for fitting of Chebyshev Polynomials)

CHEMACT (84) is a computer code that uses the quantum version of RRK theory (QRRK) to evaluate k(E) and modified strong collision theory of Gilbert et al. for falloff as functions of temperature for various channels that can result in addition, combination and insertion reactions as stated in the previous section. Since the rate constants depend on both pressure and temperature of interest in the modeling calculation, a fitting optional of flexible a N x M Chebyshev polynomials expression as discussed previously is now available with user definable data grid. An output file that is CHEMKIN input compatible, is generated for pasting to the input file of modified CHEMGIN drivers for kinetic modeling.

3.5.6 CHEMKIN Interpreter and Library (modified in this study)

The CHEMKIN interpreter has been modified in this study to take the flexible matrix size of Chebyshev polynomials and to generate the appropriate link file for further processing. Subroutines involving kinetic rate coefficient calculation and array size
initialization are also modified to incorporate the Chebyshev polynomials expression. The advantage of modifying the CHEMKIN library instead of involving modification of drivers is any existing drivers can just link with the modified CHEMKIN library to take temperature and pressure dependent Chebyshev polynomials without modification. A couple of drivers have been linked with the new CHEMKIN library and tested. These include a shock tube driver, a constant temperature and pressure driver, and an adiabatic constant pressure driver.
CHAPTER 4

CH$_2$OH + O$_2$ AND CH$_2$O + HO$_2$ REACTION SYSTEMS

4.1 Introduction

The CH$_2$OH + O$_2$ reaction is important in the methanol oxidation reaction system. CH$_2$OH is formed from the abstraction of H from methanol by radicals. The reaction system is also important in evaluating kinetic parameters for O$_2$ reactions with alkyl hydroxyl radicals, that result from OH addition to olefins and aromatic compounds. The second reaction is responsible for creation of hydroxyl peroxy radicals which can further react to form CO$_2$ and acids.

Dr. Lev N. Krasnoperov initials the idea of hydrogen bonded complex for this reaction systems and helps me with the kinetic and thermodynamic calculations on this reaction systems. Without his help, this work can not be done.

Radford (85) measured a rate constant of CH$_2$OH + O$_2$ reaction of $1.2 \times 10^{12}$ cm$^3$ mole$^{-1}$ s$^{-1}$ at 300 K and 0.5 Torr from relative HO$_2$ formation using laser magnetic resonance (LMR) to monitor HO$_2$ and by varying O$_2$ concentration for a fixed residence time. The rate constant measured is similar to a value of $8.4 \times 10^{11}$ cm$^3$ mole$^{-1}$ s$^{-1}$ at 298 K and 1.5 Torr obtained by Wang et al. (86) who detected HO$_2$ from OH photofragment emission. Those two experiments are similar in that they use a discharge flow reactor, but the HO$_2$ measurement are different. Both of the rate constant obtained are lower than other measurements.
A rate constant of $5.7 \times 10^{12}$ cm$^3$ mole$^{-1}$ s$^{-1}$ at 298 K and 0.3 to 0.9 Torr is obtained by Grotheer et al. (87) from CH2OH profiles on a molecular beam sampling mass spectrometer. This reaction forms CH$_2$O as the major channel as concluded from simultaneous CH$_2$O measurements. The MS method was also used by Payne et al. (88) and a rate constant of $5.2 \times 10^{12}$ cm$^3$ mole$^{-1}$ s$^{-1}$ at 298 K and 1.0 Torr was obtained.

Dobe et al. (89) monitored the CH$_2$OH decay by LMR in a isothermal flow system and obtained rate constant of $6.4 \times 10^{12}$ cm$^3$ mole$^{-1}$ s$^{-1}$ at 296 K and 0.52 to 4.88 Torr. No pressure dependent in the pressure range of above was observed.

Pagsberg et al. (90) detected the CH$_2$OH radical by UV absorption in a static reactor and reported a rate constant of $5.3 \times 10^{12}$ cm$^3$ mole$^{-1}$ s$^{-1}$ at 298 K and 760 Torr.

A rate experiment using a discharge flow system at 215 to 300 K and 1.0 Torr was performed by Nesbitt et al. (91). Rate constant was determined by monitoring the decay of CH$_2$OH using mass spectrometry. However, large intercepts in the k vs. O$_2$ plots may indicate a wall problem.

Miyoshi et al. (92) reported a rate constant of $7.0 \times 10^{12}$ cm$^3$ mole$^{-1}$ s$^{-1}$ at 296 K and 1.3 to 5.5 Torr by laser flash photolysis-photoionization mass spectrometry.

For the reaction of CH$_2$O + HO$_2$ → H$_2$(OH)COO which is the reverse of CH$_3$OH + O$_2$, the forward and reverse rate constants are measured by Veyret et al. (93) using the technique of flash photolysis kinetic spectroscopy over a temperature range of 275 to 333 K and a pressure range of 85 to 170 Torr. Same experiment but at 298 K and 55 to 265 Torr was also performed by Veyret et al. (94)
Evleth et al. (95) characterized the CH\textsubscript{2}O + HO\textsubscript{2} reaction at ab initio levels and reported the pathway via hydrogen-bonded complex. The experimental kinetic data can not be explained unless the hydrogen-bonded complex CH\textsubscript{2}O..HOO• is incorporated into the potential energy diagram and reaction system.

4.2 Calculations and Input Parameters

Branching ratios at different temperatures and pressures are calculated using computer code "CHEMACT" discussed in previous chapter. A quantum version of RRK theory (QRRK) was used to evaluate energy dependent rate constants, k(E), for different channels. All input information required for QRRK calculations (set of frequencies or Cp(T)) was obtained from literature data as well as by Group Additivity.

The potential energy diagram is shown as Figure B.79 and the input parameters for QRRK calculations are summarized in Table 4.1 and 4.2. The potential energy diagram for the reaction system includes H(OH)C=O + OH, H\textsubscript{2}C=O + HO\textsubscript{2}, H(OOH)C=O + H, H\textsubscript{2}(OH)CO• product channels with H\textsubscript{2}(OH)COO•, C• H(OOH)OH, CH\textsubscript{2}(OOH)O• and H\textsubscript{2}CO..HOO• intermediates. This is based on existing experimental and theoretical data and on Group Additivity estimation techniques. Analysis indicates that the experimental kinetic data can not be explained unless the hydrogen-bonded complex CH\textsubscript{2}O..HOO• is incorporated into the potential energy diagram and reaction system.
Table 4.1 QRRK Input Parameters for CH$_2$OH + O$_2$ $\leftrightarrow$ [H$_2$(OH)COO•]* $\rightarrow$ Products

<table>
<thead>
<tr>
<th>Reaction</th>
<th>A ($s^{-1}$ or cm$^3$/mol*s))</th>
<th>$E_a$ (kcal/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CH$_2$OH + O$_2$ $\rightarrow$ H$_2$(OH)COO•</td>
<td>6.00E+12</td>
</tr>
<tr>
<td>-1</td>
<td>H$_2$(OH)COO• $\rightarrow$ CH$_2$OH + O$_2$</td>
<td>2.93E+16 T$^{0.54}e^{0.00118T}$</td>
</tr>
<tr>
<td>2</td>
<td>H$_2$(OH)COO• $\rightarrow$ H$_2$(OH)CO• + O</td>
<td>7.55E+14</td>
</tr>
<tr>
<td>3</td>
<td>H$_2$(OH)COO• $\rightarrow$ H$_2$CO..HOO•</td>
<td>4.96E+6 T$^{2.11}e^{0.00069T}$</td>
</tr>
<tr>
<td>-3</td>
<td>H$_2$CO..HOO• $\rightarrow$ H$_2$(OH)COO•</td>
<td>3.76E+7 T$^{1.0}$</td>
</tr>
<tr>
<td>4</td>
<td>H$_2$CO..HOO• $\rightarrow$ H$_2$C=O + HO$_2$ (I)</td>
<td>5.87E+17 T$^{2.68}e^{0.00007T}$</td>
</tr>
<tr>
<td>5</td>
<td>H$_2$(OH)COO• $\rightarrow$ CH$_2$(OOH)O•</td>
<td>6.86E+8 T$^{1.0}$</td>
</tr>
<tr>
<td>-5</td>
<td>CH$_2$(OOH)O• $\rightarrow$ H$_2$(OH)COO•</td>
<td>5.56E+8 T$^{0.84}e^{0.00042T}$</td>
</tr>
<tr>
<td>6</td>
<td>CH$_2$(OOH)O• $\rightarrow$ H$_2$C=O + HO$_3$ (II)</td>
<td>1.40E+14</td>
</tr>
<tr>
<td>7</td>
<td>H$_2$(OH)COO• $\rightarrow$ C•H(OOH)OH</td>
<td>5.90E+9 T$^{1.0}$</td>
</tr>
<tr>
<td>-7</td>
<td>C•H(OOH)OH $\rightarrow$ H$_2$(OH)COO•</td>
<td>4.19E+9 T$^{0.8}e^{0.00064T}$</td>
</tr>
<tr>
<td>8</td>
<td>C•H(OOH)OH $\rightarrow$ H(OH)C=O + OH</td>
<td>3.31E+13</td>
</tr>
<tr>
<td>9</td>
<td>C•H(OOH)OH $\rightarrow$ H(OOH)C=O + H</td>
<td>3.24E+13</td>
</tr>
</tbody>
</table>

frequencies/degenercies (from CPFIT): 415 cm$^{-1}$/6.178, 1506 cm$^{-1}$/5.998, 3198 cm$^{-1}$/2.824

Lennard-Jones parameters: $\sigma = 4.83$ Å, $\epsilon/k = 488$ K

$\langle\Delta E\rangle_{down}$ = 800 cal/mol, Bath gas = N$_2$

$k_1$ A$_1$ taken as double of CC•+O$_2$ addition from J. W. Bozzelli and A. M. Dean, J. Phys. Chem., 1990.

$k_2$ Microscopic Reversibility (MR) with the temperature range of 298 K and 2000 K.

$k_3$ A$_3$ = 2.00E+13 based on addition of O to CH$_3$.

$k_4$ $A_3 = (10^{13.55})(10^{AS/4.6}), \Delta S^d = -6.4; E_a = \Delta H_3 + 2.0$ by transition state calculation, transition state thermo properties based on Evleth et al., J. Phys. Chem., 1993.

$k_5$ $A_3 = (ekT/h)[\exp(\Delta S^d/R)], \Delta S^d = -14.4; E_a = 2.0$ by transition state calculation, transition state thermo properties based on Evleth et al., J. Phys. Chem., 1993.

$k_6$ Based on the study of CH$_2$O + HO$_2$ system in this paper.

$k_7$ $A_5 = (ekT/h)[\exp(\Delta S^d/R)], \Delta S = -8.6$ (loss of 2 rotors); $E_a = \Delta H_3 + \text{ring strain (4.7) + E}_{\text{abstraction}} (3.0)$, ring strain from J. W. Bozzelli and A. M. Dean, J. Phys. Chem., 1990.

$k_8$ MR with the temperature range of 298 K and 2000 K.

$k_7$  \[ A_7 = (ekT/h)[\exp(\Delta S^o/R)](2), \Delta S = -4.3 \text{ (loss of 1 rotors)}; \quad Ea_7 = \Delta H_7 + \text{ring strain } (23) + E_{\text{abstraction}} (8.6), \text{ ring strain from J. W. Bozzelli and A. M. Dean, J. Phys. Chem., 1990.} \]

$k_{7.8}$  \[ \text{MR with the temperature range of 298 K and 2000 K.} \]

$k_8$  \[ A_8 \text{ from MR with } A_{8.9} = 2.7E+12, \text{ which is one half of the rate constant for addition of OH to } C_2H_4, \text{ Atkinson et al., J. Phys. Chem., 1989; } Ea_8 \text{ from Soto and Page, Chem. Phys. 1991.} \]

$k_9$  \[ A_9 \text{ from MR with } A_{9.10} = 1.46E+13, \text{ which is one half of the rate constant for addition of H to } C_2H_4 \text{ and } Ea_9 = \Delta H_9 + 2.7, \text{ based on NIST fitting.} \]
### Table 4.2 QRRK Input Parameters for H₂C=O + HO₂ ↔ [H₂CO..HOO•]* → Products

<table>
<thead>
<tr>
<th>Reaction</th>
<th>A (s⁻¹ or cm³/(mol·s))</th>
<th>Eₐ (kcal/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 H₂C=O + HO₂ → H₂CO..HOO•</td>
<td>5.00E+10</td>
<td>0.0</td>
</tr>
<tr>
<td>-1 H₂CO..HOO• → H₂C=O + HO₂</td>
<td>5.87E+17 T⁻².⁶⁸e⁻₀.⁰⁰₀₀₀₇₇T</td>
<td>7.2</td>
</tr>
<tr>
<td>2 H₂CO..HOO• → H₂(OH)COO•</td>
<td>3.76E+7 T⁻¹.⁰</td>
<td>1.3</td>
</tr>
<tr>
<td>-2 H₂(OH)COO• → H₂CO..HOO•</td>
<td>4.96E+6 T²⁻¹¹e⁻₀.⁰₀₀₆₉T</td>
<td>8.6</td>
</tr>
<tr>
<td>3 H₂(OH)COO• → CH₂OH + O₂</td>
<td>9.00E+14</td>
<td>34.4</td>
</tr>
<tr>
<td>4 H₂(OH)COO• → H₂(OH)COO• + O</td>
<td>7.55E+14</td>
<td>57.5</td>
</tr>
<tr>
<td>5 H₂(OH)COO• → CH₂(OOH)O•</td>
<td>6.86E+8 T⁻¹.⁰</td>
<td>21.8</td>
</tr>
<tr>
<td>-5 CH₂(OOH)O• → H₂(OH)COO•</td>
<td>5.56E+8 T⁻⁰.₈₄e⁻₀.⁰₀₀₄₂T</td>
<td>7.7</td>
</tr>
<tr>
<td>6 CH₂(OOH)O• → CH₂O + HO₂</td>
<td>1.40E+14</td>
<td>5.0</td>
</tr>
<tr>
<td>7 H₂(OH)COO• → C•H(OOH)OH</td>
<td>5.90E+9 T⁻¹.⁰</td>
<td>37.1</td>
</tr>
<tr>
<td>-7 C•H(OOH)OH → H₂(OH)COO•</td>
<td>4.19E+9 T⁻⁰.₈₄e⁻₀.⁰₀₀₆₄T</td>
<td>31.6</td>
</tr>
<tr>
<td>8 C•H(OOH)OH → H(OH)C=O + OH</td>
<td>3.31E+13</td>
<td>1.0</td>
</tr>
<tr>
<td>9 C•H(OOH)OH → H(OOH)C=O + H</td>
<td>3.24E+13</td>
<td>22.5</td>
</tr>
</tbody>
</table>

**frequencies/degenercies (from CPFIT):** 188 cm⁻¹/5.857, 1308 cm⁻¹/5.714, 2952 cm⁻¹/3.430

**Lennard-Jones parameters:** \( \sigma = 4.83 \text{ Å}, \epsilon/k = 488 \text{ K} \)

\(<\Delta E>_{\text{down}} = 800 \text{ cal/mol}, \text{ Bath gas} = N₂\)

\(k₁\) This study, estimated from the reaction of CH₂O+HO₂ = CQ·H₂OH, Veyret et al., J. Phys. Chem., 1989.

\(k₂\) Microscopic Reversibility (MR) with the temperature range of 298 K and 2000 K.

\(k₃\) \(A₃\) from MR with \(A₃\) taken as double of CC•O₂ addition from J. W. Bozzelli and A. M. Dean, J. Phys. Chem., 1990.

\(k₄\) \(A₄\) from MR with \(A₄=2.00E+13\) based on addition of O to CH₃.

\(k₅\) \(A₅\) from MR with \(A₅=6.05E+11\) and \(Eₐ₅=\Delta H₃ + \text{ring strain (4.7)} + E_{\text{ abstraction}} (3.0)\), ring strain from J. W. Bozzelli and A. M. Dean, J. Phys. Chem., 1990.

\(k₆\) MR with the temperature range of 298 K and 2000 K.

\(k₇\) \(A₇\) from MR with \(A₇=6.05E+11\) and \(Eₐ₇=\Delta H₄ + 5.0\), based on addition of HO₂ to C₂H₄ by Tsang's recommendation, J. Phys. Chem. Ref. Data, 1987.
\[ k_7 \quad A_7 = \left( \frac{ekT}{h} \right) \left[ \exp \left( \frac{\Delta S^e}{R} \right) \right] (2), \quad \Delta S = -4.3 \text{ (loss of 1 rotors)}; \quad \text{Ea}_7 = \Delta H_r + \text{ring strain} \text{ (23) + E}_{\text{abstraction}} (8.6), \text{ ring strain from J. W. Bozzelli and A. M. Dean, J. Phys. Chem., 1990.} \]

\[ k_7 \quad \text{MR with the temperature range of 298 K and 2000 K.} \]

\[ k_8 \quad A_8 \text{ from MR with } A_8 = 2.7E+12, \text{ which is one half of the rate constant for addition of OH to } C_2H_4, \text{ Atkinson et al., J. Phys. Chem., 1989; } \text{Ea}_8 \text{ from Soto and Page, Chem. Phys. 1991.} \]

\[ k_9 \quad A_9 \text{ from MR with } A_9 = 1.46E+13, \text{ which is one half of the rate constant for addition of H to } C_2H_4 \text{ and Ea}_7 = \Delta H_r + 2.7, \text{ based on NIST fitting.} \]
The input parameters required for QRRK analysis are discussed in previous chapter except the estimation of high pressure rate constants. Elimination or dissociation rate constants are calculated thermochemically from the reverse (e.g. addition, combination) reactions and microscopic reversibility if literature data is not available. The addition or combination rate constants are taken from literature or estimated from the trends in homologous series of this type of reactions or analogous reaction. Isomerization rate constants are analyzed via Transition-State-Theory (TST) and thermochemical kinetic methods of Benson. The high pressure A factor is calculated as \( A = (ekT/h)[\exp(\Delta S^\# /R)] \), where \( \Delta S^\# \) is the entropy difference from reactant to transition state. The activation energy is calculated as the sum of abstraction barrier, ring strain, plus enthalpy of isomerization where the reaction is endothermic.

4.3 Results and Discussion
The results of \( \text{CH}_2\text{OH} + \text{O}_2 \) calculations are displayed in Figure B.80 to B.84 and the comparison with experimental data is in Figure B.85. There are two pathways to form the \( \text{CH}_2\text{O} + \text{HO}_2 \) products. One is via \( \text{H}_2\text{C(OO)}*\text{OH} \) isomerization to \( \text{CH}_2(\text{OOH})\text{O}* \), which then decomposes to \( \text{CH}_2\text{O} + \text{HO}_2 \). The rate constant via this pathway is about two orders of magnitude smaller than the pathway via hydrogen-bonded complex and reported experimental data at low pressure and temperature. The dominance of hydrogen-bonded channel is why the experimental kinetic data can not be explained unless the hydrogen-bonded complex \( \text{CH}_2\text{O}..\text{HO}_2* \) is incorporated into the potential energy diagram and reaction system. The formation of \( \text{CH}_2\text{O} + \text{HO}_2 \) adduct is the
dominate channel for pressures < 1 atm and temperature < 2000 K. The stabilization adduct of H$_2$C(00•)OH formation is important for pressures > 3 atm at 298 K and for $p > 25$ atm at 900 K. The Acid formation adduct is important for $T > 2000$ K.

Figure B.86 to B.89 show the results of CH$_2$O + HO$_2$ reaction. A comparison of CH$_2$O + HO$_2$ $\leftrightarrow$ H$_2$(OH)COO• with forward and reverse experimental data is plotted in Figure B.90 which shows good agreement for both directions. Adduct formation is important for pressures above 0.01 atm at 298 and $p$ above 0.15 atm at 900 K. Here, formation of CH$_2$OH + O$_2$ and HCO$_3$H + H surpasses stabilization in importance at 0.005 atm and 900 K, and increase in importance with increasing temperature.

Results of this study confirm the importance of hydrogen bonded complex for this reaction system and suggest the possible importance of such a complexes in other, similar systems. The reaction description is based entirely on "complex formation" mechanism without "direct reaction". Results of these calculations, which are based on limited, low-pressure and narrow temperature range experimental data, can be used in combustion models over a wide temperature and pressure range.


CHAPTER 5

METHANOL PYROLYSIS AND OXIDATION

5.1 Methanol Pyrolysis

5.1.1 Experimental Results

Methanol pyrolysis is studied over a pressure range from 1 to 10 atm at 1073 K under isothermal reactor conditions. The reactant concentrations at different pressures are listed in Table 5.1.

<table>
<thead>
<tr>
<th>Pressure</th>
<th>Methanol Molar Fraction</th>
<th>Argon Molar Fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 atm</td>
<td>0.03950</td>
<td>0.96050</td>
</tr>
<tr>
<td>3 atm</td>
<td>0.01317</td>
<td>0.98683</td>
</tr>
<tr>
<td>5 atm</td>
<td>0.00790</td>
<td>0.99210</td>
</tr>
<tr>
<td>8 atm</td>
<td>0.00494</td>
<td>0.99506</td>
</tr>
<tr>
<td>8 atm</td>
<td>0.00250</td>
<td>0.99750</td>
</tr>
<tr>
<td>10 atm</td>
<td>0.00395</td>
<td>0.99605</td>
</tr>
</tbody>
</table>

Experimental results on methanol pyrolysis at 1073 K and a pressure range from 1 to 10 atm are displayed in Appendix B, Figure B-7 to B-12. Methanol decay curves at different pressures are displayed as normalized concentrations (C/Co) versus reaction time in Figure B.91.

Methanol decays and forms the intermediate stable products: formaldehyde which then decomposes to carbon monoxide. The decay rate of methanol increases with increasing pressure, however, the differences between 5, 8 and 10 atmospheres are
less than 5%. Normalized concentrations (C/Co) of methane formation are about the same for different pressures.

5.1.2 Kinetic Modeling

The initial decomposition steps of methanol below are analyzed by QRRK formalism for k(E) and modified strong collision theory of Gilbert et al. for falloff. The potential energy diagram is shown in Figure B.92 and the input parameters for QRRK analysis are in Appendix C. The barriers of HCOH + H₂, ¹CH₂ + H₂O and CH₂O + H₂ channels are taken from the study of Walch (96) using complete-active-space self-consistent-field (CASSCF) / internally contracted configuration-interaction (CCI), ab initio calculations. The reverse of ¹CH₂ + H₂O channel is found to have no barrier. The barriers of the reverse of HCOH + H₂ and CH₂O + H₂ channels are calculated as -5.2 and 1.7 kcal/mol with respect to the energy level of CH₃ + OH channel. The barriers of HCOH + H₂, ¹CH₂ + H₂O and CH₂O + H₂ channels are then can be derived as 87.4, 90.9 and 94.3 kcal/mol respectively. The unimolecular dissociations show that CH₃ + OH channel is dominant at moderate temperature (i.e. about 1000 K) and HCOH + H₂ channel becomes important when temperature increases.

\[
\begin{align*}
\text{CH₃OH} & \rightarrow \text{CH₃ + OH} \quad \Delta H_{\text{ran}} = 92 \text{ kcal/mol} \\
& \rightarrow \text{HCOH + H₂} \quad \Delta H_{\text{ran}} = 71 \text{ kcal/mol} \\
& \rightarrow ¹\text{CH₂ + H₂O} \quad \Delta H_{\text{ran}} = 92 \text{ kcal/mol} \\
& \rightarrow \text{CH₂O + H₂} \quad \Delta H_{\text{ran}} = 74 \text{ kcal/mol} \\
& \rightarrow \text{CH₂OH + H} \quad \Delta H_{\text{ran}} = 97 \text{ kcal/mol} \\
& \rightarrow \text{CH₃O + H} \quad \Delta H_{\text{ran}} = 104 \text{ kcal/mol}
\end{align*}
\]
Abstraction reactions: two H atom abstraction reactions are found to be the most significant after the initial decomposition of methanol to form CH$_2$OH and CH$_3$O radicals.

\[
\text{CH}_3\text{OH} + \text{H} \rightarrow \text{CH}_2\text{OH} + \text{H}_2
\]

\[
\text{CH}_3\text{OH} + \text{H} \rightarrow \text{CH}_3\text{O} + \text{H}_2
\]

There are no well accepted rate coefficients for \(\text{CH}_3\text{OH} + \text{H}\) abstractions available. Warantz's data are adopted in Norton's mechanism and this study. The ratio for forming \(\text{CH}_2\text{OH}\) and \(\text{CH}_3\text{O}\) is 4 to 1. The \(\text{CH}_2\text{OH}\) and \(\text{CH}_3\text{O}\) radicals decompose beta scission to form formaldehyde:

\[
\text{CH}_2\text{OH} + \text{M} \rightarrow \text{CH}_2\text{O} + \text{H} + \text{M}
\]

\[
\text{CH}_3\text{O} + \text{M} \rightarrow \text{CH}_2\text{O} + \text{H} + \text{M}.
\]

These two unimolecular decomposition reactions are pressure dependent and are analyzed by QRRK analysis.

Formaldehyde, important intermediate decays mostly by H abstraction reaction:

\[
\text{CH}_2\text{O} + \text{H} \rightarrow \text{HCO} + \text{H}_2.
\]

The decomposition reaction of \(\text{HCO} + \text{M}\) \(\rightarrow\) \(\text{CO} + \text{H} + \text{M}\) is found to be the dominant channel to form the pyrolysis final product of carbon monoxide.

Comparisons of modeling and experimental results at 1 and 3 atm are illustrated in Figure B.93 and B.94 and show good agreement for both reactant decay and main product formations at 1 atm. At 3 atm, the modeling result is a little faster than experimental data. Methane formations are under predicted at both conditions. The model predicts even faster decay of methanol than experimental at higher pressure.
5.2 Methanol Oxidation

5.2.1 Experimental Results

Methanol oxidation experiments are performed with two reactant ratio sets which represent stoichiometric and fuel lean conditions over temperature range of 873 to 1073 K and a pressure range of 1 to 5 atm. The reactant concentration ratios are:

Fuel Lean \( \phi = 0.75 \) \( \text{CH}_3\text{OH} : \text{O}_2 : \text{AR} = 0.0078 : 0.0156 : 0.9766 \)

Stoichiometric \( \phi = 1.0 \) \( \text{CH}_3\text{OH} : \text{O}_2 : \text{AR} = 0.0078 : 0.0117 : 0.9805 \).

Results of methanol oxidation are shown in Figures B-13 to B-60, which plot the mole fraction as a function of reaction time and temperature, at conditions ranging from of 873 to 1073 K, pressures of 1, 3 and 5 atm and two equivalence ratios of 1.0 and 0.75.

Changes in methanol conversion, intermediates and products profiles under conditions of excess oxygen (equivalence ratio = 0.75) are small relative to reaction at stoichiometric conditions for all temperatures and pressures studied. Methanol starts to decay at 923 K and 100% conversion is observed at 1073 K, at 1 atm and reaction time of 0.15 second. For 3 atm and reaction time of 0.6 second, methanol decay starts at 873 K and is 95% converted at 923 K. For 5 atm and reaction time of 0.6 second, methanol conversion is 5% at 873 K and 99% at 923 K.

At increased temperatures, maximum formaldehyde levels decrease and this intermediate only exists for a short residence time (ca. 0.2 seconds). CO increases rapidly with the conversion of methanol. The formation of \( \text{CO}_2 \) starts to occur when
significant amounts of CO produced. At higher temperature, oxidation of CO occurs rapidly to the final product CO₂.

Methane is observed as a minor product. Impurity, methane, has been found in the supply gas at a level of few ppm. The experimental result of methane is calibrated by the blank (by-pass) correction. Due to the low concentration of methane, hydrocarbon molecular weight growth species are not detected (probably below few ppm, instrument detection limit).

5.2.2 Kinetic Modeling

After methanol decomposition occurs, the following reactions play important roles for the methanol decay in the oxidation condition and intermediate temperature. In contrast to pyrolysis where H atom is important, CH₃OH + OH is now the most responsible reaction after unimolecular decomposition for the methanol decay.

\[
\begin{align*}
\text{CH}_3\text{OH} + \text{OH} & \rightarrow \text{CH}_2\text{OH} / \text{CH}_3\text{O} + \text{H}_2\text{O}_2 \\
\text{CH}_3\text{OH} + \text{HO}_2 & \rightarrow \text{CH}_2\text{OH} / \text{CH}_3\text{O} + \text{H}_2 \\
\text{CH}_3\text{OH} + \text{H} & \rightarrow \text{CH}_2\text{OH} / \text{CH}_3\text{O} + \text{H}_2
\end{align*}
\]

The reaction of CH₂OH radical with O₂ to form formaldehyde and hydroperoxy radical is the most important reaction for the production of important intermediate, formaldehyde.

\[
\begin{align*}
\text{CH}_2\text{OH} + \text{O}_2 & \rightarrow [\text{OOCH}_2\text{OH}]# \rightarrow \text{OCH}_2\text{OH} + \text{O} \\
& \rightarrow [\text{HOOC.HOH}]# \rightarrow \text{HCO}_2 + \text{OH} \\
& \rightarrow [\text{CH}_2\text{O...HO}_2]# \rightarrow \text{CH}_2\text{O} + \text{HO}_2 \\
& \rightarrow [\text{HOOCH}_2\text{O}.]# \rightarrow \text{CH}_2\text{O} + \text{HO}_2
\end{align*}
\]
\[ \text{CH}_2\text{OH} + \text{O}_2 \quad \text{(Reverse - No Rxn)} \]
\[ \rightarrow \text{.OOCH}_2\text{OH} \quad \text{(Stabilization)} \]

A more detailed discussion of the CH\(_2\text{OH} + \text{O}_2\) reaction is described in Chapter 4. There are two pathways to form CH\(_2\text{O} + \text{HO}_2\). One through the isomerization of [\text{.OOCH}_2\text{OH}]^\# to [\text{HOOCH}_2\text{O}]^\# then forms CH\(_2\text{O} + \text{HO}_2\) when another through the isomerization of .\text{OOCH}_2\text{OH} to [\text{CH}_2\text{O}...\text{HOO}]^\#, hydrogen-bonded complex. The experimental kinetic data cannot be explained through the first route unless the hydrogen-bonded complex CH\(_2\text{O}..\text{HOO}^\bullet\) is characterized and incorporated into the potential energy diagram and reaction scheme of Evleth et al.

Hydroperoxy radical reacts with hydrogen atom to form hydroxyl radical. This is found to be an important source of hydroxyl radical for the CH\(_3\text{OH} + \text{OH}\) abstraction reaction.

Formaldehyde then decays mostly by the reaction of CH\(_2\text{O} + \text{OH} \rightarrow \text{HCO} + \text{H}_2\text{O}\). The decomposition reaction of \text{HCO} + M \rightarrow \text{CO} + \text{H} + M\) is found to be the dominant channel to form carbon monoxide and \text{CO} + \text{O} + M \rightarrow \text{CO}_2 + M\) to form final product of carbon dioxide.

Comparisons of the model and the experimental results at 5 atm and two equivalence ratios: stoichiometric (\(\phi = 1\)) and fuel lean (\(\phi = 0.75\)), are in Figures B.95 and B.96. These show reasonable agreement for reactant decay and main product formation. Reaction changes in methanol conversion and the intermediates or products profiles under conditions of excess oxygen (equivalence ratio = 0.75) are small relative to those illustrated at stoichiometric conditions throughout all
temperatures and pressures studied, in both the experimental data and model predictions.

Figures B.97 to B.99 shows the comparisons of modeling and experimental results for reactant conversion and product distribution versus temperature at 5 atm, average residence times of 0.8, 1.0 and 1.5 seconds respectively. The reactant decay and major product distributions are in reasonable agreement but the model predicts faster CO oxidation at higher temperature (ca. > 1000 K). The methane impurity is possibly why the methane been observed even at more than 99% of methanol been convert to CO₂.

At 3 atmospherics pressure, modeling result is a little slower than the observed experimental data. Figure B.100 shows a time delay of 0.15 second for modeling result relative to experimental data. In general, there is a longer time delay for the model compared with experimental data at lower pressures. At lower pressure and temperature, decay will occur at longer residence times which are unfortunately, out of the flow range of the flow controller used in this study.

5.3 Summary

The pyrolysis and oxidation of methanol follow the decomposition order of CH₃OH → CH₃O → CO → CO₂. CO₂ is observed in the oxidation studies only. The important reactions can be summarized as below.
CHAPTER 6

EFFECTS OF METHANOL ADDITION TO METHANE OXIDATION

6.1 Experimental Results

Experiments and model calculations were performed to determine the effects of methanol on methane oxidation. The total mole fraction of CH$_4$ + CH$_3$OH was held constant at 1.56%. Experiments on methane oxidation for pure methane and for mixture of CH$_4$ / CH$_3$OH at 100% increments of CH$_3$OH up to equal molar were conducted. The results show that addition of CH$_3$OH to CH$_4$ oxidation at a near constant fuel equivalence ratio, dramatically increased CH$_4$ conversions.

Experimental results on oxidation of methane / methanol mixtures are shown as Figures B-61 to B-72. The presence of a small quantity of methanol increases the rate of CH$_3$OH decay. Methanol is observed to have a significant acceleration effect on methane oxidation. The experimental results are obvious and a uniform trend is observed over the temperature, pressure and relative concentration range of the methane / methanol mixtures.

The ratio of methane / methanol is varied with a constant total mixture concentration and equivalence ratio to understand and verify the methanol acceleration effect on methane oxidation. At equal molar mixtures conversion of methane is enhanced by 50% at 1.5 seconds reaction time and further enhancement is slight.

Molecular weight growth species: ethane and ethylene are also observed which result from CH$_3$ + CH$_3$ recombination reactions.
6.1 QRRK analysis of CH$_3$ + O$_2$ reactions

CH$_3$ is the initial radical intermediate in CH$_4$ oxidation. The major reaction of CH$_3$ at low CH$_3$ concentration is reaction with O$_2$. A potential energy level diagram for CH$_3$ reaction with O$_2$ is shown in Figure B.101 and a reaction scheme is as below.

\[
\begin{align*}
\text{CH}_3O + O & \quad \uparrow \\
\text{CH}_3 + O_2 & \Leftrightarrow [\text{CH}_3OO]^* \Leftrightarrow [\text{CH}_2OOH]^* \rightarrow \text{CH}_2O + \text{OH} \\
& \quad \downarrow \\
\text{CH}_2OO & \quad \text{CH}_2OOH
\end{align*}
\]

CH$_3$ reaction with O$_2$ forms [CH$_3$OO]* complex. The [CH$_3$OO]* complex can either stabilize, form CH$_3$O + O, isomize to [CH$_2$OOH]* or go back to reactants. The [CH$_2$OOH]* isomer can also isomerize back to [CH$_3$OO]*, stabilize or form the products: CH$_2$O + OH.

High-pressure-limit input parameters for the CH$_3$ + O$_2$ combination to form the CH$_3$OO complex, are taken from Cobos et al. (97). The input parameters for dissociation of the complex back to reactants are calculated by microscopic reversibility with the temperature range 298 to 2000 K. Parameters for the CH$_3$O + O product channel are obtained from an estimate of $5.0 \times 10^{13}$ cm$^3$ mol$^{-1}$ s$^{-1}$ for the high-pressure recombination rate constant via microscopic reversibility. The A factor of isomerization is taken as Transition State Theory loss of one rotor ($\Delta S^t = -4.3$ cal/mol) and the degeneracy of 3. The activation energy is estimated as a sum of reaction enthalpy, ring strain (26 kcal/mol) and H abstraction (6 kcal/mol). The A factor of exit channel, CH$_2$O + OH, is calculated from microscopic reversibility with reverse taken as
the addition of OH to CH$_3$CHO by Semmes et al. (98) with Ea is estimated as 2 kcal/mol from the intrinsic activation energy expected for OH addition.

Comparisons of experimental and QRRK results for CH$_3$ + O$_2$ → CH$_3$OO in argon and nitrogen bath gas are displayed in Figure B.102 and CH$_3$O + O and CH$_3$O + OH channels are in Figure B.103. Results for CH$_3$ + O$_2$ to all channels at atmospheric pressure are displayed in Figure B.104.

Here one can see a dramatic difference from the CH$_2$OH + O$_2$ reaction system; there is no low energy exit channel such as the CH$_2$O + HO$_2$ product set in CH$_3$ + O$_2$.

6.2 Comparison of Methane oxidation versus methanol oxidation

The comparison of methane and methanol oxidation can be described in three stages:

6.2.1 Initiation

Several channels of CH$_3$OH unimolecular decomposition are much lower in energy than CH$_4$ and therefore faster than CH$_4$ decomposition.

\[
\begin{align*}
\text{CH}_3\text{OH} & \rightarrow \text{CH}_3 + \text{OH} & \Delta H_{\text{rxn}} = 92 \text{ kcal/mol} \\
& \rightarrow \text{HCOH} + \text{H}_2 & \Delta H_{\text{rxn}} = 71 \text{ kcal/mol} \\
& \rightarrow \text{CH}_3 + \text{H}_2\text{O} & \Delta H_{\text{rxn}} = 92 \text{ kcal/mol} \\
& \rightarrow \text{CH}_2\text{O} + \text{H}_2 & \Delta H_{\text{rxn}} = 74 \text{ kcal/mol} \\
\text{CH}_4 & \rightarrow \text{CH}_3 + \text{H} & \Delta H_{\text{rxn}} = 103 \text{ kcal/mol}
\end{align*}
\]

6.2.2 Propagation

H–CH$_2$OH has a lower bond energy (97 kcal/mol) than H–CH$_3$ (105 kcal/mol). CH$_2$OH forms easier than CH$_3$ (from CH$_4$) by decomposition as well as through
abstraction reactions. CH$_2$OH also has a much more rapid unimolecular decomposition channel relative CH$_3$.

\[
\begin{align*}
\text{CH}_3 & \rightarrow \text{CH}_2 + \text{H} \quad \Delta H_{\text{rea}} = 108 \text{ kcal/mol} \\
\text{CH}_2\text{OH} & \rightarrow \text{CH}_2\text{O} + \text{H} \quad \Delta H_{\text{rea}} = 30 \text{ kcal/mol}
\end{align*}
\]

Therefore, propagation reactions of CH$_3$OH oxidation are faster than CH$_4$ oxidation too.

6.2.3 Oxidation

Methane oxidation, CH$_3$ + O$_2$ which will be discussed in next section is much slower than that of CH$_2$OH + O$_2$. The difference between CH$_2$OH + O$_2$ and CH$_3$ + O$_2$ reaction system is that CH$_2$OH + O$_2$ has a low energy exit channel. Barrier is below the energy level of initial CH$_2$OH + O$_2$ reactants to products, CH$_2$O + HO$_2$. This explains the faster CH$_2$OH + O$_2$ reaction.

6.3 Acceleration of Methane oxidation by Added Methanol

Comparisons of modeling and experimental results for oxidation of methane /methanol mixtures with equal initial concentration at 873 K and 5 atm in stoichiometric reaction condition are shown in Figure B.105 and B.106. Methane conversion is more than 50% at 1.5 seconds, when there is almost no conversion at all, without methanol under the same conditions.

Methanol decomposes faster than methane as stated in previous section. The important intermediate, CH$_2$OH, then reacts with O$_2$ and undergoes unimolecular decay to accelerate the radical pool production. The formation of OH and HO$_2$ radicals accelerated and the OH initiates methane abstraction reactions.
Figure B.107 shows the methanol acceleration effect on methane oxidation at varied ratios of methane/methanol (100/0, 90/10, 85/15, 75/25, 50/50) but constant total initial mixture concentration and equivalence ratio for both the experimental and modeling results. The more methanol added, the faster is methane conversion. 50% methane conversion is reached at about 1.5 seconds reaction time with equal initial concentration of methane and methanol, when there is no decay of methane with no methanol added.

Figure B.108 shows the effect of methane added to methanol oxidation at 873 K and 5 atm in stoichiometric (overall) condition is small. Comparisons of methanol decay for 0.78% methanol (neat methanol) and a mixture of 0.39% methane and 0.39% methanol, initial concentration, shows very small difference in conversion results. Comparing the 0.39% and 0.78% methanol oxidation in Figures B.108 and B.109, we observe effectively the same CH₃OH conversion. There is very little effect of adding methane to methanol oxidation.
CHAPTER 7

MTBE (METHYL TERT-BUTYL ETHER) OXIDATION

7.1 Pyrolysis of MTBE

7.1.1 Experimental Results

MTBE pyrolysis experiments have been performed in collaboration with Chiung-Ju Chen (NJIT) at atmospheric pressure and over a temperature range of 873 and 948 K with a 0.5% constant initial concentration using argon as bath gas. The pyrolysis of MTBE is dominated by the unimolecular decomposition of MTBE at low to intermediate temperatures of this study. That this \( \text{C}_3\text{COC} \rightarrow \text{C}_2\text{C} = \text{C} + \text{CH}_3\text{OH} \) reaction dominates, is verified in this study by a combination of QRRK analysis and mechanism sensitivity analysis.

The experimental data of MTBE decay is plotted as \(-\ln (C/\text{C}_0)\) versus reaction time in Figure B.110, in order to get the overall reaction rate constant of MTBE pyrolysis. The regression results are in Table 7.1.

Table 7.1 Regression results of overall rate constant for MTBE pyrolysis experimental data

<table>
<thead>
<tr>
<th>Temperature / K</th>
<th>( k_{\text{MTBE pyrolysis}} / \text{s}^{-1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>873</td>
<td>0.167</td>
</tr>
<tr>
<td>898</td>
<td>0.493</td>
</tr>
<tr>
<td>923</td>
<td>1.331</td>
</tr>
<tr>
<td>948</td>
<td>2.936</td>
</tr>
</tbody>
</table>

7.1.2 Unimolecular Dissociation of \( \text{C}_3\text{COC} \)

The initiation reactions are unimolecular decompositions of MTBE:
The MTBE dissociation reaction is the important reaction in both the MTBE pyrolysis and oxidation reaction systems. The energy level diagram and input parameters for the chemical activation calculations are shown in Figure B.111 and Appendix C.

The channels involving loss of a H atom are of no importance due to the higher energy barrier of breaking the C—H bond compared with C—O and C—C bonds. The relative A factors for C—H bond cleavage are also lower, further reducing the relative probability for these reactions. The calculation results indicate that the dominant channels are the dissociations to C\textsubscript{2}C=\textsubscript{C} + CH\textsubscript{3}OH and C\textsubscript{3}C- + CH\textsubscript{3}O. The apparent rate constants of both reactions are consistent with their high pressure limits when the temperature is below than 800 K and the pressure higher than 0.001 atm.

The unimolecular elimination, channel (1), is dominant when the temperature is lower than \(~900\) K. This is consistent with Daly et al. (1) and Choo et al. (2) conclusion's that the decomposition reaction of MTBE is a four-center molecular elimination at temperatures \(~700\) K.
The effect of temperature on the falloff behavior of dissociation reactions is
dramatic. Increasing the temperature from 800 to 1200 K for $C_3CO + \text{CH}_3\text{O} \rightarrow C_3C^+ + \text{CH}_3\text{O}$ shifts the falloff curve by 4 orders of magnitude toward higher pressure. The
result indicates the dramatic differences between application of the present approach
and use of the assumption that the dissociation is always at the high pressure limit. The
difference between the actual dissociation rate constant and the high pressure limit
increases when the temperature is above 1000 K at the atmospheric pressure.

The apparent rate constant of $C_3CO \rightarrow C_2=CH + \text{CH}_3\text{OH}$ reaction is not in
the high pressure limit regime under temperature range from 873 to 948 K at 1 atm.
Therefore, a fitting process is required to get the apparent rate constants and the input
high pressure limit parameters for the conditions of interested. The results of this
optimization are shown below:

\[
C_3CO \rightarrow C_2=CH + \text{CH}_3\text{OH} \quad k_{\text{app}} = 1.55 \times 10^{15} T^{-0.287} \exp(-60186 / RT)
\]
\[
k_{\infty} = 2.2 \times 10^{14} \exp(-60,000 / RT).
\]

Note that the apparent rate constants are fitted between temperatures of 873
and 973 K at atmospheric pressure.

**7.1.3 Comparison of Modeling and Experimental Results**

A comparison of modeling results and experimental data is displayed in Figure B.112
for validation. Good agreement between the model and experimental results is shown
for the pyrolysis of MTBE in different temperatures.
7.2 Oxidation of MTBE

7.2.1 Kinetic Modeling

The MTBE dissociation reaction is the initial reaction in the MTBE pyrolysis and also in this oxidation reaction system and discussed in detail in previous section. Initial abstraction reactions of MTBE are mainly by radical species: H, O, OH, HO2, CH3O, and C3C.

\[
\begin{align*}
\text{C}_3\text{COC} + \text{H} & \rightarrow \text{C}_3\text{COC} \cdot + \text{H}_2 \\
& \rightarrow \text{C}_3\text{COC} + \text{H}_2 \\
\text{C}_3\text{COC} + \text{O} & \rightarrow \text{C}_3\text{COC} \cdot + \text{OH} \\
& \rightarrow \text{C}_3\text{COC} + \text{OH} \\
\text{C}_3\text{COC} + \text{OH} & \rightarrow \text{C}_3\text{COC} \cdot + \text{H}_2\text{O} \\
& \rightarrow \text{C}_3\text{COC} + \text{H}_2\text{O} \\
\text{C}_3\text{COC} + \text{HO}_2 & \rightarrow \text{C}_3\text{COC} \cdot + \text{H}_2\text{O}_2 \\
& \rightarrow \text{C}_3\text{COC} + \text{H}_2\text{O}_2
\end{align*}
\]

The elementary reaction rate parameters for abstraction reactions are based upon literature survey, thermodynamics and generic A factors with Evans Polanyi plots for Ea's.

The thermal decompositions (Beta scissions) of C3COC, C3-COC, C3C, and C2C-OC radicals and abstractions by other radicals are the important reactions for the MTBE oxidation system. These radical decomposition rate constants are determined from the QRRK calculation as described in the previous section.
\[ C_3\text{C} \rightarrow C_2\text{CC} \rightarrow C=\text{CC} + \text{CH}_3 \]

\[ \downarrow \]

\[ C_2=\text{C} + \text{H} \]

\[ C_3\text{COC} \text{ and } C_3\text{COC} \text{ decomposition reactions are also important in MTBE reaction system.} \]

\[ C_3\text{COC} \leftrightarrow C_3\text{COC} \rightarrow C_2\text{C} = \text{C} + \text{CH}_3\text{O} \]

\[ \downarrow \]

\[ C_3\text{C} + \text{CH}_3\text{O} \rightarrow C=\text{C(C)OC} + \text{CH}_3 \]

The corresponding radicals of iso-propyl methyl ether, second radical, are also the important species in MTBE oxidation system. The following reactions contribute to the formation of acetone and propene.

\[ C_2\text{C}=\text{O} + \text{CH}_3 \]

\[ \uparrow \]

\[ C_2\text{C} \cdot \text{OC} \leftrightarrow C_2\text{COC} \rightarrow C_2\text{C} + \text{CH}_3\text{O} \]

\[ \downarrow \]

\[ C_2\text{COC} + \text{H} \leftrightarrow C_2\text{COC} \rightarrow C=\text{COC} + \text{CH}_3 \]

\[ \downarrow \]

\[ C=\text{CC} + \text{CH}_3\text{O} \]
CH₃ and CH₃O abstract other species to form CH₄ and CH₃OH or recombine with O, H, and OH. CH₃ also reacts with O₂ to CH₃OO and CH₂OOH. Higher temperatures or oxygen rich conditions accelerate radical generation reactions and the formation of CO and CO₂.

7.2.2 QRRK Analysis

A QRRK analysis of the chemically activated system, using generic estimates or literature values for high pressure rate constants and species thermodynamic properties for the enthalpies of reaction, yields thermodynamically and kinetically plausible apparent rate constants. The input rate parameters used in these calculations and results from the calculation are summarized in Appendix C.

**Unimolecular Dissociation of C₃C⁻**

The potential energy diagram and input parameters are shown in Appendix C.

\[
\begin{align*}
\text{C}_3\text{C}^- & \xrightarrow{k_1} \; \text{C}_2\text{C} = \text{C} + \text{H} \quad \Delta H_{\text{rxn}} = 97 \text{ kcal/mol} \\
\text{C}_2\text{C}^- & \xrightarrow{k_2} \; \text{C}_2\text{C} = \text{C} + \text{H}^- \\
\end{align*}
\]

\[
\Delta H_{\text{rxn}} = 97 \text{ kcal/mol}
\]

There are two channels for C₃C⁻ decomposition breaking the C--H bond to form iso-butene and intramolecular isomerization (H atom shift) through a 3 member cyclic intermediate, to C₂C=C. Reaction (1) is about ten times faster than reaction (2) for pressures above 1.0 atm, and more than ten times faster for pressures below 1.0 atm. That is due to high energy barrier and tight transition state (low A factor) to isomerization, although reaction (2) goes to a lower final energy level.
Unimolecular Dissociation of C₃-COC

The potential energy diagram and input parameters are shown in Appendix C.

\[
C_3\text{COC} \xrightarrow{k_1} C_3\text{COC}^- \quad \Delta H_{\text{rxn}} = 97 \text{ kcal/mol}
\]

\[
C_3\text{COC} \xrightarrow{k_2} C=\text{C(C)}\text{OC} + \text{CH}_3 \quad \Delta H_{\text{rxn}} = 97 \text{ kcal/mol}
\]

\[
C_3\text{COC} \xrightarrow{k_3} C_3\text{C} = \text{C} + \text{CH}_3\text{O} \quad \Delta H_{\text{rxn}} = 97 \text{ kcal/mol}
\]

The rate constants of these reactions are below their high pressure limit at atmospheric pressure and 1000 K. Reaction (3) of C₃-COC dissociation to C=\text{C(C)}\text{OC} (iso-butene) and CH₃ is the dominant channel. Reaction (1) which undergoes an intramolecular isomerization, 5 member cyclic intermediate, to C₃-COC⁻, is about ten times slower than reaction (3) due to the high energy barrier, although it goes to a lower final energy level. Reaction (2) is also limited by a lower A factor than Reaction (3) even though it is slightly more thermodynamically favorable.

Unimolecular Dissociation of C₃-COC⁻

The potential energy diagram and input parameters are shown in Appendix C.

\[
C_3\text{COC}^- \xrightarrow{k_1} C_3\text{C}^- + \text{CH}_2\text{O} \quad \Delta H_{\text{rxn}} = 97 \text{ kcal/mol}
\]

\[
C_3\text{COC}^- \xrightarrow{k_2} C_3\text{COC} \quad \Delta H_{\text{rxn}} = 97 \text{ kcal/mol}
\]

The rate constants of these reactions are below their high pressure limit at atmospheric pressure and 1000 K. Channel (1) is dominant. Channel (2) which undergoes a intramolecular isomerization, 5 member cyclic intermediate, to C₃-COC, does not occur due to the high energy barrier.
Unimolecular Dissociation of C$_2$C-OC

The potential energy diagram and input parameters are shown in Appendix C.

\[
\begin{align*}
C_2C-OC & \xrightarrow{k_1} C_2C=O + CH_3 \quad \Delta H_{rn} = 97 \text{ kcal/mol} \\
 & \xrightarrow{k_2} C=C(C)OC + H \quad \Delta H_{rn} = 97 \text{ kcal/mol} \\
 & \xrightarrow{k_3} C_2COC \quad \Delta H_{rn} = 97 \text{ kcal/mol} \\
 & \xrightarrow{k_4} C_2^+COC \quad \Delta H_{rn} = 97 \text{ kcal/mol}
\end{align*}
\]

The rate constants of these reactions are below their high pressure limit at atmospheric pressure and 1000 K. Reaction (1) is dominant. Reaction (2), (3), and (4) are not important due to the higher energy barriers.

Unimolecular Dissociation of C$_2$COC·

The potential energy diagram and input parameters are shown in Appendix C.

\[
\begin{align*}
C_2COC· & \xrightarrow{k_1} C_2COC \\
 & \xrightarrow{k_2} C_2C· + CH_2O \\
 & \xrightarrow{k_3} C_2^+COC
\end{align*}
\]

The rate constants of these reactions are below their high pressure limit at atmospheric pressure and 1000 K. Reaction (2) is dominant. Reaction (1) and (3) which undergo the intramolecular isomerizations, 4 and 5 member cyclic intermediate respectively, to C$_2$C-OC· and C$_2$-COC, do not occur to a significant degree due to the high energy barriers, although they go to a lower final energy levels.

Unimolecular Dissociation of C$_2$-COC

The potential energy diagram and input parameters are shown in Appendix C.
\[
\begin{align*}
C_2\text{COC} & \rightarrow_{k_1} C_2\text{COC} . & \rightarrow & C_2\text{C} + \text{CH}_2\text{O} & \Delta H_{\text{rxn}} = 97 \text{ kcal/mol} \\
& \rightarrow_{k_2} C=\text{COC} + \text{CH}_3 & \Delta H_{\text{rxn}} = 97 \text{ kcal/mol} \\
& \rightarrow_{k_3} C=\text{C(C)OC} + \text{H} & \Delta H_{\text{rxn}} = 97 \text{ kcal/mol} \\
& \rightarrow_{k_4} C=\text{CC} + \text{CH}_3\text{O} & \Delta H_{\text{rxn}} = 97 \text{ kcal/mol} \\
& \rightarrow_{k_5} C_2\text{C-OC} & \Delta H_{\text{rxn}} = 97 \text{ kcal/mol}
\end{align*}
\]

The rate constants of these reactions are below their high pressure limit at atmospheric pressure and 1000 K. The rate constants for all channels are close to their high pressure limit when the pressure reaches 10 atm. Reaction (1) and (2) are dominant.

**7.2.3 Mechanism Validation**

A detailed reaction kinetic mechanism which is listed in Table A.4, was developed to describe the MTBE oxidation reaction system of reactions studied based on methanol oxidation reaction system studied in previous chapters. Elementary reaction rate parameters for abstraction reactions are based upon evaluated literature comparison, thermodynamics, Transition State Theory determination of Arrhenius A factor and energies of activation when literature data was not available. QRRK calculation, as described in previous section, were used to estimate apparent rate parameters for dissociation and combination reactions reported for a temperature range of 300 to 2500 K and at pressure of 0.001 to 50 atm for \(\text{N}_2\) as bath gas.
Model predictions and experimental data of Norton et al. are shown in Fig B.113 which shows reasonable agreement between calculated and experimental results for MTBE decay and methanol formation. An offset of 10 msec has been taken for the model result to compensate the reagent injection and mixing issues with their experimental setup. Model result of Iso-butene formation is consistent with experimental data in the initial stage, but over predicted for propagation which is probably due to the complexity of C4 oxidation. Even the mechanism of 448 reactions is not complete enough for the iso-butene oxidation. For the decomposition order of C4 → C3 → C2 → C1, C3 and C2 are therefore under predicted.
CHAPTER 8

CONCLUSIONS

This study presents experimental data on the decomposition of methanol in several different reaction environments - fuel lean to stoichiometric at a temperature range of 873 and 1073 K and a pressure range of 1 and 5 atm. Methane fuel is also added in several of the systems studied in order to provide experimental data to understand the methanol addition effect on the methane oxidation.

Computer codes: ThermCal, ThermSrt and ThermCvt have been developed for the thermal property calculations of stable molecules by the Benson group additivity method and of radicals by the NJIT hydrogen bond increment method.

Pressure dependent rate coefficients have been expressed using Chebyshev polynomials adopted for complex chemical activated reaction systems in this study, as well as unimolecular decomposition reactions. This method has also been tested and shows significant improvement over two convention methods, Troe’s and SRI. The Levenberg-Marquardt algorithm has been incorporated with the QRRK code, CHEMACT, for the fitting of Chebyshev polynomials.

A pressure dependent mechanism which consists 147 species and 448 elementary reactions, based on thermochemical kinetic principals has been developed and calibrated by the experimental data. The reaction mechanisms (models) include pathways for formation of higher molecular weight products, such as the formation of methyl ethers. This accurate model based on principles of thermochemical kinetics and
statistical mechanics will not only provide fundamental understanding, but can be used to suggest directions toward process optimization for experimental testing.

The CHEMKIN interpreter has been modified in this study to take the flexible matrix size of Chebyshev polynomials and to generate the appropriate link file for further processing. Subroutines involving kinetic rate coefficient calculation and array size initialization are also modified to incorporate the Chebyshev polynomials expression. Existing CHEMKIN drivers can simply link with the modified CHEMKIN library to take temperature and pressure dependent Chebyshev polynomials without modification. A couple of drivers have been linked with the new CHEMKIN library and tested. These include a shock tube driver, a constant temperature and pressure driver, and an adiabatic constant pressure driver.

The mechanism is validated with methanol oxidation and pyrolysis experimental data and serves as a basis to build upon during the subsequent efforts on higher molecular weight oxygenated hydrocarbon (MTBE in this study). The methanol addition shows dramatic acceleration effect on the methane oxidation experimentally and predicted by the model.
APPENDIX A

TABLES

<table>
<thead>
<tr>
<th>Compounds</th>
<th>Retention Time (min)</th>
<th>Relative Response Factor (RRF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO</td>
<td>0.9</td>
<td>0.976</td>
</tr>
<tr>
<td>CH₄</td>
<td>1.3</td>
<td>1.000</td>
</tr>
<tr>
<td>CO₂</td>
<td>3.2</td>
<td>0.899</td>
</tr>
<tr>
<td>CH₂=CH₂</td>
<td>4.3</td>
<td>1.759</td>
</tr>
<tr>
<td>CH₃OH</td>
<td>5.5</td>
<td>1.981</td>
</tr>
<tr>
<td>CH=CH</td>
<td>6.8</td>
<td>0.811</td>
</tr>
<tr>
<td>CH₂O</td>
<td>10.6</td>
<td>1.114</td>
</tr>
<tr>
<td>CH₂=CHCH₃</td>
<td>11.2</td>
<td>2.744</td>
</tr>
<tr>
<td>CH₃CH₂CH₃</td>
<td>11.5</td>
<td>2.960</td>
</tr>
<tr>
<td>CH=CH₃</td>
<td>12.8</td>
<td>1.070</td>
</tr>
<tr>
<td>CH₃OCH₃</td>
<td>13.0</td>
<td>1.668</td>
</tr>
<tr>
<td>CH₃OH</td>
<td>14.4</td>
<td>1.088</td>
</tr>
<tr>
<td>CH₃CH=O</td>
<td>14.9</td>
<td>---</td>
</tr>
<tr>
<td>CH₂=CHCH₂CH₃</td>
<td>15.6</td>
<td>3.458</td>
</tr>
<tr>
<td>CH₃CH₂CH₂CH₃</td>
<td>15.9</td>
<td>3.963</td>
</tr>
<tr>
<td>CH₂=CHCH₂CH₂CH₃</td>
<td>20.7</td>
<td>4.390</td>
</tr>
<tr>
<td>CH₃CH₂CH₂CH₂CH₃</td>
<td>21.2</td>
<td>4.901</td>
</tr>
<tr>
<td>CH₃OCH₂OCH₃</td>
<td>22.1</td>
<td>2.492</td>
</tr>
<tr>
<td>CH₂=CHCH₂CH₂CH₂CH₃</td>
<td>32.9</td>
<td>5.541</td>
</tr>
<tr>
<td>CH₃CH₂CH₂CH₂CH₂CH₃</td>
<td>33.8</td>
<td>5.748</td>
</tr>
</tbody>
</table>
Table A.2 Average Retention Times and Relative Response Factors for Column B, 90 m x 0.53 mm Hewlett Packard Fused Silica Capillary Column

<table>
<thead>
<tr>
<th>Compounds</th>
<th>Retention Time (min)</th>
<th>Relative Response Factor (RRF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH₄</td>
<td>3.50</td>
<td>1.000</td>
</tr>
<tr>
<td>CH₂=CH₂</td>
<td>3.58</td>
<td>1.618</td>
</tr>
<tr>
<td>CH₂=CH₂</td>
<td>3.58</td>
<td>1.735</td>
</tr>
<tr>
<td>CH₃CH₃</td>
<td>3.62</td>
<td>1.861</td>
</tr>
<tr>
<td>CH₄O</td>
<td>3.8</td>
<td>&lt;0.01</td>
</tr>
<tr>
<td>CH₂=CHCH₃</td>
<td>3.9</td>
<td>2.655</td>
</tr>
<tr>
<td>CH₃CH₂CH₃</td>
<td>4.1</td>
<td>2.835</td>
</tr>
<tr>
<td>CH₃OCH₃</td>
<td>4.2</td>
<td>1.067</td>
</tr>
<tr>
<td>CH₃OH</td>
<td>4.5</td>
<td>0.729</td>
</tr>
<tr>
<td>CH₂=CHCH₂CH₃</td>
<td>4.6</td>
<td>3.437</td>
</tr>
<tr>
<td>CH₃CH₂CH₂CH₃</td>
<td>4.7</td>
<td>3.842</td>
</tr>
<tr>
<td>CH₃CH=O</td>
<td>4.7</td>
<td>---</td>
</tr>
<tr>
<td>CH₂=CHCH₂CH₂CH₃</td>
<td>6.3</td>
<td>4.414</td>
</tr>
<tr>
<td>CH₃CH₂CH₂CH₂CH₃</td>
<td>6.4</td>
<td>4.726</td>
</tr>
<tr>
<td>CH₃OCH₂OCH₃</td>
<td>7.2</td>
<td>1.626</td>
</tr>
<tr>
<td>CH₂O Trimer</td>
<td>8.1</td>
<td>---</td>
</tr>
<tr>
<td>CH₂=CHCH₂CH₂CH₂CH₃</td>
<td>8.9</td>
<td>5.507</td>
</tr>
<tr>
<td>CH₃CH₂CH₂CH₂CH₂CH₃</td>
<td>9.1</td>
<td>5.551</td>
</tr>
</tbody>
</table>
o

o

o

o

O

o

©

©

©

o

o

©

o

O

O

o

©

o

©

o

«H o

0

0

0

0

0

0

0

0

O

©

©

©

o

O

o

O

©

©

rH rH rH rH

rH rH rH rH rH CM CM CM CM CM CM CM CM CM CM CM CM CM CM ©

©

©

©

rH rH CM o

o

o

rH rH o

rH CM CM CM CM CM CM rH o

©

o

o

©

o

o

©

o

©

C5 CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD 0
O o © © o o © © o O © © © o © o © o o © O O © ©

o

©

©

o

©

o

rH CM ©

©

rH CM rH rH rH H

0 0 0 0 0

1 5 0 t o 1 5 CD w

«—t o

rH

%

o

o

o

o

o

o

o

o

o

o

o

o

H

CM o

©

CM «H CM *H H

©

©

rH rH rH CM CM ©

©

©

o o

O

o

DATE

ELEMENTS

o o o

z

o
CD
00
rH
o
CM
H

fT}
cn
-v ,
CM

1500
1000
800
600

CD

m

p

©

rH o
03 o

xt

p

500

O

I

z

I

z

z

Z

p

CD p
CD ©
CM c n
rH ©
CM CM
*H

CD
ao
cn
rH
CM
rH

o
p

© O ©
c n CD

c o XX p

©
©
c n rH
o ©
CM CM
rH

CD

u

rH
©
N.
©

o

rH
©
P.
©
*n

u

©
©
CM
rH
CM
rH

u

©
©
CM
rH
CM
rH

o o
03 m

CM p
rH

O
©

©
©

rH © ©
»H rH

p © O
0 3 "X ©
xr ©

o
p
©
•v .
CM
rH
©

u

©
©
CM
rH
CM
rH

o
o

o
o

XX p

m

CD

rH p
rH

o o o
c n rH p
p

O in ©
rH rH

o ©
cn m

o p
•H

o ©
cn p

cn xr
rH

©

©

cn n

©

©

o o

O

O

o

O

rH rH rH CM ©

o o o

O

CJ u

XT r - m

o*

cn o
03 o

o o O
XT 0 3 ©
p

XT

o o
CM c n

p

© o © o
XT »H CD ©
03

© ©
© cn

p

©

o ©
rH x r

O

o o o o

O
o

o o o © © © ©
c n rH © rH CM XT O

©

©

©

©

p

XT p

CD

p

©

©

rH
rH

O O
rH ©

rH CM «<x p

©
H

©

©

©
©

rH ©
CM ©
rH
- v P* o
•H
©
rH

©

©

z

©
rH
O
o

z

CM r ~ r * © © O
CM XX XX CM CM O

©
o

O
O

o
O

if)

©

CM CM CM P - r~
rH rH rH rH rH

©

O

P
p

rH rH rH © ©
© © © r- ©

o
©

o
o

© © rH rH rH o
rH *H © © © o

O
rH

©

©

©
o

© © o
CM CM ©

CM ©
XX ©

p

©

rH XT ID ID r~
rH «H rH rH rH

0 3 CM © P P CD ©
rH rH rH rH

CM CM rH ©
rH
r* ©

O

p © © © © © © ©
p © O rH CD CD
© © P

©

©

©

© o p
© © © ©

CM XT ©

P* XX
p p
rH rH CO © © ©

©

p

©

©

©

©

p

©

©

p-

CM O
p ©

o
©

CM rH P
xx p ©

CD

©
o

©
o

©
©

©
m

©
©

CM ©
if) ©

©
©

C J CJ u
,—1
ro
CJ
£

p
H

o
o

O
o

© p
XT

P P ©
rH rH ©

© ©
o« ©
V
CJ © © ©
r-H H rH rH
N.
-•v "•s
o o o o
rH rH
rH
©
©

©
©

o © © © ©
CM CM XT CO

© O
CM o

©
O

rH rH ©
© © ©

O
©

CM CM CM ID CM ©
© c n CM © ©

o
©

rH rH o
CM x r p

©

CD

OO CD o j
rH
rH

P
©

©
©

P
©

©
©

P ©
t—t

P

P ©
rH x r

CM © rH rH x f ©
© © rH © CM CM O o
rH rH
rH rH rH rH CM CM CM CM CM CM rH rH

rH © Ov © rH ©
CM CM © x r
rH
rH rH CM CM rH rH rH CM rH

©

xr ©

©

©
CO

rH o

CD rH CD rH
© © © ©

CM CM © x r CM CM CM ©
CM CM CM CM CM pH rH

if)

rH rH
ro ro
CJ U
£ E

JZ JZ JZ
Eh H E -

© © rH o
rH rH CM ©

XT

P
©
rH

rH rH © rH ©
X© XT
© X

CD

©

O

CM x r
r-1 rH

o
o

©

xr P
O ©

© xf

*
■
*

xX CM P - P - P»
CM © P* P* ©
©

©

p* © P*
© © ©

rH rH ©
rH

.H

© XT x r P
© O
© c n © © CO ©

CM © © XX ©
rH rH rH rH

©

©

© o ©
rH rH

O

CM o © © © P © © © o
rH CM rH rH rH r t rH rH r' H

© o XT xx CM CM CM XX © © © © © P* © © ©
CM CM © © © CM CM xX CM © © P* P* xx © © CM
©

O
O

©

©
©
V
CM CM CM
— \
rH rH
o o
M
rH rH © © ©
rH rH rH

rH rH rH © CM © rH P P P
CM CM © © xx © x r CM CM CM ©

© © ©

O © x r XX ID ©
rH rH
rH rH

©

©

©
©

U

rH
© ro
© U
N. £
© jz

© © © x r xx © © XX XX x r
O © rH O rH rH © i n
CM CM CM rH rH rH rH rH rH CM CM CM CM CM CM CM CM rH rH

CM ©
rH p

©

© © P
rH XT P

©
©

p~
CM

rH o
XX p

p

C J CJ u

©

J

0 3 P © © o © rH P
rH CM CM XX CM CM © P

© © © P P © © ©
rH rH rH rH rH rH rH rH

©

©

©

©

o

P
©

© J ©
rH rH

rH P
© P

rH ©

rH
rH

Cp

XT ID m

p

o
O

p ID o
cn 03 o

CD

o o
CM ©

xt p

© © p
© ©

xx p

©
o

©
©
—
CM
rH

CM CM
rH rH

o
©

©
©

o rH © P rH rH r - © i n © XX XT P* O P*
p CM *** © rH r H «D rH XT XT © © © xx i f ) ©
© o ’ © XX P © © o © o o XX XX © © XT XT
rH rH CM CM
rH rH rH rH rH CM rH rH rH rH
rH

p

p

©
©

if)

©
M
CM
rH
P.
o
«H

XX ©
© ©

u

rH •H
ro ro ro ©
CJ ( J U ©
£ £ £ ■V.
JZ JZ J Z ©
H H E-r
©

E-*
©
© © ©
N ©
© © M
CM
M \
rH
o O \
rH rH O
rH
if)

©
©
N.
CM

CJ u

JZ J Z

©

CM p
rH

CM ©
rH

ID m

xt

r* ©

O
©

©
o

O
©

CM
P*
•Xs.
CM

CJ CJ u

rH
ro
CJ
£

o
©

XT

©

O
O

CJ u
rH
ro
CJ
£

xT p

cn

p

©
03
M
CM
rH
p. N
o o
rH rH

C J CJ CJ u

P o
03 ©

© o o © CM p
0 3 CM 0 3 rH © ©

p

JZ JZ JZ
fr> ( -

H

©
© ©
© M
CM
CM rH
rH \
M P
©
rH

rH
ro
u
£
.C
£-

u

rH rH © ©
rH rH rH rH

p

in

CJ u

rH rH rH
ro ro ro
CJ < J U
£ £ £

rH
© ro
© CJ
CM E
rH j z
CM fr>
rH

©
©

u

© p
rH

o o ©
CD 0 3 o

p

p
p

CJ u

XT p

r- o o
© CM o

r o XX
rH

XT O
p ©

© ©
rH

o o © o o © © © p © O
c n O l © c n «T H CM H 0 3 © o

i n o o XT p

r- o
© m

p
4 00

©

o o z z Z z z z Z z z Z Z z z z Z z z z Z Z Z Z z z Z z z z z
o O
rH CM CM CM rH rH rH rH rH rH rH rH rH rH rH rH rH rH r H rH rH rH rH rH rH rH rH rH rH rH rH rH rH rH rH rH

•v*
r-

p 03 o
cn O o

300

o

o o

p
rH o
cn © O

S

CD p o o
© p XT o

o © o
<n 0 3 p

o O O
CM rH ©

©
©

H
©

CD i n 0 0 c n p c n XT rH m i n rH ©
© XT © TT CM XT m c n x r © CM ©

Hf

o
O

o
o

o
m
03
in

o
O

o
f—1

o o
cn m

CM 0 3 CM
in

o
o

O
©

©
©

p

CM
©
1

m
I

O
©

©
©

o O
CM O
p

rH ©

xx ©

XX

o O
XT o

©
o

CD 0 3 0 0 CM if) CM i f ) © CM © CM XT CM ©
© ©
O © © © rH © © rH © CM ©

© ©
in ©

©
©

©
©

©
©

xx xx xX © XT ©* ©
r * P P* xr XX x r ©

o

o

o
©

©
©

©
©

o

xx XX ©

©
CM o o
p CM © XX rH H
rH 1 1 rH
o

o
©

©
©

©

if)

©

©

xr ©

©
©

©
©

o
©

CM ©
CM ©

CM © ©
P xr

© ©
in m

XX ©
© P

c n xx P
p P

©
P

©

p

©
m

o ©
CM ©

o xr o
XT © o

O

O

©
©

p

xx xX o

©
©

CD

©
CD o

©
©

©
©

©
©

CM ©
rH ©

O
©

© o
03 ©

© 03 © rH rH © CM rH CM CD CM © O XT P rH © P P rH xx ©
rH
rH © rH © © ©
rH © xx © © © © O © xx CM CM ©
1 1
1
rH
1 1
1
1 1 1

CM

5

CM
z O

CM
O z

o

_

CM
O o ©
r e o CM CM CM
o z z Z z U

CM
o

u u

o
U

z

u

o
o
z

+

O
z
u

CM

O
o

o
+
©

o cj o o
c j CJ
(J u
z z U z z

Eh
O
*
X
w
O ' CM CM CM o
Z Z Z U
c j ( J CJ U z

H
z
o
o
o o
o
• z z O
z
>
CM
CJ
z o
cj
cj
O CM o z z u u < y z
• o o
C J Z CJ
z u I o o z z z o

xX ©
O

CD

03
©

©

© ©
1

©
1

Z

o
© ©
z n:
CJ U

O o
© ©
n: z

z
o
©
z

H

SPECIES

Table A.3 Thermodynamic Properties

O

©

Z

Z

O
O

O
O

o

u

o
o
z
O

©

z

u

o Z

CJ CJ C J


### Table A.4 Detailed Mechanism for Methanol and MTBE Oxidation

**ELEMENTS**

<table>
<thead>
<tr>
<th>CH4</th>
<th>CH3OH</th>
<th>CO</th>
<th>CO2</th>
<th>CH2O</th>
</tr>
</thead>
</table>

**SPECIES**

<table>
<thead>
<tr>
<th>C2H4</th>
<th>C2H6</th>
<th>COC</th>
<th>COCCO</th>
<th>C2H2</th>
</tr>
</thead>
</table>

**H2O**

<table>
<thead>
<tr>
<th>H2</th>
<th>H2O2</th>
</tr>
</thead>
</table>

**AR**

<table>
<thead>
<tr>
<th>N2</th>
<th>O2</th>
</tr>
</thead>
</table>

**CH**

<table>
<thead>
<tr>
<th>CH2OH</th>
<th>CH3O</th>
<th>CH3</th>
<th>CH2</th>
<th>CH2E</th>
</tr>
</thead>
</table>

**CO**

<table>
<thead>
<tr>
<th>COCOC</th>
<th>COCOH</th>
<th>COCOO</th>
<th>C.CC</th>
<th>C.CCH</th>
</tr>
</thead>
</table>

**HCO**

<table>
<thead>
<tr>
<th>HOC</th>
<th>HOCO</th>
<th>HCOO</th>
<th>HCOH</th>
<th>HCO3</th>
</tr>
</thead>
</table>

**HCCO**

<table>
<thead>
<tr>
<th>CCOC</th>
<th>CCCO</th>
<th>C.CCO</th>
<th>C.CCOH</th>
<th>C.CCOC</th>
</tr>
</thead>
</table>

**HCO2**

<table>
<thead>
<tr>
<th>CO2</th>
<th>CO2O</th>
<th>CO2C</th>
<th>CO2C*</th>
<th>CO2C*O</th>
</tr>
</thead>
</table>

**HCO3**

<table>
<thead>
<tr>
<th>CO3</th>
<th>CO3O</th>
<th>CO3C</th>
<th>CO3C*</th>
<th>CO3C*O</th>
</tr>
</thead>
</table>

**HCOH**

<table>
<thead>
<tr>
<th>COH</th>
<th>COCOH</th>
<th>COHOH</th>
<th>C.COH</th>
<th>C.CCOH</th>
</tr>
</thead>
</table>

**HCCO2**

<table>
<thead>
<tr>
<th>CO2C</th>
<th>CO2CO</th>
<th>CO2COH</th>
<th>C.COC</th>
<th>C.CCOC</th>
</tr>
</thead>
</table>

**HCCOC**

<table>
<thead>
<tr>
<th>COOC</th>
<th>COOCO</th>
<th>COOCOH</th>
<th>C.COC</th>
<th>C.CCOC</th>
</tr>
</thead>
</table>

**END**

**REACTIONS**

<table>
<thead>
<tr>
<th>O + O + M =&gt; O2 + M</th>
</tr>
</thead>
</table>

**AR/1.0/**

<table>
<thead>
<tr>
<th>O + H + M =&gt; OH + M</th>
</tr>
</thead>
</table>

**AR/1.0/ H/2/3.41/ N/2/2.29/ H2O/2.53/**

<table>
<thead>
<tr>
<th>H + H + M =&gt; H2 + M</th>
</tr>
</thead>
</table>

**AR/1.0/ N/2/1.53/**

<table>
<thead>
<tr>
<th>H2 + O =&gt; H + OH</th>
</tr>
</thead>
</table>

**OH + H + M => H2O + M |
|-----------------------|

**AR/1.0/ H/2/2.65/ H2O/16.96/**

<table>
<thead>
<tr>
<th>OH + OH =&gt; H2O + O</th>
</tr>
</thead>
</table>

**H2 + H => H2O + H2O2**

<table>
<thead>
<tr>
<th>H2O2 + O =&gt; OH + O2</th>
</tr>
</thead>
</table>

**H2O2 + H => H2O + H2O2**

<table>
<thead>
<tr>
<th>H2O2 + O =&gt; OH + H2O2</th>
</tr>
</thead>
</table>

**H2O2 + OH => H2O + H2O2**

<table>
<thead>
<tr>
<th>C + O2 =&gt; O + CO</th>
</tr>
</thead>
</table>

**C + OH => H + CO |
|------------------|

**C + CH2 => H + C2H |
|---------------------|

**C + CH3 => H + C2H2 |
|----------------------|

**CO + O + M => CO2 + M**

| H2/2.0/ O2/6.0/ H2O/6.0/ CH4/2.0/ CO/1.5/ CO2/3.5/ C2H6/3.0/ AR/0.5/ |

**CO + OH => CO2 + H**

<table>
<thead>
<tr>
<th>CO2 + O =&gt; CO2 + OH</th>
</tr>
</thead>
</table>

**CO + H2O => CO2 + OH**

<table>
<thead>
<tr>
<th>CH + O =&gt; H + CO</th>
</tr>
</thead>
</table>

**CH + O2 => CO + OH |
|---------------------|

**CH + O2 => HCO + O**

<table>
<thead>
<tr>
<th>H2O2 + O =&gt; H2O + O2</th>
</tr>
</thead>
</table>

**H2O2 + H => H2O + H2O2**

<table>
<thead>
<tr>
<th>H2O2 + O =&gt; OH + H2O2</th>
</tr>
</thead>
</table>

**H2O2 + OH => H2O + H2O2**

<table>
<thead>
<tr>
<th>C + O2 =&gt; O + CO</th>
</tr>
</thead>
</table>

**C + OH => H + CO |
|------------------|

**C + CH2 => H + C2H |
|---------------------|

**C + CH3 => H + C2H2 |
|----------------------|

**CO + O + M => CO2 + M**

| H2/2.0/ O2/6.0/ H2O/6.0/ CH4/2.0/ CO/1.5/ CO2/3.5/ C2H6/3.0/ AR/0.5/ |

**CO + OH => CO2 + H**

<table>
<thead>
<tr>
<th>CO2 + O =&gt; CO2 + OH</th>
</tr>
</thead>
</table>

**CO + H2O => CO2 + OH**

<table>
<thead>
<tr>
<th>CH + O =&gt; H + CO</th>
</tr>
</thead>
</table>

**CH + O2 => CO + OH |
|---------------------|

**CH + O2 => HCO + O**

<table>
<thead>
<tr>
<th>H2O2 + O =&gt; H2O + O2</th>
</tr>
</thead>
</table>

**H2O2 + H => H2O + H2O2**

<table>
<thead>
<tr>
<th>H2O2 + O =&gt; OH + H2O2</th>
</tr>
</thead>
</table>

**H2O2 + OH => H2O + H2O2**

<table>
<thead>
<tr>
<th>C + O2 =&gt; O + CO</th>
</tr>
</thead>
</table>

**C + OH => H + CO |
|------------------|

**C + CH2 => H + C2H |
|---------------------|

**C + CH3 => H + C2H2 |
|----------------------|

**CO + O + M => CO2 + M**
CHEB/ 7 3  5.946E+00  2.1452E+00 -1.9621E-01  2.1867E+00/
CHEB/ 3.8262E-01  1.9660E-01 -3.9233E-01 -8.054E-03  1.1095E-02/
CHEB/ -6.0977E-02 -2.7320E-02 -1.5235E-02  9.7130E-03 -5.4188E-03/
CHEB/ -4.9043E-03  3.2732E-03 -2.4838E-03  8.2242E-04  8.4029E-04/
CHEB/ 1.4555E-03  9.8956E-04/

C*O (+M) = CO2 + OH (+M) 1.00E+00 .000 0. !ING341 10/95

LOW / 1.0 0.0 0.0 /
CHEB/ 7 3  4.8872E+00  2.1452E+00 -1.9621E-01  2.1867E+00/
CHEB/ 3.8262E-01  1.9660E-01 -3.9233E-01 -8.054E-03  1.1095E-02/
CHEB/ -6.0977E-02 -2.7320E-02 -1.5235E-02  9.7130E-03 -5.4188E-03/
CHEB/ -4.9043E-03  3.2732E-03 -2.4838E-03  8.2242E-04  8.4029E-04/
CHEB/ 1.4555E-03  9.8937E-04/

CH2 + O = H2 + CO 1.50E+13 .000 0.

CH2S + O = H + HCO 1.50E+13 .000 0.

CH2S + H = CH + H2 3.00E+13 .000 0.

CH2S + O2 = CO + H2O 1.20E+13 .000 0.

CH2S + OH = CH2O + H 3.00E+13 .000 0.

CH2S + H2 = CH3 + H 7.23E+13 .000 0.

CH2S + H2O = CH2 + H2O 3.00E+13 .000 0.

CH2S + CH3 = H + C2H4 1.20E+13 .000 -570.

CH2S + CO = CH2 + CO2 9.00E+12 .000 0.

CH2S + CO2 = CH2 + CO2 7.00E+12 .000 0.

CH2S + CO2 = CO + CH2O 1.40E+13 .000 0.

CH2S + M = CH2 + M 1.00E+13 .000 0. !MILLER

CH2S + CH4 = 2CH3 4.00E+13 .000 0. !MILLER

CH2S + CH5 = CH3 + C2H5 1.20E+13 .000 0. !MILLER

CH2S + O2 = CH + OH + H 3.00E+13 .000 0. !MILLER

CH2S + H = CH2 + H 2.00E+14 .000 0. !MILLEK

CH2 + H (+M) = CH3 (+M) 2.50E+16 -8.0 0.

LOW / 3.200E+27 -3.140 1230.00/

TROE/ .6800 78.00 1995.00 5590.00 /

H2/2.00/ H2O/6.00/ CH4/2.00/ CO/1.50/ CO2/2.00/ C2H6/3.00/ AR/ .70/}

CH2 + H2 = H + CH3 5.00E+05 2.00 7230.

CH2 + CH2 = H2 + C2H2 3.200E+13 .000 0.

CH2 + CH3 = H + C2H4 4.00E+13 .000 0.

CH2 + CH4 = 2CH3 2.460E+06 2.00 8270.

CH2 + CO (+M) = C*O (+M) 8.100E+11 .50 4510.

LOW / 2.690E+33 -5.110 7095.00/

TROE/ .5907 275.00 1226.00 5185.00 /

H2/2.00/ H2O/6.00/ CH4/2.00/ CO/1.50/ CO2/2.00/ C2H6/3.00/ AR/ .70/}

CH2 + HCCO = C2H3 + CO 3.00E+13 .000 0.

CH2 + O = CO + H + H 5.00E+13 .000 0. !MILLER

CH2 + O2 = CO2 + H + H 1.60E+12 0.00 1000. !MILLER

CH2 + O2 = CH2O + O 5.00E+13 .000 9011. !MILLER

CH2 + O2 = CO2 + H2 6.90E+11 0.00 502. !MILLER

CH2 + O2 = CO + OH + H 8.60E+10 0.00 -502. !MILLER

CH2 + O2 = HCO + H 4.30E+10 0.00 -502. !MILLER

CH2 + H = CH + H2 4.00E+13 0.0 0. !MILLER

CH2 + OH = CH + H2O 1.13E+07 2.0 3011. !MILLER

CH2 + OH = CH2O + H 2.50E+13 0.0 0. !MILLER

HCO (+M) = CH2O (+M) 1.00E+00 .000 0. !ING11) 10/95

LOW / 1.0 0.0 0.0 /

CHEB/ 7 3  6.1562E+00  2.0857E+00 -2.0733E-01  1.8817E+00/

CHEB/ 4.3977E-01  2.0117E-01 -3.5772E-01 -7.3873E-03  1.4671E-02/

CHEB/ -5.0220E-02 -2.7614E-02 -1.3531E-02  8.7047E-03 -6.1176E-03/

CHEB/ -4.9343E-03  6.9705E-03  1.3861E-03  1.6764E-04  9.3469E-04/

CHEB/ 1.1733E-03  6.7250E-04/

HCOH + O2 (+M) = HOCHHO.T (+M) 1.00E+00 .000 0. !ING12 10/95

LOW / 1.0 0.0 0.0 /

CHEB/ 7 3  9.8754E+00  2.4000E+00 -6.9199E-02 -1.3323E+00/

CHEB/ 1.2612E-01  8.5682E-02 -3.9071E-01 -2.0950E-02 -1.2240E-02/

CHEB/ -7.1692E-02 -8.4939E-03 -6.5982E-03  5.6278E-03  4.6319E-04/

CHEB/ 9.7938E-05  7.1146E-03  1.3651E-03  1.0155E-03 -1.4034E-03/

CHEB/ 4.1476E-04  3.4317E-04/

HCOH + O2 (+M) = HOCH2 + O (+M) 1.00E+00 .000 0. !ING12 10/95

LOW / 1.0 0.0 0.0 /

CHEB/ 7 3  1.2574E+01 -9.4583E-02 -6.5418E-02 -1.3958E-01/

CHEB/ 1.2250E-01  8.3300E-02 -1.2444E-01 -2.2717E-02 -1.3643E-02/

CHEB/ -5.6074E-02 -7.9408E-03 -6.2684E-03 -2.0959E-02  9.0130E-04/

CHEB/ 4.3687E-04 -7.1897E-03  1.3556E-03  1.0246E-03 -2.3960E-03/

CHEB/ 3.0783E-04  2.6365E-04/

HCOH + O2 (+M) = O.H.OCONT (+M) 1.00E+00 .000 0. !ING12 10/95

LOW / 1.0 0.0 0.0 /

\[ CH_3 + OH \rightarrow CH_3O + H_2 \quad 1.00E+00 \quad 0.00 \quad 0 \quad ! \quad ING101 \quad 10/95 \]

\[ CH_3 + OH \rightarrow CH_2O + H_2 \quad 1.00E+00 \quad 0.00 \quad 0 \quad ! \quad ING101 \quad 10/95 \]

\[ CH_3O + CH_3 \rightarrow CH_2O + CH_4 \quad 2.41E+03 \quad 0.00 \quad 0 \quad ! \quad NORTON \quad 10/95 \]

\[ CH_3O + CH_2O \rightarrow CH_3OH + HCO \quad 1.02E+00 \quad 0.00 \quad 0 \quad ! \quad NORTON \quad 10/95 \]

\[ CH_3O + O_2 \rightarrow CH_2O + H_2O \quad 3.61E+10 \quad 0.00 \quad 0 \quad ! \quad NORTON \quad 10/95 \]

\[ CH_3O + O \rightarrow CH_2O + OH \quad 6.02E+12 \quad 0.00 \quad 0 \quad ! \quad NORTON \quad 10/95 \]

\[ CH_3O + H \rightarrow CH_2O + H_2 \quad 1.99E+13 \quad 0.00 \quad 0 \quad ! \quad NORTON \quad 10/95 \]

\[ CH_3O + H_2 \rightarrow CH_2O + H_2O \quad 3.01E+11 \quad 0.00 \quad 0 \quad ! \quad NORTON \quad 10/95 \]

\[ CH_3O + CO \rightarrow CH_3 + CO2 \quad 1.57E+13 \quad 0.00 \quad 0 \quad ! \quad NORTON \quad 10/95 \]

\[ CH_3O + CH_2O \rightarrow CH_3OH + HCO \quad 1.02E+11 \quad 0.00 \quad 0 \quad ! \quad TSANG \quad 10/95 \]

\[ CH_3O + CH_3 \rightarrow CH_2O + CH_4 \quad 2.41E+13 \quad 0.00 \quad 0 \quad ! \quad TSANG \quad 10/95 \]

\[ CH_3O + CH_3 + \rightarrow COC + (H) \quad 1.00E+00 \quad 0.00 \quad 0 \quad ! \quad ING131 \quad 10/95 \]

\[ CH_3O + CH_3O \rightarrow CH_2O + CH_3OH \quad 1.00E+00 \quad 0.00 \quad 0 \quad ! \quad ING101 \quad 10/95 \]
\[
\begin{align*}
\text{CH}_30H \quad (+M) \quad &\rightleftharpoons \quad \text{CH}_20H \quad + \quad H \quad (+M) \\
\text{CH}_30H \quad (+M) \quad &\rightleftharpoons \quad \text{CH}_20H \quad + \quad H_2 \quad (+M) \\
\text{CH}_30H \quad (+M) \quad &\rightleftharpoons \quad HCOH \quad + \quad H_2O \quad (+M) \\
\text{CH}_30H \quad (+M) \quad &\rightleftharpoons \quad \text{CH}_2S \quad + \quad H_2O \quad (+M) \\
\text{CH}_30H \quad (+M) \quad &\rightleftharpoons \quad \text{CH}_3 \quad + \quad OH \quad (+M) \\
\text{CH}_4 \quad + \quad H_2O \quad &\rightleftharpoons \quad \text{CH}_3 \quad + \quad H_2O_2 \\
\text{CH}_30H \quad (+M) \quad &\rightleftharpoons \quad \text{CH}_20H \quad + \quad OH \quad (+M) \\
\text{CH}_30H \quad (+M) \quad &\rightleftharpoons \quad \text{CH}_2OH \quad + \quad OH \quad (+M) \\
\end{align*}
\]
<table>
<thead>
<tr>
<th>Reaction</th>
<th>Rate Constant</th>
<th>Temperature</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_2H_5 + O )</td>
<td>( 1.00E+00 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
COCOCH + \text{OH} \leftrightarrow \text{COCO} + \text{H}_2\text{O} \quad 6.46 \times 10^{-14} \quad 0.00 \quad 43000.

COCO + \text{H}_2\text{O} \leftrightarrow \text{COCO} + \text{H} \quad 3.00 \times 10^{-12} \quad 0.00 \quad 39000.

\begin{align*}
\text{COCO} \quad \text{(H)} & \leftrightarrow \text{COCO} + \text{H} \quad 1.00 \times 10^{-0} \quad 0.00 \quad 0.1 \quad \text{ING151} \quad 10/95 \\
\text{LOW} / 1.0 \quad 0.0 & / 0.0 \\
\text{CHEB} / 7 \quad 3 & -2.687 \times 10^{-0} \quad 5.879 \times 10^{-0} \quad -1.364 \times 10^{-0} \quad 3.129 \times 10^{-0} / 0.1 \quad \text{ING151} \quad 10/95 \\
\text{CHEB} / 8 \quad 7.664 \times 10^{-1} & -1.179 \times 10^{-0} \quad 6.663 \times 10^{-0} \quad 4.984 \times 10^{-0} \quad 8.584 \times 10^{-0} / 0.3 \quad \text{ING151} \quad 10/95 \\
\text{CHEB} / -3.859 \times 10^{-0} & 2.186 \times 10^{-0} \quad 5.438 \times 10^{-0} \quad -2.076 \times 10^{-0} \quad 6.783 \times 10^{-0} / 0.1 \quad \text{ING151} \quad 10/95 \\
\text{CHEB} / -4.598 \times 10^{-0} & -1.043 \times 10^{-0} \quad 4.418 \times 10^{-0} \quad 2.345 \times 10^{-0} \quad -4.896 \times 10^{-0} / 0.2 \quad \text{ING151} \quad 10/95 \\
\text{CHEB} / -1.433 \times 10^{-0} & 6.635 \times 10^{-0} / 0.3 \\
\text{COCO} \quad \text{(H)} & \leftrightarrow \text{COCO} + \text{H} \quad 1.00 \times 10^{-0} \quad 0.00 \quad 0.1 \quad \text{ING151} \quad 10/95 \\
\text{LOW} / 1.0 \quad 0.0 & / 0.0 \\
\text{CHEB} / 7 \quad 3 & -2.259 \times 10^{-0} \quad 3.608 \times 10^{-0} \quad -6.499 \times 10^{-0} \quad 2.837 \times 10^{-0} / 0.1 \quad \text{ING151} \quad 10/95 \\
\text{CHEB} / 6.416 \times 10^{-1} & -1.004 \times 10^{-0} \quad -5.411 \times 10^{-0} \quad 4.566 \times 10^{-0} \quad -4.052 \times 10^{-0} / 0.2 \quad \text{ING151} \quad 10/95 \\
\text{CHEB} / -3.748 \times 10^{-0} & 2.599 \times 10^{-0} \quad 7.736 \times 10^{-0} \quad -2.050 \times 10^{-0} \quad 1.151 \times 10^{-0} / 0.1 \quad \text{ING151} \quad 10/95 \\
\text{CHEB} / 2.652 \times 10^{-0} & -1.097 \times 10^{-0} \quad 3.394 \times 10^{-0} \quad 2.318 \times 10^{-0} \quad -5.487 \times 10^{-0} / 0.2 \\
\text{CHEB} / -5.795 \times 10^{-0} & 1.234 \times 10^{-0} / 0.3 \\
\text{C} \quad \text{C} + \text{H} & \leftrightarrow \text{C} \quad 5.00 \times 10^{-0} \quad 0.00 \quad 5000.
\end{align*}
C2CC. (+ M ) \iff \ C2C*C + H \ (+ M ) 1.00E+00 .000 0. ! 11/95

C3C. (+ M ) \iff \ C*CC + CH3 (+ M ) 1.00E+00 .000 0. ! 11/95

C3C. (+ M ) \iff \ C2CC. (+M ) 1.00E+00 .000 . ! 11/95

C2CC. + O H \iff \ C2C*C + H2O 2.41E+13 0.0 8285.

C2CC. + CH30 \iff \ C2C*C + CH30H 2.00E+13 0.0 8000.

C2C*C + C H 30H \iff \ C2C*C + C H 30H 2.00E+13 0.0 8000.

C2C*C + CH20H \iff \ C2C*C + CH20H 2.00E+13 0.0 8000.

C2C*C + CH30H \iff \ C2C*C + CH30H 2.00E+13 0.0 8000.

C2C + CH30 \iff \ C2C + CH30 2.00E+13 0.0 8000.

C3C. (+ M ) \iff \ C2C + CH3 + (+ M ) 1.00E+00 .000 0. ! 9 11/95

C3C. (+ M ) \iff \ C2C + C C + (+ M ) 2.40E+12 0.0 0.

C2C + C C + (+ M ) \iff \ C2C + (+ M ) 2.40E+12 0.0 0.

C2C + C C + (+ M ) \iff \ C2C + (+ M ) 2.40E+12 0.0 0.

C2C + C C + (+ M ) \iff \ C2C + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.

C2C + H CH3 + (+ M ) \iff \ C2C + H CH3 + (+ M ) 2.40E+12 0.0 0.
CHEB/ -2.5273E-01 -4.9276E-02 9.9577E-02 -1.4479E-01 -1.7943E-01
CHEB/ 6.0428E-03 -2.9015E-02 9.7163E-02 -4.6495E-02 1.4702E-02
CHEB/ -6.1343E-03 3.4947E-02
C3C. + CH3O (+M) <= C3COC. + H (+M) 1.00E+00 .000 0 ! 0 11/95
LON/ 1.0 0.0 0.0 /
CHEB/ 7 3 1.8420E+00 -1.6478E+00 -7.9528E-02 7.2695E+00
CHEB/ 1.3293E+00 6.2204E-02 2.4576E-01 5.7127E-01 8.0170E-02
CHEB/ -3.0517E-01 2.0511E-03 1.0106E-01 -1.7282E-01 -1.7700E-01
CHEB/ 2.5238E-02 -3.7867E-02 -1.1786E-01 -3.6253E-02 1.4978E-02
CHEB/ -2.1126E-02 -3.9070E-02
C2COC. (+M) <= CC.C + CH2O (+M) 1.00E+00 .000 0 ! 4 11/95
LON/ 1.0 0.0 0.0 /
CHEB/ 7 3 5.3069E+00 1.4971E+00 -2.0475E-01 3.3677E+00
CHEB/ 1.1338E+00 1.6016E-01 -7.7226E-01 5.7127E-02 9.3844E-02
CHEB/ -6.3355E-02 -1.4002E-01 -2.1929E-02 8.6012E-02 -5.6832E-02
CHEB/ -3.1740E-02 4.1528E-02 9.0699E-03 -6.0416E-03 -1.0724E-02
CHEB/ 1.7693E-02 5.9315E-03
C2COC. (+M) <= C2CO C. (+M) 1.00E+00 .000 0 ! 4 11/95
LON/ 1.0 0.0 0.0 /
CHEB/ 7 3 2.5298E-01 3.6211E+00 -1.0178E-01 1.8813E+00
CHEB/ 1.2395E+00 9.9399E-02 -1.0650E+00 2.5313E-01 3.1540E-02
CHEB/ -9.7818E-02 -6.2776E-02 -2.6742E-02 8.8201E-02 -7.6076E-02
CHEB/ -2.0871E-02 3.5573E-02 -3.6810E-02 4.2775E-03 -1.8468E-02
CHEB/ -1.2472E-02 -1.2429E-02
C2C.OC (+M) ↔ CC.C + CH2O (+M) 1.00E+00 .000 0 ! 6 11/95
LOW / 1.0 0.0 0.0 / CHEB/ 7 3 -1.090E01 +3.651E00 +2.334E01 .124E01/ CHEB/ 1.229E00 0.446E02 -0.999E01 2.758E00 1.344E01/ CHEB/ -1.221E01 6.901E02 4.662E02 3.903E02 -9.724E02/ CHEB/ -1.974E02 7.223E03 -3.637E02 -2.671E02 -2.425E02/

C2C.OC (+M) ↔ C*COC + CH3 (+M) 1.00E+00 .000 0 ! 6 11/95
LOW / 1.0 0.0 0.0 /
CHEB/ 7 3 -1.066E01 1.325E00 -2.319E01 .152E00/ CHEB/ 1.247E00 1.391E00 -1.089E01 1.237E00 1.397E01/ CHEB/ -1.719E02 1.445E00 .173E02 -8.490E02/ CHEB/ -4.586E02 -2.490E02 -4.910E03 -2.279E02 -2.071E02/
CHEB/ 2.239E02 4.139E03/

C2C.OC (+M) ↔ C*CC + CH3O (+M) 1.00E+00 .000 0 ! 5 11/95
LOW / 1.0 0.0 0.0 /
CHEB/ 7 3 -1.206E01 1.323E00 -2.317E01 .152E00/ CHEB/ 1.248E00 1.357E00 1.139E01 1.245E01 .380E01/ CHEB/ -1.787E02 -1.442E00 3.979E03 .394E02 8.519E02/ CHEB/ -4.583E02 -2.498E02 -5.166E02 .216E02 -2.082E02/ CHEB/ -2.354E02 -2.478E02 -1.798E04 -1.151E02 -2.639E02/
CHEB/ 2.619E02 2.274E03/

C2C.OC (+M) ↔ C*C(C)C + H (+M) 1.00E+00 .000 0 ! 5 11/95
LOW / 1.0 0.0 0.0 /
CHEB/ 7 3 1.682E00 1.362E00 -2.824E00 7.318E00/ CHEB/ -1.031E00 1.343E01 -5.926E01 1.746E01 1.349E01/ CHEB/ -1.607E01 -1.047E02 2.160E02 -4.369E05 -5.667E02/
CHEB/ -2.144E02 -2.758E02 -1.815E03 -1.372E02 -2.707E02/
CHEB/ 1.686E02 1.729E03/

C2C.OC (+M) ↔ C*COC + CH3 (+M) 1.00E+00 .000 0 ! 5 11/55
LOW / 1.0 0.0 0.0 /
CHEB/ 7 3 3.677E01 1.490E00 -2.984E01 8.244E00/ CHEB/ 9.513E01 1.767E01 -5.735E01 1.452E01 1.234E01/ CHEB/ -1.500E02 7.351E02 1.201E04 2.370E04 2.416E02/ CHEB/ 2.534E02 4.352E02 -6.370E05 2.142E02 2.632E02/
CHEB/ 1.611E02 3.334E03/

C2C.OC (+M) ↔ CC.C + CH2O (+M) 1.00E+00 .000 0 ! 5 11/95
LOW / 1.0 0.0 0.0 /
CHEB/ 7 3 3.911E00 3.470E01 -3.513E01 4.349E01/ CHEB/ 1.969E00 1.769E01 -7.915E01 3.108E01 9.858E02/ CHEB/ -2.779E01 1.096E01 3.422E02 4.405E02 8.597E02/ CHEB/ 1.073E02 9.617E03 1.311E02 1.482E02 2.490E02/
CHEB/ 2.515E02 4.614E03/

C2C.OC (+M) ↔ C*C*O + CH3 (+M) 1.00E+00 .000 0 ! 5 11/95
LOW / 1.0 0.0 0.0 /
CHEB/ 7 3 -4.600E00 1.660E00 -3.203E01 1.068E01/ CHEB/ 8.316E01 2.602E01 -5.306E01 7.129E02 8.476E02/ CHEB/ -1.251E01 7.462E02 1.284E02 -1.895E02 -3.979E02/
CHEB/ -2.467E02 -1.542E02 4.511E03 -5.621E03 -2.421E02/
CHEB/ 1.475E02 6.140E03/

C3OC.C ↔ C3.COC 2.84E+00 0.0 10800.
C3.COC ↔ C2*C*C + CH3O 1.08E+12 0.0 17900.
C3.COC ↔ C*C(C)COC + CH3 9.66E+10 0.0 17900.
C3.COC ↔ C3.C + CH3O (+M) 1.00E+00 .000 0 ! 5 11/95
LOW / 1.0 0.0 0.0 /
CHEB/ 7 3 -1.574E01 5.288E01 6.819E02 2.405E01/ CHEB/ 9.351E01 -1.026E01 -8.624E01 6.340E01 2.773E02/
CHEB/ -5.344E01 2.994E01 3.650E02 2.672E01 5.565E02/
CHEB/ 5.735E02 9.057E02 5.551E02 3.973E02 -9.198E03/
CHEB/ -6.593E02 1.015E02/
C3COC (+M) <=> C2COC + CH3OH (+M) 1.00E+00 .000 0.1 ing503 11/95

LOW / 1.0 0.0 0.0 /
CHEB/ 7 3 -1.71E0 +01 3.59E-01 -4.97E0 +01 1.84E0 +02 /
CHEB/ 1.96E0 +01 5.49E-01 -7.97E0 +01 3.89E0 +02 /
CHEB/ 5.13E0 +01 3.59E-01 -5.30E0 +01 4.84E0 +02 /
CHEB/ -4.24E0 -01 2.40E-01 3.07E0 -02 2.87E0 -02 /
CHEB/ -2.31E0 -02 2.12E0 -02 /

C3COC (+M) <=> C2COC + CH3 (+M) 1.00E+00 .000 0.1 ing503 11/95

LOW / 1.0 0.0 0.0 /
CHEB/ 7 3 -1.65E0 +01 5.40E-01 -6.49E0 +01 2.49E0 +02 /
CHEB/ 1.38E0 +01 5.40E-01 -8.97E0 +01 2.49E0 +02 /
CHEB/ -5.30E0 -01 2.40E-01 3.07E0 -02 2.87E0 -02 /
CHEB/ 5.13E0 +01 3.59E-01 -5.30E0 +01 4.84E0 +02 /
CHEB/ -2.31E0 -02 2.12E0 -02 /

C3COC (+M) <=> C3COC + H (+M) 1.00E+00 .000 0.1 ing503 11/95

LOW / 1.0 0.0 0.0 /
CHEB/ 7 3 -2.46E0 +01 7.28E-01 -8.93E0 +01 2.99E0 +02 /
CHEB/ 1.18E0 +00 5.06E-01 -1.07E0 +00 6.18E-01 /
CHEB/ -5.30E0 -01 2.40E-01 3.07E0 -02 2.87E0 -02 /
CHEB/ 5.13E0 +01 3.59E-01 -5.30E0 +01 4.84E0 +02 /
CHEB/ -2.31E0 -02 2.12E0 -02 /

C3COC (+M) <=> C3COC + CH3 + H2O 1.00E+00 .000 0.1 ing503 11/95

LOW / 1.0 0.0 0.0 /
CHEB/ 7 3 -2.33E0 +01 6.49E-01 -7.71E0 +01 2.79E0 +02 /
CHEB/ 1.09E0 +00 9.64E-02 -1.02E0 +00 6.49E-01 /
CHEB/ -5.62E-01 2.17E0 +02 7.35E0 -02 -2.31E0 +02 /
CHEB/ 6.40E0 +00 9.79E0 -02 2.13E0 +02 1.70E0 +02 /
CHEB/ -6.67E0 -02 1.16E0 +02 /

C3COC (+M) <=> C3COC + CH3 + H2O2 1.00E+00 .000 0.1 ing503 11/95

LOW / 1.0 0.0 0.0 /
CHEB/ 7 3 -2.33E0 +01 6.49E-01 -7.71E0 +01 2.79E0 +02 /
CHEB/ 1.09E0 +00 9.64E-02 -1.02E0 +00 6.49E-01 /
CHEB/ -5.62E-01 2.17E0 +02 7.35E0 -02 -2.31E0 +02 /
CHEB/ 6.40E0 +00 9.79E0 -02 2.13E0 +02 1.70E0 +02 /
CHEB/ -6.67E0 -02 1.16E0 +02 /

C3COC (+M) <=> C3COC + CH3 + H2O2 1.00E+00 .000 0.1 ing503 11/95

LOW / 1.0 0.0 0.0 /
CHEB/ 7 3 -2.33E0 +01 6.49E-01 -7.71E0 +01 2.79E0 +02 /
CHEB/ 1.09E0 +00 9.64E-02 -1.02E0 +00 6.49E-01 /
CHEB/ -5.62E-01 2.17E0 +02 7.35E0 -02 -2.31E0 +02 /
CHEB/ 6.40E0 +00 9.79E0 -02 2.13E0 +02 1.70E0 +02 /
CHEB/ -6.67E0 -02 1.16E0 +02 /

C3COC (+M) <=> C3COC + CH3 + H2O2 1.00E+00 .000 0.1 ing503 11/95

LOW / 1.0 0.0 0.0 /
CHEB/ 7 3 -2.33E0 +01 6.49E-01 -7.71E0 +01 2.79E0 +02 /
CHEB/ 1.09E0 +00 9.64E-02 -1.02E0 +00 6.49E-01 /
CHEB/ -5.62E-01 2.17E0 +02 7.35E0 -02 -2.31E0 +02 /
CHEB/ 6.40E0 +00 9.79E0 -02 2.13E0 +02 1.70E0 +02 /
CHEB/ -6.67E0 -02 1.16E0 +02 /
Figure B.1 Experimental Apparatus
Figure B.2 Calibration curve of mass flow controller 1 for argon with inlet pressure of 200 psig.
Figure B.3 Calibration curve of mass flow controller 2 for Argon with inlet pressure of 200 psig.
Figure B.4 Calibration curve of mass flow controller 3 for methane with inlet pressure of 200 psig.
Figure B.5 Calibration curve of mass flow controller 4 for oxygen with inlet pressure of 200 psig.
Figure B.6 Reactor Temperature Profiles at 1.3 gm Ar / min
Figure B.7 Reactor Temperature Profiles at 1 atm
Figure B.8 Experimental result of methanol pyrolysis: $T = 1073 \text{ K}; p = 1 \text{ atm}; X_{CH_3OH} = 0.0395$
Figure B.9 Experimental result of methanol pyrolysis: $T = 1073 \text{ K}; p = 3 \text{ atm}; X_{0, \text{CH}_3\text{OH}} = 0.01317$
Figure B.10 Experimental result of methanol pyrolysis: $T = 1073$ K; $p = 5$ atm; $X_{CH_{3}OH} = 0.0079$
Figure B.11 Experimental result of methanol pyrolysis: $T = 1073$ K; $p = 8$ atm; $X_{0,\text{CH}_3\text{OH}} = 0.00494$
Figure B.12 Experimental result of methanol pyrolysis: $T = 1073$ K; $p = 8$ atm; $X_{o,\text{CH}_3\text{OH}} = 0.0025$
Figure B.13 Experimental result of methanol pyrolysis: $T = 1073$ K; $p = 10$ atm; $X_{\text{CBOH}} = 0.00395$
Figure B.14 Experimental result of methanol oxidation: $\phi = 1.0; T = 873$ K; $p = 3$ atm; $X_{CH_3OH} = 0.0078$. 
Figure B.15 Experimental result of methanol oxidation: $\varphi = 0.75$; $T = 873$ K; $p = 3$ atm; $X_{0,\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.16 Experimental result of methanol oxidation: $\phi = 1.0$; $T = 873$ K; $p = 5$ atm; $X_{\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.17 Experimental result of methanol oxidation: $\phi = 0.75$; $T = 873$ K; $p = 5$ atm; $X_{\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.18 Experimental result of methanol oxidation: $\varphi = 1.0$; $T = 923$ K; $p = 1$ atm; $X_{\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.19 Experimental result of methanol oxidation: $\phi = 0.75$; $T = 923$ K; $p = 1$ atm; $X_{\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.20 Experimental result of methanol oxidation: $\varphi = 1.0$; $T = 923$ K; $p = 3$ atm; $X_{\text{c,CH}_3\text{OH}} = 0.0078$. 
Figure B.21 Experimental result of methanol oxidation: $\varphi = 0.75$; $T = 923$ K; $p = 3$ atm; $X_{o, \text{CH}_3\text{OH}} = 0.0078$. 
Figure B.22 Experimental result of methanol oxidation: $\varphi = 1.0; T = 923 \text{ K}; p = 5 \text{ atm}; X_{\text{CH}_3\text{OH}} = 0.0078$. 
**Figure B.23** Experimental result of methanol oxidation: $\varphi = 0.75; \ T = 923 \ \text{K}; \ p = 5 \ \text{atm}; \ X_{0, \text{CH}_3\text{OH}} = 0.0078.$
Figure B.24 Experimental result of methanol oxidation: φ = 1.0; T = 973 K; p = 1 atm; X_{CH\textsubscript{3}OH} = 0.0078.
Figure B.25 Experimental result of methanol oxidation: $\varphi = 0.75$; $T = 973$ K; $p = 1$ atm; $X_{\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.26 Experimental result of methanol oxidation: $\varphi = 1.0; T = 973 \text{ K}; p = 3 \text{ atm}; X_{o,\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.27 Experimental result of methanol oxidation: $\phi = 0.75$; $T = 973$ K; $p = 3$ atm; $X_{o,\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.28 Experimental result of methanol oxidation: $\varphi = 1.0$; $T = 973$ K; $p = 5$ atm; $X_{0,\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.29 Experimental result of methanol oxidation: $\phi = 0.75; T = 973 \, K; p = 5 \, \text{atm}; X_{\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.30 Experimental result of methanol oxidation: $\varphi = 1.0; T = 1023$ K; $p = 1$ atm; $X_{0,\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.31 Experimental result of methanol oxidation: $\varphi = 0.75$; $T = 1023$ K; $p = 1$ atm; $X_{\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.32 Experimental result of methanol oxidation: $\varphi = 1.0$, $T = 1023$ K, $p = 3$ atm; $X_{\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.33 Experimental result of methanol oxidation: $\varphi = 0.75$; $T = 1023$ K; $p = 3$ atm; $X_{e,\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.34 Experimental result of methanol oxidation: $\varphi = 1.0; T = 1023 \text{ K}; p = 5 \text{ atm}; X_{\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.35 Experimental result of methanol oxidation: $\varphi = 0.75$; $T = 1023$ K; $p = 5$ atm; $X_{0,\text{CH}_{3}OH} = 0.0078$. 
Figure B.36 Experimental result of methanol oxidation: $\phi = 1.0$; $T = 1073$ K; $p = 1$ atm; $X_{0,\text{OH}} = 0.0078$. 
Figure B.37 Experimental result of methanol oxidation: \( \varphi = 0.75; T = 1073 \text{ K}; p = 1 \text{ atm}; X_{\text{CH}_3\text{OH}} = 0.0078.\)
Figure B.38 Experimental result of methanol oxidation: $\phi = 1.0$; $T = 1073$ K; $p = 3$ atm; $X_{\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.39 Experimental result of methanol oxidation: $\varphi = 0.75$; $T = 1073$ K; $p = 3$ atm; $X_{\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.40 Experimental result of methanol oxidation: \( \varphi = 1.0; T = 1073 \text{ K}; p = 5 \text{ atm}; X_{0,\text{CH}_3\text{OH}} = 0.0078. \)
Figure B.41 Experimental result of methanol oxidation: \( \varphi = 0.75; \, T = 1073 \, K; \, p = 5 \, \text{atm; } X_{0,\text{CH}_3\text{OH}} = 0.0078. \)
Figure B.42 Experimental result of methanol oxidation: $\varphi = 1.0$; $t = 0.15$ s; $p = 1$ atm; $X_{o,\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.43 Experimental result of methanol oxidation: $\phi = 0.75; t = 0.15 \text{ s}; p = 1 \text{ atm}; X_{o,\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.44 Experimental result of methanol oxidation: $\phi = 1.0; t = 0.2 \text{ s}; p = 1 \text{ atm}; X_{0,\text{CH}_3\text{OH}} = 0.0078.$
Figure B.45 Experimental result of methanol oxidation: $\varphi = 0.75$; $t = 0.2$ s; $p = 1$ atm; $X_{\text{CH}_3\text{OH}} = 0.0078$. 

The graph shows the mole fraction of various products (CH$_2$OH, CO$_2$, CO, CH$_2$O, and CH$_4$) as a function of temperature (K) over the range of 800 to 1050 K.
Figure B.46 Experimental result of methanol oxidation: $\varphi = 1.0; \ t = 0.25 \ \text{s}; \ p = 1 \ \text{atm}; \ X_{\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.47 Experimental result of methanol oxidation: $\varphi = 0.75$; $t = 0.25$ s; $p = 1$ atm; $X_{o,\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.48 Experimental result of methanol oxidation: $\phi = 1.0; t = 0.4 \text{ s}; p = 3 \text{ atm}; X_{\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.49 Experimental result of methanol oxidation: $\phi = 0.75; t = 0.4 \text{ s}; p = 3 \text{ atm}; X_{\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.50 Experimental result of methanol oxidation: $\varphi = 1.0; t = 0.6 \text{ s}; p = 3 \text{ atm}; X_{\text{CH}_3\text{OH}} = 0.0078.$
Figure B.51 Experimental result of methanol oxidation: $\varphi = 0.75; t = 0.6 \text{ s}; p = 3 \text{ atm}; X_{e,\text{CH}_3\text{OH}} = 0.0078.$
Figure B.52 Experimental result of methanol oxidation: $\phi = 1.0$; $t = 0.8$ s; $p = 3$ atm; $X_{o,\text{CH}_3\text{OH}} = 0.0078$. 
**Figure B.53** Experimental result of methanol oxidation: \( \phi = 0.75; \ t = 0.8 \ s; \ p = 3 \ \text{atm}, \ X_{o,\text{CH}_{3}\text{OH}} = 0.0078.\)
**Figure B.54** Experimental result of methanol oxidation: $\varphi = 1.0; t = 0.6$ s; $p = 5$ atm; $X_{CH_3OH} = 0.0078$. 
Figure B.55 Experimental result of methanol oxidation: \( \varphi = 0.75; t = 0.6 \text{ s}; p = 5 \text{ atm}; \\ X_{\text{CH}_3\text{OH}} = 0.0078.\)
Figure B.56 Experimental result of methanol oxidation: $\varphi = 1.0; t = 0.8 \text{ s}; p = 5 \text{ atm}; X_{o,\text{CH}_3\text{OH}} = 0.0078.$
Figure B.57 Experimental result of methanol oxidation: $\phi = 0.75; t = 0.8 \text{ s}; p = 5 \text{ atm}; X_{\text{CH}_3\text{OH}} = 0.0078.$
Figure B.58 Experimental result of methanol oxidation: $\varphi = 1.0; t = 1.0 \text{ s}; p = 5 \text{ atm}; X_{o, \text{CH}_3\text{OH}} = 0.0078.$
Figure B.59 Experimental result of methanol oxidation: $\phi = 0.75$; $t = 1.0$ s; $P = 5$ atm; $X_{CH_3OH} = 0.0078$. 
Figure B.60 Experimental result of methanol oxidation: $\phi = 1.0; t = 1.5 \text{ s}; p = 5 \text{ atm}; X_{\text{CH}_3\text{OH}} = 0.0078.$
Figure B.61 Experimental result of methanol oxidation: $\phi = 0.75; t = 1.5 \text{ s}; p = 5 \text{ atm}; X_{\text{CH}_3\text{OH}} = 0.0078$. 
Figure B.62 Experimental result of methane / methanol oxidation: $\varphi = 1.0; T = 873 \, K; p = 5 \, \text{atm}; X_{0,\text{CH}_4} = 0.0078; X_{0,\text{CH}_3\text{OH}} = 0.0078.$
Figure B.63 Experimental result of methane / methanol oxidation: $\phi = 1.0$; $T = 873$ K; $p = 5$ atm; $X_{0,CH_4} = 0.01014$; $X_{0,CH_3OH} = 0.00546$. 
Figure B.64 Experimental result of methane / methanol oxidation: \( \varphi = 1.0; T = 873 \text{ K}; p = 5 \text{ atm}; X_{\text{CH}_4} = 0.0117; X_{\text{CH}_3\text{OH}} = 0.0039.\)
Figure B.65 Experimental result of methane / methanol oxidation: $\varphi = 1.0$; $T = 873 \text{ K}$; $p = 5 \text{ atm}$; $X_{0,\text{CH}_4} = 0.01326$; $X_{0,\text{CH}_3\text{OH}} = 0.00234$. 
Figure B.66 Experimental result of methane / methanol oxidation: $\varphi = 1.0; T = 873 \, \text{K}; p = 5 \, \text{atm}; X_{O_{2,CH_4}} = 0.01404; X_{O_{2,CH_3OH}} = 0.00156$. 

- $\text{CH}_4$
- $\text{CH}_3\text{OH}$
- $\text{CH}_2\text{O}$
- CO
- $\text{C}_2\text{H}_6$
- $\text{C}_2\text{H}_4$
Figure B.67 Experimental result of methane / methanol oxidation: $\varphi = 1.0$; $T = 873$ K; $p = 5$ atm; $X_{i,CH_4} = 0.0039$; $X_{i,CH_3OH} = 0.0039$. 
Figure B.68 Experimental result of methane/methanol oxidation: $\varphi = 1.0$; $T = 873$ K; $p = 5$ atm; $X_{0,\text{CH}_4} = 0.00273$; $X_{0,\text{CH}_3\text{OH}} = 0.00507$. 
Figure B.69 Experimental result of methane / methanol oxidation: $\varphi = 1.0$; $T = 873$ K; $p = 5$ atm; $X_{a,\text{CH}_4} = 0.00195$; $X_{a,\text{C}_2\text{H}_5\text{OH}} = 0.00585$. 
Figure B.70 Experimental result of methane / methanol oxidation: $\varphi = 1.0; \ T = 1023$ K; $p = 1$ atm; $X_{0,CH_4} = 0.0078$; $X_{0,CH_3OH} = 0.0078$. 
Figure B.71 Experimental result of methane / methanol oxidation: $\varphi = 1.0$; $T = 1073$ K; $p = 1$ atm; $X_{\text{CH}_4} = 0.0156$; $X_{\text{CH}_3\text{OH}} = 0.0$. 
Figure B.72 Experimental result of methane / methanol oxidation: $\varphi = 1.0; T = 1073$ K; $p = 1$ atm; $X_{o,CH_4} = 0.0078; X_{o,CH_3OH} = 0.0078$. 
Figure B.73 Experimental result of methane / methanol oxidation: $\varphi = 1.0; T = 1073$ K; $p = 1$ atm; $X_{0,\text{CH}_4} = 0.0; X_{0,\text{CH}_3\text{OH}} = 0.0156$. 
$\text{C}_2\text{H}_6 \rightarrow \text{CH}_3 + \text{CH}_3$

Figure B.74 Rate coefficients for $\text{C}_2\text{H}_6 \rightarrow \text{CH}_3 + \text{CH}_3$
Figure B.75 Rate coefficients for (a) \( \text{CH}_2\text{OH} + \text{O}_2 \rightarrow \text{CH}_2\text{O} + \text{HO}_2 \)
(b) \( \text{CH}_2\text{OH} + \text{O}_2 \rightarrow \text{HOC(OO)H}_2 \)
Figure B.76 Relative errors for \( \text{CH}_2\text{OH} + \text{O}_2 \rightarrow \text{HOC(OO)H}_2 \)
(a) Chebyshev 7x3 (b) Chebyshev 9x5
Figure B.77 Relative errors for CH₂OH + O₂ → CH₂O + HO₂
(a) Chebyshev 7×3  (b) Chebyshev 9×5
Figure B.78 Relative errors for CH$_2$OH + O$_2$ → HOC(O)H$_2$

(a) Chebyshev 7×3  (b) Chebyshev 9×5
Figure B.79 Potential Energy Diagram of CH₂OH + O₂ and H₂C=O + HO₂ Reaction Systems
Figure B.80 Results of QRRK Analysis for CH$_2$OH + O$_2$ $\leftrightarrow$ [H$_2$(OH)COO•]* $\rightarrow$
Product at 300K
Figure B.81 Results of QRRK Analysis for \( \text{CH}_2\text{OH} + \text{O}_2 \leftrightarrow [\text{H}_2\text{(OH)COO}^*] \rightarrow \) Product at 900K
Figure B.82 Results of QRRK Analysis for CH$_2$OH + O$_2$ ↔ [H$_2$(OH)COO•]* → Product at 1800K
Figure B.83 Results of QRRK Analysis for $\text{CH}_2\text{OH} + \text{O}_2 \leftrightarrow [\text{H}_2(\text{OH})\text{COO}^\bullet]^\text{+} \rightarrow$ Product at 1 atm
Figure B.84 Results of QRRK Analysis for CH$_2$OH + O$_2$ $\leftrightarrow$ [H$_2$(OH)COO•]* $\rightarrow$
Product at 10 atm
Figure B.85 Comparison of QRRK Analysis and Experimental Data for CH$_2$OH + O$_2$ → H$_2$C=O + HO$_2$
Figure B.86 Results of QRRK Analysis for $\text{H}_2\text{C}=\text{O} + \text{HO}_2 \leftrightarrow [\text{H}_2\text{CO..HOO\textbullet}]^* \rightarrow$ Product at 300 K
Figure B.87 Results of QRRK Analysis for H$_2$C=O + HO$_2$ ↔ [H$_2$CO..HO0•]* → Product at 900 K
Figure B.88 Results of QRRK Analysis for $\text{H}_2\text{C}=\text{O} + \text{HO}_2 \leftrightarrow [\text{H}_2\text{CO}.\text{HOO}\cdot]^*$ → Product at 1800 K
**Figure B.89** Results of QRRK Analysis for H$_2$C=O + HO$_2$ ↔ [H$_2$CO..HOO•]•* → Product at 1 atm
Figure B.90 Comparison of QRRK Analysis and Experimental Data for H₂C=O + HO₂ ↔ H₂(OH)COO⁻
Figure B.91 Experimental Results of Methanol Pyrolysis at 873 K
Figure B.92 Potential Energy Diagram for CH$_3$OH $\rightarrow$ Products
Figure B.93 Comparison of Model and Experimental Results for Methanol Pyrolysis at 1073 K, 1 atm, $X_o = 3.95\%$
Figure B.94 Comparison of Model and Experimental Results for Methanol Pyrolysis at 1073 K, 3 atm, X₀ = 1.317%
Figure B.95 Comparison of Model and Experimental Results for Methanol Oxidation at 873 K, 5 atm, $X_0 = 0.78\%$, $\phi = 1.0$
Figure B.96 Comparison of Model and Experimental Results for Methanol Oxidation at 873 K, 5 atm, $X_o = 0.78\%$, $\phi = 0.75$
Figure B.97 Comparison of Model and Experimental Results for Methanol Oxidation at Residence Time = 0.8 second, 5 atm, $X_o = 0.78\%$, $\phi = 1.0$
Figure B.98 Comparison of Model and Experimental Results for Methanol Oxidation at Residence Time = 1.0 second, 5 atm, $X_0 = 0.78\%$, $\phi = 1.0$
Figure B.99 Comparison of Model and Experimental Results for Methanol Oxidation at Residence Time = 1.5 second, 5 atm, $X_0 = 0.78\%$, $\phi = 1.0$
Figure B.100 Comparison of Model and Experimental Results for Methanol Oxidation at 923 K, 3 atm, $X_0 = 0.78\%$, $\phi = 1.0$
Figure B.101 Potential Energy Diagram for CH$_3$ + O$_2$ $\leftrightarrow$ [CH$_3$OO]$^+$ $\rightarrow$ Products
Figure B.102 Comparisons of QRRK Analysis Results and Experimental Data for CH$_3$ + O$_2$ → CH$_3$OO (a) Bath Gas = Ar (b) Bath Gas = N$_2$
Figure B.103 Comparisons of QRRK Analysis Results and Experimental Data for (a) $\text{CH}_3 + \text{O}_2 \rightarrow \text{CH}_3\text{O} + \text{O}$ (b) $\text{CH}_3 + \text{O}_2 \rightarrow \text{CH}_2\text{O} + \text{OH}$
Figure B.104 Results of QRRK Analysis for CH$_3$ + O$_2$ → [CH$_3$OO]* → Products
Figure B.105 Comparison of Model and Experimental Results for Methane/Methanol Oxidation at 873 K, 1 atm, $X_o(\text{CH}_4) = 0.78\%$, $X_o(\text{CH}_3\text{OH}) = 0.78\%$, $\phi = 1.0$
Figure B.106 Comparison of Model and Experimental Results for Methane/Methanol Oxidation at 873 K, 1 atm, $X_0(\text{CH}_4) = 0.39\%$, $X_0(\text{CH}_3\text{OH}) = 0.39\%$, $\phi = 1.0$
Figure B.107 Comparison of Methane Decay of Model and Experimental Results for Methane/Methanol Oxidation at 873 K, 1 atm, $X_{0,\text{TOTAL}} = 1.56$, $\phi = 1.0$
Figure B.108 Comparison of Methanol Decay of Model and Experimental Results for Methanol and Methane/Methanol Oxidation at 873 K, 5 atm, $\phi = 1.0$
**Figure B.109** Comparison of Methanol Decay of Model and Experimental Results for Methanol and Methane/Methanol Oxidation at 873 K, $X_{\text{TOTAL}} = 0.78\%$, 5 atm, $\phi = 1.0$
Figure B.110 MTBE Pyrolysis, $C_0 = 0.5\%$, $-\ln(C/C_0)$ versus Retention Time
Potential Energy Diagram for \((\text{CH}_3)_2\text{COCH}_3\) Dissociation

Figure B.111 Potential Energy Diagram for MTBE → Products
Figure B.112 Comparisons of Experimental and Model Results for MTBE Pyrolysis, $C_o = 0.5\%$
Figure B.113 Model Comparison with Norton's Experimental Data for MTBE Oxidation at 1024 K, 1 atm, $\phi = 0.96$ (Model Result 10 msec offset)
APPENDIX C

QRRK INPUT PARAMETERS

C.1 CH₃OH → Products

<table>
<thead>
<tr>
<th>reaction</th>
<th>A (s⁻¹ or cm³/(mol*s))</th>
<th>Ea (kcal/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CH₃ + OH → CH₃OH</td>
<td>2.62E+13</td>
<td>0.0</td>
</tr>
<tr>
<td>-1 CH₃OH → CH₃ + OH</td>
<td>6.30E+15</td>
<td>91.8</td>
</tr>
<tr>
<td>2 CH₃OH → ¹HCOH + H₂</td>
<td>1.09E+14</td>
<td>85.0</td>
</tr>
<tr>
<td>3 CH₃OH → ¹CH₂ + H₂O</td>
<td>4.40E+14</td>
<td>89.8</td>
</tr>
<tr>
<td>4 CH₃OH → CH₂O + H₂</td>
<td>1.27E+13</td>
<td>91.9</td>
</tr>
<tr>
<td>5 CH₃OH → CH₂OH + H</td>
<td>1.30E+15</td>
<td>95.0</td>
</tr>
<tr>
<td>6 CH₃OH → CH₂O + H</td>
<td>2.39E+14</td>
<td>103.4</td>
</tr>
</tbody>
</table>

frequencies/degeneracies (from CPFIT): 738 cm⁻¹/3.049, 1729 cm⁻¹/5.932, 3695 cm⁻¹/2.518

Lennard-Jones parameters: σ = 3.78 Å, ε/k = 377 K

Microscopic Reversibility (MR).


C.2 CH\textsubscript{3} + O\textsubscript{2} → [CH\textsubscript{3}OO•]\* → Products

\[ \text{CH}_3 + \text{O}_2 \rightarrow \text{CH}_3\text{OO}• \rightarrow \text{Products} \]

Kcal/mol

<table>
<thead>
<tr>
<th>reaction</th>
<th>A (s\textsuperscript{-1} or cm\textsuperscript{3}/(mol\textsuperscript{*}s))</th>
<th>Ea (kcal/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CH\textsubscript{3} + O\textsubscript{2} → CH\textsubscript{3}OO•</td>
<td>6.44E+10 T\textsuperscript{0.53}</td>
<td>0.0</td>
</tr>
<tr>
<td>-1 CH\textsubscript{3}OO• → CH\textsubscript{3} + O\textsubscript{2}</td>
<td>3.09E+15 T\textsuperscript{-0.33}e\textsuperscript{-0.00207T}</td>
<td>30.2</td>
</tr>
<tr>
<td>2 CH\textsubscript{3}OO• → CH\textsubscript{3}O + O</td>
<td>3.10E+15</td>
<td>58.5</td>
</tr>
<tr>
<td>3 CH\textsubscript{3}OO• → CH\textsubscript{2}OOH</td>
<td>6.00E+12</td>
<td>42.3</td>
</tr>
<tr>
<td>-3 CH\textsubscript{2}OOH → CH\textsubscript{3}OO•</td>
<td>1.60E+18 T\textsuperscript{-2.49}e\textsuperscript{-0.00207T}</td>
<td>32.0</td>
</tr>
<tr>
<td>4 CH\textsubscript{2}OOH → CH\textsubscript{2}O + OH</td>
<td>5.50E+13</td>
<td>2.0</td>
</tr>
</tbody>
</table>

frequencies/degen (from CPFIT): 525 cm\textsuperscript{-1}/3.204, 801 cm\textsuperscript{-1}/2.752, 1912 cm\textsuperscript{-1}/5.544

Lennard-Jones parameters: \( \sigma = 4.36 \text{ Å}, \epsilon / k = 471 \text{ K} \)

\( k_1 \) Taken from Cobos et al., *J. Phys. Chem.*, 89, 4332 (1985).

\( k_{-1} \) Microscopic Reversibility (MR) with the temperature range of 298 K and 1500 K.

\( k_2 \) A\textsubscript{2} from MR with A\textsubscript{3} = 5.00E+13 for the high-pressure recombination rate constant.

\( k_3 \) A\textsubscript{3} = 6.00E+12 as Benson’s for H-transfer, E\textsubscript{a3} = \triangle H\textsubscript{3} + ring strain (26.0) + E\textsubscript{abstraction} (6.0)

\( k_4 \) MR with the temperature range of 298 K and 2000 K.

\( k_4 \) A\textsubscript{4} from MR with reverse taken as the addition of OH to CH\textsubscript{3}CHO, Semmes et al.; E\textsubscript{a4} = 2.0 from the intrinsic activation energy expected for OH addition.
C.3 $\text{CH}_2\text{OH} + \text{O}_2 \leftrightarrow [\text{H}_2(\text{OH})\text{COO}^\bullet]^* \rightarrow \text{Product}$
<table>
<thead>
<tr>
<th>Reaction</th>
<th>$A$ (s$^{-1}$ or cm$^3$/mol*s)</th>
<th>$E_a$ (kcal/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 $\text{CH}_2\text{OH} + \text{O}_2 \rightarrow \text{H}_2\text{(OH)COO}^\ast$</td>
<td>6.00E+12</td>
<td>0.0</td>
</tr>
<tr>
<td>-1 $\text{H}_2\text{(OH)COO}^\ast \rightarrow \text{CH}_2\text{OH} + \text{O}_2$</td>
<td>2.93E+16 $T^{0.54}e^{-0.00118T}$</td>
<td>34.4</td>
</tr>
<tr>
<td>2 $\text{H}_2\text{(OH)COO}^\ast \rightarrow \text{H}_2\text{(OH)CO}^\ast + \text{O}$</td>
<td>7.55E+14</td>
<td>57.5</td>
</tr>
<tr>
<td>3 $\text{H}_2\text{(OH)COO}^\ast \rightarrow \text{H}_2\text{CO..HOO}^\ast$</td>
<td>4.96E+6 $T^{2.11}e^{-0.00069T}$</td>
<td>8.6</td>
</tr>
<tr>
<td>-3 $\text{H}_2\text{CO..HOO}^\ast \rightarrow \text{H}_2\text{(OH)COO}^\ast$</td>
<td>3.76E+7 $T^{1.0}$</td>
<td>1.3</td>
</tr>
<tr>
<td>4 $\text{H}_2\text{CO..HOO}^\ast \rightarrow \text{H}_2\text{C}=\text{O} + \text{HO}_2$ (I)</td>
<td>5.87E+17 $T^{2.68}e^{-0.00007T}$</td>
<td>7.2</td>
</tr>
<tr>
<td>5 $\text{H}_2\text{(OH)COO}^\ast \rightarrow \text{CH}_2\text{(OOH)O}^\ast$</td>
<td>6.86E+8 $T^{1.0}$</td>
<td>21.8</td>
</tr>
<tr>
<td>-5 $\text{CH}_2\text{(OOH)O}^\ast \rightarrow \text{H}_2\text{(OH)COO}^\ast$</td>
<td>5.56E+8 $T^{0.84}e^{0.00042T}$</td>
<td>7.7</td>
</tr>
<tr>
<td>6 $\text{CH}_2\text{(OOH)O}^\ast \rightarrow \text{H}_2\text{C}=\text{O} + \text{HO}_2$ (II)</td>
<td>1.40E+14</td>
<td>5.0</td>
</tr>
<tr>
<td>7 $\text{H}_2\text{(OH)COO}^\ast \rightarrow \text{C}^\ast\text{H(OOH)OH}$</td>
<td>5.90E+9 $T^{1.0}$</td>
<td>37.1</td>
</tr>
<tr>
<td>-7 $\text{C}^\ast\text{H(OOH)OH} \rightarrow \text{H}_2\text{(OH)COO}^\ast$</td>
<td>4.19E+9 $T^{0.8}e^{0.00004T}$</td>
<td>31.6</td>
</tr>
<tr>
<td>8 $\text{C}^\ast\text{H(OOH)OH} \rightarrow \text{H(OH)C}=\text{O} + \text{OH}$</td>
<td>3.31E+13</td>
<td>1.0</td>
</tr>
<tr>
<td>9 $\text{C}^\ast\text{H(OOH)OH} \rightarrow \text{H(OOH)C}=\text{O} + \text{H}$</td>
<td>3.24E+13</td>
<td>22.5</td>
</tr>
</tbody>
</table>

frequencies/degeneracies (from CPFIT): 415 cm$^{-1}$/6.178, 1506 cm$^{-1}$/5.998, 3198 cm$^{-1}$/2.824
Lennard-Jones parameters: $\sigma = 4.83 \text{ Å}, \varepsilon/k = 488 \text{ K}$
$<\Delta E>_{\text{down}} = 800 \text{ cal/mol}, \text{ Bath gas} = \text{N}_2$

$k_1$ $A_1$ taken as double of CC·$+\text{O}_2$ addition from J. W. Bozzelli and A. M. Dean, J. Phys. Chem., 1990.

$k_2$ Microscopic Reversibility (MR) with the temperature range of 298 K and 2000 K.

$k_3$ $A_3 = (10^{13.55})(10^{\Delta S/4.6})$, $\Delta S^\# = -6.4$; $Ea_3 = \Delta H_3 + 2.0$ by transition state calculation, transition state thermo properties based on Evleth et al., J. Phys. Chem., 1993.

$k_4$ $A_4 = (ekT/h)[\exp(\Delta S^\#/R)]$, $\Delta S^\# = -14.4$; $Ea_4 = 2.0$ by transition state calculation, transition state thermo properties based on Evleth et al., J. Phys. Chem., 1993.

$k_5$ Based on the study of $\text{CH}_2\text{O} + \text{HO}_2$ system in this paper.

$k_6$ $A_6$ from MR with $A_6 = 6.05E+11$ and $Ea_6 = \Delta H_6 + 5.0$, based on addition of $\text{HO}_2$ to $\text{C}_2\text{H}_4$ by Tsang's recommendation, J. Phys. Chem. Ref. Data, 1987.

$k_7$ $A_7 = (ekT/h)[\exp(\Delta S^\#/R)](2)$, $\Delta S = -4.3$ (loss of 2 rotors); $Ea_7 = \Delta H_7 + \text{ring strain} (8.6) + E_{\text{abstraction}} (3.0)$, ring strain from J. W. Bozzelli and A. M. Dean, J. Phys. Chem., 1990.

$k_8$ MR with the temperature range of 298 K and 2000 K.

$k_9$ $A_9$ from MR with $A_9 = 6.05E+11$ and $Ea_9 = \Delta H_9 + 5.0$, based on addition of $\text{HO}_2$ to $\text{C}_2\text{H}_4$ by Tsang's recommendation, J. Phys. Chem. Ref. Data, 1987.

$k_{10}$ $A_{10} = (ekT/h)[\exp(\Delta S^\#/R)](2)$, $\Delta S = -4.3$ (loss of 1 rotors); $Ea_{10} = \Delta H_9 + \text{ring strain} (8.6) + E_{\text{abstraction}} (8.6)$, ring strain from J. W. Bozzelli and A. M. Dean, J. Phys. Chem., 1990.

$k_{11}$ MR with the temperature range of 298 K and 2000 K.
$k_8$ A$_8$ from MR with $A_{A_8} = 2.7\times10^{12}$, which is one half of the rate constant for addition of OH to C$_2$H$_4$, Atkinson et al., J. Phys. Chem., 1989; $E_{A_8}$ from Soto and Page, Chem. Phys. 1991.

$k_9$ A$_9$ from MR with $A_{A_9} = 1.46\times10^{13}$, which is one half of the rate constant for addition of H to C$_2$H$_4$ and $E_{A_9} = \Delta H_2 + 2.7$, based on NIST fitting.
C.4 CH₂O + HO₂ ↔ [CH₂O...HOO•]⁺ → Product

<table>
<thead>
<tr>
<th>Reaction</th>
<th>A (s⁻¹ or cm³/(mol·s))</th>
<th>E⁺ (kcal/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 H₂C=O + HO₂ → H₂C=O.HOO•</td>
<td>5.00E+10</td>
<td>0.0</td>
</tr>
<tr>
<td>-1 H₂C=O.HOO• → H₂C=O + HO₂</td>
<td>5.87E+17 T⁻².66e⁻0.00007T</td>
<td>7.2</td>
</tr>
<tr>
<td>2 H₂CO.HOO• → H₂(OH)COO•</td>
<td>3.76E+7 T⁻¹.0</td>
<td>1.3</td>
</tr>
<tr>
<td>-2 H₂(OH)COO• → H₂CO.HOO•</td>
<td>4.96E+6 T⁻².11e⁻0.00009T</td>
<td>8.6</td>
</tr>
<tr>
<td>3 H₂(OH)COO• → CH₂OH + O₂</td>
<td>9.00E+14</td>
<td>34.4</td>
</tr>
<tr>
<td>4 H₂(OH)COO• → H₂(OH)CO• + O</td>
<td>7.55E+14</td>
<td>57.5</td>
</tr>
<tr>
<td>5 H₂(OH)COO• → CH₂(OOH)OH</td>
<td>6.86E+8 T⁻¹.0</td>
<td>21.8</td>
</tr>
<tr>
<td>-5 CH₂(OOH)OH → H₂(OH)COO•</td>
<td>5.56E+8 T⁻⁰.⁸⁴e⁰.⁰⁰⁰⁰⁴²T</td>
<td>7.7</td>
</tr>
<tr>
<td>6 CH₂(OOH)OH → CH₂O + HO₂</td>
<td>1.40E+14</td>
<td>5.0</td>
</tr>
<tr>
<td>7 H₂(OH)COO• → C•H(OOH)OH</td>
<td>5.90E+9 T⁻¹.⁰</td>
<td>37.1</td>
</tr>
<tr>
<td>-7 C•H(OOH)OH → H₂(OH)COO• + 0</td>
<td>4.19E+9 T⁻⁰.⁸⁶e⁰.⁰⁰⁰⁰⁴⁴T</td>
<td>31.6</td>
</tr>
<tr>
<td>8 C•H(OOH)OH → H(OH)C=O + OH</td>
<td>3.31E+13</td>
<td>1.0</td>
</tr>
<tr>
<td>9 C•H(OOH)OH → H(OOH)C=O + H</td>
<td>3.24E+13</td>
<td>22.5</td>
</tr>
</tbody>
</table>

frequencies/degenercies (from CPFIT): 188 cm⁻¹/5.857, 1308 cm⁻¹/5.714, 2952 cm⁻¹/3.430

Lennard-Jones parameters: σ = 4.83 Å, e/k = 488 K

<ΔE> down = 800 cal/mol, Bath gas = N₂

k₁ This study, estimated from the reaction of CH₂O+HO₂ = CQ·H₂OH, Veyret et al., J. Phys. Chem., 1989.

k₂ This study, estimated from the reaction of CH₂O+HO₂ = CQ·H₂OH, Veyret et al., J. Phys. Chem., 1989.

k₂ Microscopic Reversibility (MR) with the temperature range of 298 K and 2000 K.

k₃ from MR with A³ taken as double of CC·+O₂ addition from J. W. Bozzelli and A. M. Dean, J. Phys. Chem., 1990.

k₄ from MR with A₄ = 2.00E+13 based on addition of O to CH₃.

k₅ from MR with A₅ = 2.00E+13 based on addition of O to CH₃.

\[ k_7 \quad A_7 = \left( \frac{ekT}{h} \right) \exp(\frac{\Delta S}{R}) \] (2), \( \Delta S = -4.3 \) (loss of 1 rotors); \( E_{a7} = \Delta H_7 + \) ring strain (23) + \( E_{\text{abstraction}} \) (8.6), ring strain from J. W. Bozzelli and A. M. Dean, J. Phys. Chem., 1990.

\( k_7 \) MR with the temperature range of 298 K and 2000 K.

\( k_8 \) \( A_8 \) from MR with \( A_{a8} = 2.7E+12 \), which is one half of the rate constant for addition of OH to \( \text{C}_2\text{H}_4 \), Atkinson et al., J. Phys. Chem., 1989; \( E_{a8} \) from Soto and Page, Chem. Phys. 1991.

\( k_9 \) \( A_9 \) from MR with \( A_{a9} = 1.46E+13 \), which is one half of the rate constant for addition of H to \( \text{C}_2\text{H}_4 \) and \( E_{a7} = \Delta H_7 + 2.7 \), based on NIST fitting.
C.5 CH$_2$OH + HO$_2$ $\leftrightarrow$ [CQH$_2$OH]$^*$ $\rightarrow$ Product

<table>
<thead>
<tr>
<th>Reaction</th>
<th>$A$ (s$^{-1}$ or cm$^3$/mol s)</th>
<th>$E_a$ (kcal/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CH$_2$OH + HO$_2$ $\rightarrow$ CQH$_2$OH</td>
<td>2.20E+13</td>
<td>0.0</td>
</tr>
<tr>
<td>-1 CQH$_2$OH $\rightarrow$ CH$_2$OH + HO$_2$</td>
<td>3.56E+16</td>
<td>76.1</td>
</tr>
<tr>
<td>2 CQH$_2$OH $\rightarrow$ CH$_2$OHO $\cdot$ + OH</td>
<td>2.54E+15</td>
<td>45.7</td>
</tr>
<tr>
<td>3 CQH$_2$OH $\rightarrow$ CQ $\cdot$ H$_2$OH + H</td>
<td>3.91E+13</td>
<td>88.2</td>
</tr>
</tbody>
</table>

frequencies (from CPFIT): 1693.6 cm$^{-1}$

Lennard-Jones parameters: $\sigma = 4.49$ Å, $\epsilon/k = 217.1$ K

$k_1$ $A_1$ taken as the average of C $\cdot$ and CCC $\cdot$ + HO$_2$ addition from Tsang 86$^\prime$.  
$k_1$ MR  
$k_2$ $A_2$ from MR with $A_2 = 2.41E+13$, which is the rate constant for addition of HO$_2$ to C$\cdot$CCC $\cdot$, Tsang 88$^\prime$, $E_a_2 = \Delta H_2$  
$k_3$ $A_3$ from MR with $A_3 = 4.0E+13$, based on H addition to C$_2$H$_5$, NIST, and $E_a_3 = \Delta H_3$.  


## C.6 CH₂O → Product

<table>
<thead>
<tr>
<th>reaction</th>
<th>A (s⁻¹ or cm³/(mol·s))</th>
<th>Ea (kcal/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH₂OH → CH₂O + H</td>
<td>2.63E+13</td>
<td>33.480</td>
</tr>
</tbody>
</table>

frequencies (from CPFIT): 1414.5 cm⁻¹
Lennard-Jones parameters: \( \sigma = 4.27 \ \text{Å}, \ \epsilon/k = 454.0 \ \text{K} \)

\( k_1 \) A₁ from MR with A₁ as 1/2 of H addition to C=C and \( \Delta \text{H} = \Delta \text{H} + 2.5 \) kcal/mol, Dean's suggestion for high pressure limit
C.7 $\text{CH}_3\text{O} + \text{CH}_3 \leftrightarrow [\text{CH}_3\text{OCH}_3]^* \rightarrow \text{Product}$

<table>
<thead>
<tr>
<th>Reaction</th>
<th>$A \left( \text{s}^{-1} \text{ or cm}^2/(\text{mol}^*\text{s}) \right)$</th>
<th>$E_a \left( \text{kcal/mol} \right)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 $\text{CH}_3\text{O} + \text{CH}_3 \rightarrow \text{CH}_3\text{OCH}_3$</td>
<td>$1.21\text{E+13}$</td>
<td>0.0</td>
</tr>
<tr>
<td>-1 $\text{CH}_3\text{OCH}_3 \rightarrow \text{CH}_3\text{O} + \text{CH}_3$</td>
<td>$1.05\text{E+17} T^{-0.19e^{-0.00144T}}$</td>
<td>82.5</td>
</tr>
<tr>
<td>2 $\text{CH}_3\text{OCH}_3 \rightarrow \text{CH}_3\text{OCH}_4 + \text{H}$</td>
<td>$2.04\text{E+15}$</td>
<td>97.5</td>
</tr>
<tr>
<td>3 $\text{CH}_3\text{OCH}_3 \rightarrow ^1\text{CH}_2 + \text{CH}_3\text{OH}$</td>
<td>$2.11\text{E+16}$</td>
<td>97.3</td>
</tr>
</tbody>
</table>

Frequencies/degen (from CPFIT): $622 \text{ cm}^{-1}/7.659$, $1789 \text{ cm}^{-1}/9.169$, $3598 \text{ cm}^{-1}/4.172$

Lennard-Jones parameters: $\sigma = 4.31 \text{ Å}$, $\epsilon/\kappa = 395 \text{ K}$


$k_1$ Microscopic Reversibility (MR) with the temperature range of 298 K and 2000 K.

$k_2$ based on $k_2 = 6.40\text{E+13}$.

$k_3$ based on $A_3 = (1.00\text{E+14})^{1/4}$ and $E_a_3 = 1.0 \text{ kcal/mol}$. 
C.8 CH₂O + CH₃ ↔ [CH₃OCH₂]^* → Product

\[
\begin{align*}
\text{CH}_2\text{O} + \text{CH}_3 & \quad \text{Ea = 6.6} \\
\text{CH}_3\text{OCH}_2 & \quad \text{2.0}
\end{align*}
\]

\text{kcal/mol}

<table>
<thead>
<tr>
<th>Reaction</th>
<th>(A) (s⁻¹ or cm³/(mol·s))</th>
<th>(E_a) (kcal/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CH₂O + CH₃ \rightarrow CH₃OCH₂</td>
<td>5.19E+11</td>
<td>6.6</td>
</tr>
<tr>
<td>-1 CH₃OCH₂ \rightarrow CH₂O + CH₃</td>
<td>8.69E+17 T⁻¹.146e⁻0.00164T</td>
<td>13.7</td>
</tr>
</tbody>
</table>

frequencies/degen (from CPFIT): 631 cm⁻¹/7.878, 1899 cm⁻¹/7.530, 3593 cm⁻¹/2.592
Lennard-Jones parameters: \(\sigma = 4.31\) Å, \(\epsilon/k = 395\) K

\(k_1\) based on CH₃ + CO addition from Anastasi et al., *J. Chem. Soc. Faraday Trans. 1*, 878, 2423 (1982).

\(k_1\) Microscopic Reversibility (MR) with the temperature range of 298 K and 2000 K.
### Reaction Rates and Energies

<table>
<thead>
<tr>
<th>Reaction</th>
<th>A (s(^{-1}) or cm(^3)/(mol*s))</th>
<th>Ea (kcal/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (\text{CH}_2\text{OH} + \text{CH}_3\text{O} \rightarrow \text{CH}_3\text{OCH}_2\text{OH})</td>
<td>1.21E+13</td>
<td>0.0</td>
</tr>
<tr>
<td>(-1) (\text{CH}_3\text{OCH}_2\text{OH} \rightarrow \text{CH}_2\text{OH} + \text{CH}_3\text{O})</td>
<td>6.01E+17 T(^{0.66}e^{-0.00077T})</td>
<td>87.8</td>
</tr>
<tr>
<td>2 (\text{CH}_3\text{OCH}_2\text{OH} \rightarrow \text{CH}_3\text{OC} \cdot \text{H(OH)} + \text{H})</td>
<td>5.70E+14</td>
<td>90.6</td>
</tr>
<tr>
<td>3 (\text{CH}_3\text{OCH}_2\text{OH} \rightarrow \text{CH}_3\text{OC} \cdot \text{H}_2 + \text{OH})</td>
<td>4.84E+15</td>
<td>92.8</td>
</tr>
<tr>
<td>4 (\text{CH}_3\text{OCH}_2\text{OH} \rightarrow \text{CH}_3\text{OC} \cdot + \text{H})</td>
<td>5.34E+14</td>
<td>100.5</td>
</tr>
</tbody>
</table>

**frequencies/degen (from CPFIT):** 489 cm\(^{-1}\)/8.260, 1626 cm\(^{-1}\)/10.330, 3600 cm\(^{-1}\)/5.410

**Lennard-Jones parameters:** \(\sigma = 4.90 \text{ \AA}, \epsilon/k = 415 \text{ K}\)

\(k_1\) from \(\text{CH}_3 + \text{CH}_3\text{O}\), Tsang et al., *J. Phys. Chem.*, 15, 1087 (1986).

\(k_{-1}\) Microscopic Reversibility (MR) with the temperature range of 298 K and 2000 K.

\(k_2\) based on \(k_{-2} = 1.00E+14\).

\(k_3\) based on \(k_3 = 2.11E+13\) from NIST fitting for \(\text{CH}_3 + \text{OH}\).

\(k_4\) based on \(k_4 = 2.00E+14\).
C10 $\text{CH}_3\text{O} + \text{CH}_2\text{O} \leftrightarrow [\text{CH}_3\text{OCH}_2\text{O}]^* \rightarrow \text{Product}$

<table>
<thead>
<tr>
<th>Reaction</th>
<th>$A$ ($s^{-1}$ or cm$^{-3}$(mol$^*$s))</th>
<th>$E_a$ (kcal/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\text{CH}_3\text{O} + \text{CH}_2\text{O} \rightarrow \text{CH}_3\text{OCH}_2\text{O}$</td>
<td>$5.19\times10^{11}$</td>
</tr>
<tr>
<td>-1</td>
<td>$\text{CH}_3\text{OCH}_2\text{O} \rightarrow \text{CH}_3\text{O} + \text{CH}_2\text{O}$</td>
<td>$1.35\times10^{24}$ $T^{-2.86e^{-0.00002T}}$</td>
</tr>
<tr>
<td>2</td>
<td>$\text{CH}_3\text{OCH}_2\text{O} \rightarrow \text{CH}_3\text{OC} \cdot \text{H}_2 + \text{O}$</td>
<td>$4.83\times10^{15}$</td>
</tr>
<tr>
<td>3</td>
<td>$\text{CH}_3\text{OCH}_2\text{O} \rightarrow \text{C} \cdot \text{H}_2\text{OCH}_2\text{OH}$</td>
<td>$2.08\times10^{9}$ $T^{1.0}$</td>
</tr>
<tr>
<td>-3</td>
<td>$\text{C} \cdot \text{H}_2\text{OCH}_2\text{OH} \rightarrow \text{CH}_3\text{OCH}_2\text{O} \cdot$</td>
<td>$2.74\times10^{11}$ $T^{-0.14e^{0.0008T}}$</td>
</tr>
<tr>
<td>4</td>
<td>$\text{C} \cdot \text{H}_2\text{OCH}_2\text{OH} \rightarrow \text{CH}_2\text{OH} + \text{CH}_3\text{O}$</td>
<td>$4.31\times10^{12}$</td>
</tr>
</tbody>
</table>

Frequencies/degen (from CPFIT): 558 cm$^{-1}$/7.863, 1731 cm$^{-1}$/9.557, 3599 cm$^{-1}$/3.580

Lennard-Jones parameters: $\sigma = 4.61$ Å, $\epsilon/k = 415$ K


$k_2$ MR from $k_2 = 7.00E+13$ taken as CH$_3$ + O, Dean, 1987.

$k_3$ $A_3 = (\text{ekT}/\hbar)[\exp(\Delta S^\theta/\text{R}^*](3) = (10^{10.71T})(10^{A_{\text{Ea}}/4.6})(3), \Delta S^\theta = -8.6$ (loss of 2 rotors); $E_a = $ ring strain (5.9) + $E_{\text{abstraction}}$ (10).

$k_3$ MR with the temperature range of 298 K and 2000 K.

$k_4$ MR, $A_{\text{Ea}} = 1.06E+11$ taken as one half of CH$_3$ addition to C$_2$H$_4$ and $E_{\text{Ea}} = 7.4$ from Baulch et al., 1992.
C.11 COC· + CH₂O ↔ [COCOC·]^* → Product

\[
\begin{align*}
\text{COC.} + \text{CH}_2\text{O} &\quad \text{Ea} = 7.4 \\
\text{COCOC.} &\quad -36.4 \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>Reaction</th>
<th>( A ) ( \text{cm}^3/(\text{mol}^*\text{s}) )</th>
<th>( \text{Ea (kcal/mol)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  COC· + CH₂O → COCOC·</td>
<td>1.06E+11</td>
<td>7.4</td>
</tr>
<tr>
<td>-1 COCOC· → COC· + CH₂O</td>
<td>1.91E+20 ( T^{2.78e^{-0.00002T}} )</td>
<td>19.1</td>
</tr>
</tbody>
</table>

frequencies/degen (from CPFIT): 631 cm\(^{-1}\)/7.878, 1899 cm\(^{-1}\)/7.530, 3593 cm\(^{-1}\)/2.592

Lennard-Jones parameters: \( \sigma = 4.31 \) Å, \( \epsilon/k = 395 \) K

\( k_1 \)  \( A_1 \) taken as one half of CH₃ addition to \( C_2H_4 \) and \( \text{Ea}_1 = 7.4 \) from Baulch et al., 1992.

\( k_{-1} \) Microscopic Reversibility (MR) with the temperature range of 298 K and 2000 K.
C.12 COCO· + CH₃ → ↔ [COCOC]* → Product

<table>
<thead>
<tr>
<th>Reaction</th>
<th>A (s⁻¹ or cm³/(mol*sec))</th>
<th>Ea (kcal/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 COCO· + CH₃ → COCOC</td>
<td>1.21E+13</td>
<td>0.0</td>
</tr>
<tr>
<td>-1 COCOC → COCO· + CH₃</td>
<td>5.81E+15</td>
<td>82.5</td>
</tr>
<tr>
<td>2 COCOC → COC· + CH₃O</td>
<td>5.65E+15</td>
<td>87.8</td>
</tr>
<tr>
<td>3 COCOC → COC·OC + H</td>
<td>3.06E+15</td>
<td>94.4</td>
</tr>
<tr>
<td>4 COCOC → COCOC· + H</td>
<td>2.00E+16</td>
<td>97.6</td>
</tr>
</tbody>
</table>

frequencies/degen (from CPFIT): 566 cm⁻¹/14.05, 1802 cm⁻¹/13.09, 3598 cm⁻¹/5.86

Lennard-Jones parameters: \( \sigma = 5.28 \text{ Å}, \epsilon/k = 379 \text{ K} \)

\( k_1 \) Taken as for CH₃O + CH₃ → COC from Tsang et al., J. Phys. Chem., 15, 1087 (1986).

\( k_{-1} \) Microscopic Reversibility (MR) with the temperature range of 298 K and 2000 K.

\( k_2 \) MR with \( k_2 = 1.21E+13 \) as CH₃O + CH₃ → COC from Tsang et al., J. Phys. Chem., 15, 1087 (1986).

\( k_3 \) based on \( k_3 = 1.00E+14 \).

\( k_4 \) based on \( k_4 = 6.40E+13 \).
C.13 COC· + CH₃O → ↔ [COCOC]* → Product

\[
\begin{align*}
\text{COCOC + H} & \quad 15.8 \\
\text{COCOC + CH₃} & \quad 0.7 \\
\end{align*}
\]

\[
\text{COC} + \text{CH₃O} \quad 6.0
\]

\[
\text{COC} + \text{CH₃O} \quad -81.8
\]

\[
\text{kcal/mol}
\]

<table>
<thead>
<tr>
<th>Reaction</th>
<th>A (s⁻¹ or cm³/(mol* s))</th>
<th>Ea (kcal/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>COC· + CH₃O → COCOC</td>
<td>1.21E+13</td>
</tr>
<tr>
<td>-1</td>
<td>COCOC → COC· + CH₃O</td>
<td>5.65E+15</td>
</tr>
<tr>
<td>2</td>
<td>COCOC → COCO· + CH₃</td>
<td>5.81E+15</td>
</tr>
<tr>
<td>3</td>
<td>COCOC → COC· OC + H</td>
<td>3.06E+15</td>
</tr>
<tr>
<td>4</td>
<td>COCOC → COCOC· + H</td>
<td>2.00E+16</td>
</tr>
</tbody>
</table>

frequencies/degen (from CPFIT): 566 cm⁻¹/14.05, 1802 cm⁻¹/13.09, 3598 cm⁻¹/5.86

Lennard-Jones parameters: \( \sigma = 5.28 \ \text{Å}, \epsilon/k_B = 379 \ \text{K} \)

\( k_1 \) Taken as CH₂O + CH₃ → COC from Tsang et al., *J. Phys. Chem.*, 15, 1087 (1986).

\( k_2 \) Microscopic Reversibility (MR) with the temperature range of 298 K and 2000 K.

\( k_3 \) MR with \( k_2 = 1.21E+13 \) as CH₂O + CH₃ → COC from Tsang et al., *J. Phys. Chem.*, 15, 1087 (1986).

\( k_4 \) based on \( k_4 = 6.40E+13 \).
APPENDIX D

FORTRAN PROGRAMS

D.1 THERMCAL

PROGRAM THERMCAL

thermodynamic properties of species

IMPLICIT NONE

include 'thm_par.fi'
include 'thm_cfg.fi'

CHARACTER comment(Max_Group)*70,
2 specy_name*80, specy_formula*40,
3 specy_group*80,
4 specy_element(Max_Element)*4,
5 specy_comment*70, specy_phase, line*200, KEY(5)*3,
6 SUB(20)*100, UPCASE*4, str*100, qty*100

INTEGER specy_group_qty, specy_symmetry(specy_len,
1  specy_element_qty(Max_Element),Nloop,ILASCH,
2 ILEN, IPPLEN, NSUB, NKEY, Isymmetry(5), NVAL, Ierr,
3 I, Istr, Igroup, Ifirch, Islash, Igty(5), Ifound, M,
4 Irotor

INTEGER*2 year, month, day

REAL*8 specy_Hf, specy_S, specy_Cp300, specy_Cp400, specy_Cp500,
1 specy_Cp600, specy_Cp800, specy_Cp1000, specy_Cp1500,
2 specy_rotor, rotor(5)

LOGICAL therm, nasa, init_err, io_err, ErrCpl500, elem_err

DATA KEY/'REF','XXX','XXX','END','!'/

Initialize variables

Call Initialization subroutines, 02/95 ING

CALL TITLE
DO 10 M = 1, max_element
specy_element(M) = ' '
specy_element_qty(M) = 0
10 continue
CALL THERM_IO(io_err)
if (io_err) GO TO 6000
CALL THERM_INIT(init_err)
if (init_err) GO TO 6000
CALL GETDAT(year, month, day)
year = year - 1900
write(LOG,10)month, day, year
10 format('Thermal log file, date : ','I2', '/', 'I2', '/', 'I2')

242
C  write(lst,*),'THERMO'
write(lst,50)
50 format(' SPECIES Hf S Cp 300 400 500 60 10 800 1000 1500 DATE ELEMENTS')
C  CALL GET_GROUPS

C  READ (INP,'(A)',END=5000) LINE
100 CONTINUE
LINE = ' ',
READ (INP,'(A)',END=5000) LINE
105 CONTINUE
ILEN = IPPLEN(LINE)
IF (ILEN .EQ. 0) GO TO 100
C  CALL CKISUB (LINE(:ILEN), SUB, NSUB)
C  IS THERE A KEYWORD?
C  CALL CKCOMP ( UPCASE(SUB(1), 1) , KEY, 5, NKEY)
IF (NKEY.EQ.5) GO TO 100
CALL CKCOMP ( UPCASE(SUB(1), 3) , KEY, 4, NKEY)
IF (NKEY.EQ.4) GO TO 5000
C  IF (NKEY .GT. 0) ITASK = 0
C  IF (NKEY.EQ.1) THEN
READ (INP,'(A)',END=5000) LINE
ILEN = IPPLEN(LINE)
WRITE(LST,120) LINE(2:ILEN)
120 FORMAT(IX,A)
C  WRITE(LST,*)LINE
C  ELSE
C   write(*,*)' group (1-3) = ',group(1),' ',group(2),' ',group(3)
specy_name = SUB(1)
specy_formula = SUB(2)
C  specy_len = IPPLEN(specy_formula)
DO 150 M = 1, max_element
   specy_element(M) = ' 1
   specy_element_qty(M) = 0
150 continue
C  WRITE(*,*)specy_len,specy_formula(:specy_len)
call find_elem(specy_formula,specy_len,specy_element,
1   specy_element_qty,element,element_qty,
2   elem_err,LOG)
IF (elem_err) THEN
   lien = IPPLEN(specy_name)
   write(LOG,160)specy_name(:lien)
160 format(lx,' for specy " ',A,' "')
   write(*,170)specy_name(:lien)
170 format(lx,' specy " ',A,' " ERROR !!!')
READ (INP,'(A)',END=5000) LINE
GO TO 100
ENDIF
C  call IPPARI (SUB(3), -1, 1, Isymmetry, nval, ierr, LOG)
IF (ierr.NE. 0) THEN
   lien = IPPLEN(specy_name)
   write(LOG,310)specy_name(:lien)
310 FORMAT(' " ',A," input symmetry ERROR !!!')
   write(*,320)specy_name(:lien)
320 FORMAT(' " ',A," input symmetry ERROR !!!')
READ (INP,'(A)',END=5000) LINE
GO TO 6000
ENDIF
specy_symmetry = Isymmetry(1)
call IPPARR (SUB(4), -1, 1, rotor, nval, Ierr, LOG)
IF (Ierr.NE. 0) THEN
  Ilen = IPPLEN(specy_name)
  WRITE(LOG,330)specy_name(:Ilen)
  FORMAT(1"input rotor ERROR!!!")
  WRITE(*,340)specy_name(:Ilen)
  READ (INP,'(A)',END=5000) LINE
  GO TO 6000
ENDIF
specy_rotor = rotor(1)
CONTINUE
READ (INP,'(A)',END=5000) LINE
ILEN = IPPLEN(LINE)
IF (ILEN .EQ. 0) GO TO 400
CALL CKISUB (LINE(:ILEN), SUB, NSUB)
C
specy_Hf = 0.0
specy_S = 0.0
specy_Cp300 = 0.0
specy_Cp400 = 0.0
specy_Cp500 = 0.0
specy_Cp600 = 0.0
specy_Cp800 = 0.0
specy_Cp1000 = 0.0
specy_Cp1500 = 0.0
C
Nloop = INT(NSUB/2.0)
IF (MOD(NSUB,2).NE. 0) THEN
  WRITE(*,410)specy_name
  FORMAT(1Unbalanced groups and quantities for specy : ,A)
  WRITE(LOG,420)specy_name
  FORMAT(1Unbalanced groups and quantities for specy : ,A)
  GO TO 100
ENDIF
ErrCpi500 = .FALSE.
DO 500 I = 1, Nloop
  C
  str = SUB(I)
  Istr = LEN(str)
C
  Igroup = IFIRCH(str)
  Islash = INDEX(str,'*')
  specy_group = str(Igroup:Islash-1)
  qty = str(Islash+1:Istr)
  str = SUB(I*2-1)
  Istr = LEN(str)
  Igroup = IFIRCH(str)
  Islash = INDEX(str,')')
  if (Islash.eq.0) Islash = Istr
  specy_group = str(Igroup:Islash-1)
  qty = SUB(I*2)
  call IPPARI (qty, -1, 1, Iqty, nval, Ierr, LOG)
  specy_group_qty = Iqty(1)
  IF (Ierr.NE. 0) THEN
    WRITE(LOG,421)specy_formula(:specy_len),
               str(Igroup:Islash-1)
    FORMAT(1formula ,",A","group ",
               A,""quantity ERROR !!!")
    WRITE(*,422)specy_formula(:specy_len),
               str(Igroup:Islash-1)
    FORMAT(1formula ,",A","group ",
               A,""quantity ERROR !!!")
    GO TO 100
  ENDIF
ENDIF
CALL CKCOMP (specy_group, group, group_total, Ifound)
C write(*,*) specy_group, group_total
C write(*,*) (group(1), group(150), group(350))
IF (Ifound.EQ. 0) THEN
  ILEN = ILASCH(specy_group)
  WRITE(*,430) specy_group(:ILEN), specy_name
  FORMAT(' Group ''',A,'''' not found for specy ''',A,''''')
  WRITE(LOG, 440) specy_group(:ILEN), specy_name
  FORMAT(' Group ''',A,'''' not found for specy ''',A,''''')
  GO TO 100
ENDIF
C write(*,*)' specy_group = ', specy_group
C write(*,*)' group_total = ', group_total
C write(*,*)' Ifound = ', Ifound
specy_Hf + group_Hf(Ifound)
specy_S + group_S(Ifound)
specy_Cp300 + group_Cp300(Ifound)
specy_Cp400 + group_Cp400(Ifound)
specy_Cp500 + group_Cp500(Ifound)
specy_Cp600 + group_Cp600(Ifound)
specy_Cp800 + group_Cp800(Ifound)
specy_Cp1000 = specy_Cp1000 + group_Cp1000(Ifound)
specy_Cp1500 = specy_Cp1500 + group_Cp1500(Ifound)
specy_Cp1500 = specy_Cp1500 + group_Cp1500(Ifound)

IF (group_Cp1500(Ifound).EQ.0.0) ErrCp1500 = .true.
  write(1st, *) specy_group, ' ', group_Cp1500(Ifound),
     ' ', specy_group_qty
CONTINUE
ENDF
C
C CONTINUE
5000 IF (ErrCp1500) specy_Cp1500 = 0.0
    specy_S = specy_S - 1.98717 * ALOG(specy_symmetry*1.0)
Irotor = INT(specy_rotor)
write(LST,600) specy_name, specy_Hf, specy_S, specy_Cp300,
  specy_Cp400, specy_Cp500, specy_Cp600,
  specy_Cp800, specy_Cp1000, specy_Cp1500,
  month, day, year,
  specy_element(1), specy_element_qty(1),
  specy_element(2), specy_element_qty(2),
  specy_element(3), specy_element_qty(3),
  specy_element(4), specy_element_qty(4),
  Irotor
6000 format( ',A9,F8.2, ',F8.2, ',F8.2,1x,12, '/ ',12, '/ ',12,
         2x,'ThmCal',2x,4(A2,I3), ',G',12)
ENDF
C
C go to 100
5000 write(*,*)'... End of calculations...' c
WRITE(LST,5100)
c 5100 FORMAT(' END')
c write(*,*)' List file created as: ', File_lst
write(*,*)'... error in input file... ', File_inp
go to 7000
6000 write(*,*)'... error in input file... ', File_inp
close (log)
close (lst)
close (inp)
end
SUBROUTINE therm_init(init_err)
implicit none
include 'thm_par.fi'
include 'thm_cfg.fi'
character line*100, sub(20)*100, key(4)*5, UPCASE*4, KEY_WD*4,
1 group_n*20
integer cfg, ilen, nsub, nkey, nfile, nelem, nhelp, nkeys,
1 Ifile(5), Ielem(5), Ihelp(5), Ikeys(5), nval, Igroup(5),
2 Istart, Istr, Ierr, group_idx, IPPLEN, I, J, ngroup,
3 Ifirch, Icoma, Ierr2, Ierr3
real*8 Igp(9)
logical init_err, BD

data cfg/35/, key/'F I L E ','ELEM', 'HELP','K E Y S '/

OPEN(cfg,FILE='therm.cfg',STATUS='OLD ', ERR=30)
GO TO 50
30 CONTINUE
WRITE(*,40)
40 FORMAT( '  Configuration file THERM.CFG is not in the working direct
ory !!!')
init_err = .true.
return
50 CONTINUE
init_err = .false.
100 CONTINUE
LINE = ' '
READ (CFG,'(A)',END=5000) LINE
105 CONTINUE
ILEN = IPPLEN(LINE)
IF (ILEN .EQ. 0) GO TO 100
CALL CKISUB (LINE(:ILEN),  SUB, NSUB)
KEY_WD = SUB(1)(2:5)
CALL CKCOMP ( UPCASE(KEY_WD, 4) , KEY, 4, NKEY)
write(*,*)' key = ',key_wd,'  nkey = ',nkey
C IF (NKEY .GT. 0) ITASK = 0
C IF (NKEY.EQ.1) THEN
  call IPPARI (SUB(2), -1, 1, Ifile, nval, Ierr, LOG)
  write(*,*)'..group files = ',Ifile(l), '  Ierr = ',Ierr
  IF (Ierr.NE. 0) GO TO 6000
  nfile = Ifile (1)
group_total = 0
  do 300 1 = 1 , nfile
    READ (CFG, '  (A) ',END=5000) LINE
    ILEN = IPPLEN(LINE)
    IF (ILEN .EQ. 0) GO TO 100
    CALL CKISUB (LINE(:ILEN), SUB, NSUB)
    BD = .false.
    if (UPCASE(SUB(1),2).EQ.'BD')  BD = .T R U E .
    OPEN (100+I,FILE = SUB(1),STATUS='OLD ',ERR=130)
    C READ (100+I,'(A)',END=5000) LINE
    Istr = IPPLEN(sub(1))
    Ifirch = IFIRCH(sub(1))
    Icoma = INDEX(sub(1),'')
  C ADDED FOR RADICAL CORRECTION
    IF (UPCASE(SUB(1),2).EQ.'BD') BD=.TRUE.
    OPEN(100+I,FILE=sub(1),STATUS='OLD',ERR=130)
    C READ (100+I,'(A)',END=5000) LINE
    Istr = IPPLEN(sub(1))
    Ifirch = IFIRCH(sub(1))
    Icoma = INDEX(sub(1),'')
  C ADDED FOR RADICAL CORRECTION
    IF (UPCASE(SUB(1),2).EQ.'BD') BD=.TRUE.
    OPEN(100+I,FILE=sub(1),STATUS='OLD',ERR=130)
    C READ (100+I,'(A)',END=5000) LINE
    Istr = IPPLEN(sub(1))
    Ifirch = IFIRCH(sub(1))
    Icoma = INDEX(sub(1),'')
  C ADDED FOR RADICAL CORRECTION
    IF (UPCASE(SUB(1),2).EQ.'BD') BD=.TRUE.
    OPEN(100+I,FILE=sub(1),STATUS='OLD',ERR=130)
    C READ (100+I,'(A)',END=5000) LINE
    Istr = IPPLEN(sub(1))
    Ifirch = IFIRCH(sub(1))
    Icoma = INDEX(sub(1),'')

IF (Icoma.EQ.0) THEN
  group_n = line(Istart:Istr)
ELSE
  group_n = line(Istart:Icoma-1)
ENDIF

call IPPARI (group_n, -1, 1, Igroup, nval, ierr, LOG)
IF (ierr.NE.0 .AND. ierr2.NE.0 .AND. ierr3.NE.0) THEN
  GO TO 6000
ELSEIF (ierr.NE.0 .AND. ierr2.NE.0) THEN
  ierr3 = 1
  GROUP = 
  GO TO 110
ELSEIF (ierr.NE.0) THEN
  ierr2 = 1
  GROUP = 
  GO TO 110
ENDIF
ngroup = Igroup(1)

C

C

group_idx = Igroup(group_total + 1)
C
write(*,'(a)') ('group', ', group_total, ') = ', group(group_total)
group_total = ngroup + group_total
C
write(*,'(a)') ('', ', group_total, ') = ', group_total
C
DO 120 J = 1, ngroup
  READ (100+1, '(a)') LINE
  ILEN = IPPLEN(LINE)
  IF (ILEN .EQ. 0) GO TO 6000
  CALL CKISUB (LINE(:ILEN), SUB, NSUB)
  Istart = IPPLEN(SUB(1))
  Icoma = INDEX(SUB(1), '1')
  IF (Icoma.EQ.0) THEN
    group(group_idx) = line(Istart:Istr)
  ELSE
    group(group_idx) = line(Istart:Icoma-1)
  ENDIF
  call IPPARR (line(Istr+2:79), -1, 11, Igp, nval, ierr, LOG)
C
  CORRECT THE RADICAL OF LOSING ONE H ATOM FOR BD GROUPS
  IF (BD) THEN
    group_Hf(group_idx) = Igp(1) - 52.1
  ELSE
    group_Hf(group_idx) = Igp(1)
  ENDIF
  group_S(group_idx) = Igp(2)
  group_Cp300(group_idx) = Igp(3)
  group_Cp400(group_idx) = Igp(4)
  group_Cp500(group_idx) = Igp(5)
  group_Cp600(group_idx) = Igp(6)
  group_Cp800(group_idx) = Igp(7)
  group_Cp1000(group_idx) = Igp(8)
  IF (nval.NE.8) group_Cp1500(group_idx) = Igp(9)
  write(1st,'(a)') group(group_idx), group_idx + 1
C
120 CONTINUE
  GO TO 200
130 CONTINUE
  WRITE(*,140) SUB(1)
140 FORMAT(' Can't find group file : ', A20)
  init_err = .true.
  return
200 CLOSE(100+1)
300 CONTINUE
  GO TO 100
ELSEIF (NKEY.EQ.2) THEN
  call IPPARI (SUB(2), -1, 1, Ielem, nval, ierr, LOG)
IF (Ierr.NE. 0) GO TO 6000

`element_qty = Ielem(1)`

WRITE(*,*) 'elera_qty = ', `element_qty`

group_total = 0

do 500 I = 1, `element_qty`

`continue`

READ (CFG,' (A)', END=5000) LINE

`ILEN = IPPLEN(LINE)`

IF (ILEN .EQ. 0) GO TO 100

CALL CKISUB (LINE(1:ILEN), SUB, NSUB)

`ILEN = IPPLEN(SUB(1))`

`element(I) = UPCASE(SUB(1), ILEN)`

WRITE(*,*) 'I = ', `I', 'element = ', `element(I)`

`500 continue`

go to 100

ELSEIF (NKEY.EQ.3) THEN

`600 continue`

go to 5000

ELSEIF (NKEY.EQ.4) THEN

`700 continue`

go to 5000

ELSE

WRITE(*,*)'Errors in configuration file THERM.CFG !!!'

`init_err = .true.`

ENDIF

5000 continue

go to 7000

6000 WRITE(*,*)'Errors in configuration file THERM.CFG !!!'

`init_err = .true.`

7000 continue

close (cfg)

return

end

C-----------------------------C

SUBROUTINE find_elem(specy_formula, specy_len, specy_element, 1 specy_element_qty, element, element_qty, 2 elem_err, LOG)

IMPLICIT NONE

C

CHARACTER(*) specy_formula, specy_element(*), element(*)

CHARACTER numeric(10), char(26), elem_ch1, elem_ch2, qty_ch*3, 1 paren(2), UPCASE*4

INTEGER specy_element_qty(*), LOG, I, J, K, Ifirch, Islash, 1 Ilen, NKEY, specy_len, Index2, sym(40), 2 qty(2), Ierr, element_qty, IPPLEN, specy_n(20), 3 specy_n_temp(20), nval, M

C

LOGICAL elem_err, paren_left, paren_right

C

DATA paren/','/', 1 sym/40*0/

C

elem_err = .false.

Ilen = IPPLEN(specy_formula)

IF (Ilen.EQ.0) THEN

elem_err = .true.

write(LOG,*)' ...ERROR... specy formula did not specify.'

return

ENDIF

C

DO 30 I = 1, 20
specy_n(I) = 0
specy_n_temp(I) = 0
30 CONTINUE
DO 100 I = 1, specy_len
  K = ICHAR(specy_formula(I:I))
  paren_left = .false.
  paren_right = .false.
  IF (K.GE.48.AND. K.LE.57) THEN
    sym(I) = 2
  ELSEIF (K.GE.65.AND. K.LE.90) THEN
    sym(I) = 1
  ELSEIF (K.GE.97.AND. K.LE.122) THEN
    specy_formula(I:I) = UPCASE(specy_formula(I:I),1)
    sym(I) = 1
  ELSEIF (K.EQ.40.) THEN
    sym(I) = 3
  ELSEIF (K.EQ.41) THEN
    sym(I) = 4
  ELSE
    elem_err = .true.
    write(LOG,50)specy_formula(:Ilen),specy_formula(I:I)
    50 format(lx, specy_formula ',A',' contains unknown',
    1 ' character ',A,'')
    return
 ENDIF
100 CONTINUE
Index2 = 1
200 CONTINUE
  write(*,*)' sym = ',sym
  Index2 = ',sym
  IF (sym(Index2).EQ.1) THEN
    CALL sub_elem(Index2,specy_formula,specy_n_temp,sym,
    1 specy_len,element,element_qty,elem_err,LOG)
    GO TO 300
  ELSEIF (sym(Index2) .EQ.3) THEN
    paren_left = .true.
    Index2 = Index2 + 1
    qty(I) = 1
    EXPECTED_ROW_INDEX=69
    IF (sym(Index2).EQ.2) THEN
      qty_ch = ' '
      qty_ch = specy_formula(Index2:Index2+1)
      CALL IPPARI(qty_ch, -1, 1, qty, nval, Ierr, LOG)
      Index2 = Index2 + 2
    ELSE
      qty_ch = ' '
      qty_ch = specy_formula(Index2:Index2)
      CALL IPPARI(qty_ch, -1, 1, qty, nval, Ierr, LOG)
      Index2 = Index2 + 1
    ENDIF
  ELSEIF (sym(Index2).EQ.4) THEN
    paren_left = .true.
    Index2 = Index2 + 1
    qty(I) = 1
    EXPECTED_ROW_INDEX=69
    IF (sym(Index2+1).EQ.2) THEN
      qty_ch = ' '
      CALL IPPARI(qty_ch, -1, 1, qty, nval, Ierr, LOG)
      Index2 = Index2 + 2
    ELSE
      qty_ch = ' '
      CALL IPPARI(qty_ch, -1, 1, qty, nval, Ierr, LOG)
      Index2 = Index2 + 1
    ENDIF
  ELSEIF (sym(Index2).EQ.5) THEN
    paren_left = .true.
    Index2 = Index2 + 1
    qty(I) = 1
    EXPECTED_ROW_INDEX=69
    IF (sym(Index2+1).EQ.2) THEN
      qty_ch = ' '
      CALL IPPARI(qty_ch, -1, 1, qty, nval, Ierr, LOG)
      Index2 = Index2 + 2
    ELSE
      qty_ch = ' '
      CALL IPPARI(qty_ch, -1, 1, qty, nval, Ierr, LOG)
      Index2 = Index2 + 1
    ENDIF
  ELSEIF (sym(Index2).EQ.6) THEN
    paren_left = .true.
    Index2 = Index2 + 1
    qty(I) = 1
    EXPECTED_ROW_INDEX=69
    IF (sym(Index2+1).EQ.2) THEN
      qty_ch = ' '
      CALL IPPARI(qty_ch, -1, 1, qty, nval, Ierr, LOG)
      Index2 = Index2 + 2
    ELSE
      qty_ch = ' '
      CALL IPPARI(qty_ch, -1, 1, qty, nval, Ierr, LOG)
      Index2 = Index2 + 1
    ENDIF
  ENDIF
ENDIF
ENDIF
DO 220 J = 1, element_qty
  specy_n(J) = specy_n(J) + qty(I)*specy_n_temp(J)
220 CONTINUE
ELSE
    elem_err = .true.
    write(LOG,240) specy_formula(:Ilen)
    format(ix,' specy formula ",",A," ... error ...")
    return
ENDIF
ELSE
    elem_err = .true.
    write(LOG,260) specy_formula(:Ilen)
    format(ix,' specy formula ",",A," ... error ...")
    return
ENDIF
300 CONTINUE

    IF (Index2.GT.Ilen) return
    IF (Index2.LE.Ilen) GO TO 200
    K = 0
    DO 400 J = 1, element_qty
        IF (specy_n(J).GT.0) K = K + 1
    400 CONTINUE
    IF (K.GT.4) THEN
        elem_err = .true.
        write(LOG,500) specy_formula(:Ilen)
        format(lx,' specy formula ",",A, more than 4 elements')
    ENDIF
    IF (specy_n(1).NE.0) THEN
        M = 1
        DO 600 J = 1, element_qty
            IF (specy_n(J).NE.0) THEN
                specy_element(M) = element(J)
                specy_element_qty(M) = specy_n(J)
                M = M + 1
            ENDIF
        600 CONTINUE
    ELSEIF (specy_n(2).NE.0.AND.K.LT.3) THEN
        M = 3
        specy_element(1) = element(1)
        specy_element_qty(1) = 0
        specy_element(2) = element(2)
        specy_element_qty(2) = 0
        DO 700 J = 3, element_qty
            IF (specy_n(J).NE.0) THEN
                specy_element(M) = element(J)
                specy_element_qty(M) = specy_n(J)
                M = M + 1
            ENDIF
        700 CONTINUE
    ELSEIF (specy_n(2).NE.0.AND.K.LT.4) THEN
        M = 2
        specy_element(1) = element(2)
        specy_element_qty(1) = 0
        DO 800 J = 2, element_qty
            IF (specy_n(J).NE.0) THEN
                specy_element(M) = element(J)
                specy_element_qty(M) = specy_n(J)
                M = M + 1
            ENDIF
        800 CONTINUE
    ELSE
        M = 1
        DO 900 J = 1, element_qty
            IF (specy_n(J).NE.0) THEN
                specy_element(M) = element(J)
                specy_element_qty(M) = specy_n(J)
                M = M + 1
            ENDIF
        900 CONTINUE
SUBROUTINE sub_elem(Index2, specy_formula, specy_n, sym, specy_len, element, element_qty, elem_err, LOG)

IMPLICIT NONE

CHARACTER(*) specy_formula, element(*)

CHARACTER elem_ch, elem_ch2*2, qty_ch*3

INTEGER LOG, I, J, K, Ifirch, Islash,
  Ilen, NKEY, specy_len, Index2, sym(40),
  qty(2), ierr, M, element_qty, specy_n(*), elem_found,
  nval

LOGICAL elem_err, paren_left, paren_right

elem_found = 0

IF (sym(Index2).NE.1) THEN
  elem_err = .true.
  write(LOG,100)specy_formula(:Ilen)

100 format(lx,specy formula ' , A , . . . error ...')
  return
ENDIF

elem_ch = specy_formula(Index2:Index2)
CALL CKCOMP ( elem_ch, element, element_qty, NKEY)

c write(*,*)elem_ch,element_qty,NKEY

do 150 i = 1, element_qty

150 continue

IF (NKEY.EQ.0) THEN
  elem_err = .true.
  write(LOG,200)specy_formula(:Ilen)

200 format(lx,specy formula ' , A , . . . error ...')
  return
ENDIF

elem_found = NKEY

qty(1) = 1

IF (Index2+1.GT.specy_len) THEN
  specy_n(elem_found) = specy_n(elem_found) + qty(1)
  Index2 = Index2 + 1
  return
ENDIF

IF (sym(Index2+1).EQ.3.OR.sym(Index2+1).EQ.4) THEN
  specy_n(elem_found) = specy_n(elem_found) + 1
  Index2 = Index2 + 1
  return
ELSEIF (sym(Index2+1).EQ.2) THEN

300 continue

IF ((Index2+M+1).LE.specy_len.AND.sym(Index2+M+1).EQ.2) THEN
  M = M + 1
  CONTINUE
ENDIF

qty_ch =rega
  qty_ch = specy_formula(Index2+1:Index2+M)
CALL IPPARI(qty_ch, -1, 1, qty, nval, ierr, LOG)

specy_n(elem_found) = specy_n(elem_found) + qty(1)
Index2 = Index2 + M + 1

c write(*,*) Index2 = ',Index2,' qty = ', qty(1)
  return
ELSEIF (sym(Index2+1).EQ.1) THEN
elem_ch2 = specy_formula(Index2:Index2+1)
CALL CKCOMP (elem_ch2, element, element_qty, NKEY)
IF (NKEY.EQ.0) THEN
    specy_n(elem_found) = specy_n(elem_found) + 1
    Index2 = Index2 + 1
    return
ELSE
    elem_found = NKEY
    Index2 = Index2 + 2
ENDIF
M = 0
CONTINUE
IF ((Index2+M+1).LE.specy_len.AND.sym(Index2+M).EQ.2) THEN
    M = M + 1
    GO TO 400
ENDIF
IF (M.GT.0) THEN
    qty_ch = " 
    qty_ch = specy_formula(Index2:Index2+M-1)
    CALL IPPARI(qty_ch, -1, 1, qty, nval, Ierr, LOG)
    specy_n(elem_found) = specy_n(elem_found) + qty(1)
    Index2 = Index2 + M
    return
ELSE
    specy_n(elem_found) = specy_n(elem_found) + 1
    return
ENDIF
specy_n(elem_found) = specy_n(elem_found) + 1
Index2 = Index2 + 1
ENDIF
return
end

SUBROUTINE CKISUB (LINE, SUB, NSUB)
Generates an array of CHAR(*) substrings from a CHAR(*) string, using blanks or tabs as delimiters
Input: LINE - a CHAR(*) line
Output: SUB - a CHAR(*) array of substrings
NSUB - number of substrings found
A '!' will comment out a line, or remainder of the line.
F. Rupley, Div. 8245, 5/15/86
------------------------------------------------------------------
C
SUBROUTINE CKISUB (LINE, SUB, NSUB)
C Generates an array of CHAR(*) substrings from a CHAR(*) string,
C using blanks or tabs as delimiters
C Input: LINE - a CHAR(*) line
C Output: SUB - a CHAR(*) array of substrings
C NSUB - number of substrings found
C A '!' will comment out a line, or remainder of the line.
C F. Rupley, Div. 8245, 5/15/86
C------------------------------------------------------------------
C*****precision > double
C 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
CHARACTER(*) SUB(*), LINE
NSUB = 0
C
DO 5 N = 1, LEN(LINE)
    IF (ICHAR(LINE(N:N)) .EQ. 9) LINE(N:N) = ' '
5 CONTINUE
C
IF (IPPLEN(LINE) .LE. 0) RETURN
C
ILEN = ILASCH(LINE)
C
NSTART = IFIRCH(LINE)
10 CONTINUE
ISTART = NSTART
NSUB = NSUB + 1
SUB(NSUB) = ' '

DO 100 I = ISTART, ILEN
  ILAST = INDEX(LINE(ISTART:],', ') - 1
  IF (ILAST .GT. 0) THEN
    ILAST = ISTART + ILAST - 1
  ELSE
    ILAST = ILEN
  ENDIF
  SUB(NSUB) = LINE(ISTART:ILAST)
  IF (ILAST .EQ. ILEN) RETURN

NSTART = ILAST + IFIRCH(LINE(ILAST+1:))

C Does SUB have any slashes?

II = INDEX(SUB(NSUB),'~')
  IF (II .LE. 0) THEN
    IF (LINE(NSTART:NSTART) .NE. '-') GO TO 10
    NEND = NSTART + INDEX(LINE(NSTART+1:),'~')
    IND = INDEX(SUB(NSUB),'') + 1
    SUB(NSUB)(IND:) = LINE(NSTART:NEND)
    IF (NEND .EQ. ILEN) RETURN
    NSTART = NEND + IFIRCH(LINE(NEND+1:))
  ENDIF

C Does SUB have 2 slashes?

I2 = INDEX(SUB(NSUB)(II+1:),'~')
  IF (I2 .GT. 0) GO TO 10

NEND = NSTART + INDEX(LINE(NSTART+1:),'~')
  IND = INDEX(SUB(NSUB),')' + 1
  SUB(NSUB)(IND:) = LINE(NSTART:NEND)
  IF (NEND .EQ. ILEN) RETURN
  NSTART = NEND + IFIRCH(LINE(NEND+1:))
  GO TO 10

100 CONTINUE
RETURN
END

C-----------------------------------------------------------------------
C---- SUBROUTINE IPNPAR (LINE, NPAR, IPAR, ISTART)
C
C Returns CHAR(*) IPAR substring of CHAR(*) string LINE which
C contains NPAR real parameters
C
C Input: LINE - a CHAR(*) line
C NPAR - number of parameters expected
C Output: IPAR - the substring of parameters only
C ISTART - the starting location of IPAR substring
C A '!' will comment out a line, or remainder of the line.
C F. Rupley, Div. 8245, 5/14/86
C-----------------------------------------------------------------------
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
CHARACTER(*) LINE,IPAR
C
C-----Find Comment String ('!' signifies comment)
ILEN = IPPLEN(LINE)
ISTART = 0
N = 0
IF (ILEN.GT.0) THEN
DO 40 I = ILEN, 1, -1
ISTART = I
IPAR = ' '
IPAR = LINE(ISTART:ILEN)
IF (LINE(I:I).NE.' ') THEN
ENDIF
DO 40 CONTINUE
ENDIF
RETURN
END

C SUBROUTINE IPPARI(STRING, ICARD, NEXPEC, IVAL, NFOUND, IERR, LOUT)
C BEGIN PROLOGUE IPPARI
C REFER TO IPGETI
C DATE WRITTEN 850625 (YYMMDD)
C REVISION DATE 851725 (YYMMDD)
C CATEGORY NO. J3., J4., M2.
C KEYWORDS PARSE
C AUTHOR CLARK, G. L., GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Parses integer variables from a character variable. Called
C by IPGETI, the IOPAK routine used for interactive input.
C DESCRIPTION
C
C-----------------------------------------------------------------------------------
C IPPARI may be used for parsing an input record that contains integer
C values, but was read into a character variable instead of directly
C into integer variables.
C The following benefits are gained by this approach:
C - specification of only certain elements of the array is allowed,
C thus letting the others retain default values
C - variable numbers of values may be input in a record, up to a
C specified maximum
C - control remains with the calling program in case of an input
C error
C - diagnostics may be printed by IPPARI to indicate the nature
C of input errors
C
C The contents of STRING on input indicate which elements of IVAL
C are to be changed from their entry values, and values to which
C they should be changed on exit. Commas and blanks serve as
C delimiters, but multiple blanks are treated as a single delimiter.
C Thus, an input record such as:
C ' 1, 2, 40000, 60'
C is interpreted as the following set of instructions by IPGETR:
C
C (1) set IVAL(1) = 1
C (2) set IVAL(2) = 2
C (3) leave IVAL(3) unchanged
C (4) set IVAL(4) = 40000
C (5) leave IVAL(5) unchanged
C (6) set IVAL(6) = 60
C
C IPPARI will print diagnostics on the default output device, if
C desired.
C
C IPPARI is part of IOPAK, and is written in ANSI FORTRAN 77
Examples:

Assume IVAL = (0, 0, 0) and NEXPEC = 3 on entry:

<table>
<thead>
<tr>
<th>input string</th>
<th>IVAL on exit</th>
<th>IERR</th>
<th>NFOUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>'2, 3 45'</td>
<td>(2, 3, 45)</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>'2.15, 3'</td>
<td>(2, 0, 3)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>'3X, 25, 2'</td>
<td>(0, 0, 0)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>'10000'</td>
<td>(10000, 0, 0)</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Assume IVAL = (0, 0, 0, 0) and NEXPEC = -4 on entry:

<table>
<thead>
<tr>
<th>input string</th>
<th>IVAL on exit</th>
<th>IERR</th>
<th>NFOUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>'1, 2'</td>
<td>(1, 2)</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>'r, 37, 400'</td>
<td>(0, 0, 37, 400)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>'1, -3, 5'</td>
<td>(1, 0, -3, 0)</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

arguments: (I=input, O=output)

STRING (I) - the character string to be parsed.

ICARD (I) - data statement number, and error processing flag
< 0: no error messages printed
= 0: print error messages, but not ICARD
> 0: print error messages, and ICARD

NEXPEC (I) - number of real variables expected to be input. If
< 0, the number is unknown, and any number of values
between 0 and abs(nexpec) may be input. (see NFOUND)
PROMPT (I) - prompting string, character type. A question
mark will be added to form the prompt at the screen.
IVAL (I, O) - the integer value or values to be modified. On entry,
the values are printed as defaults. The formal parameter
corresponding to IVAL must be dimensioned at least NEXPEC
in the calling program if NEXPEC > 1.
NFOUND (O) - the number of real values represented in STRING,
only in the case that there were as many or less than
NEXPEC.
IERR (O) - error flag:
= 0 if no errors found
= 1 syntax errors or illegal values found
= 2 for too few values found (NFOUND < NEXPEC)
= 3 for too many values found (NFOUND > NEXPEC)

REFERENCES (NONE)
ROUTINES CALLED IFIRCH, ILASCH
END PROLOGUE IPPARI
C*****precision > double
IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
C*****END precision > double
C
C*****precision > single
IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
C*****END precision > single
C
CHARACTER STRING(*), ITEMP*80
DIMENSION IVAL(*)
CHARACTER *8 FMT(14)
LOGICAL OKINCR

C FIRST EXECUTABLE STATEMENT IPPARI
IERR = 0
NFOUND = 0
NEXP = IABS(NEXPEC)
IE = ILASCH(STRING)
IF (IE .EQ. 0) GO TO 500
NC = 1

C--- OKINCR is a flag that indicates it's OK to increment
C--- NFOUND, the index of the array into which the value
C--- should be read. It is set false when a space follows
C--- an integer value substring, to keep incrementing from
C--- occurring if a comma should be encountered before the
C--- next value.
C
OKINCR = .TRUE.
C
C   begin overall loop on characters in string
C
100 CONTINUE
C
IF (STRING(NC:NC) .EQ. ',') THEN
  IF (OKINCR .OR. NC .EQ. IE) THEN
    NFOUND = NFOUND + 1
  ELSE
    OKINCR = .TRUE.
  ENDIF
ENDIF
IF (STRING(NC:NC) .EQ. ' ') GO TO 450
C
C   first good character (non-delimeter) found - now find
C   last good character
C
160 CONTINUE
NC = NC + 1
IF (NC .GT. IE) GO TO 180
IF (STRING(NC:NC) .EQ. ',') THEN
  OKINCR = .FALSE.
ELSEIF (STRING(NC:NC) .EQ. ',') THEN
  OKINCR = .TRUE.
ELSE
  GO TO 160
ENDIF
ENDIF

C C--- end of substring found - read value into integer array
C
180 CONTINUE
NFOUND = NFOUND + 1
IF (NFOUND .GT. NEXP) THEN
  IERR = 3
  GO TO 500
ENDIF

C IES = NC - 1
NCH = IES - IBS + 1
DATA FMT/' (II)', '  (12)', '  (13)', '  (14)', '  (15)',
1  '  (16)', '  (17)', '  (18)', '  (19)', '(110)',
2 '  (111)', '  (112)', '  (113)', '  (114)/
ITEMP = ' 'ITEMP = STRING(IBM:IES)
READ (ITEMP(1:NCH), FMT(NCH), ERR = 400) IVAL(NFOUND)
GO TO 450
400 CONTINUE
IERR = 1
GO TO 510
450 CONTINUE
NC = NC + 1
IF (NC .LE. IE) GO TO 100
C
500 CONTINUE
IF (NEXPEC .GT. 0 .AND. NFOUND .LT. NEXP) IERR = 2
510 CONTINUE
C
IF (IERR .EQ. 0 .OR. ICARD .LT. 0) RETURN
IF (ICARD .NE. 0) WRITE (LOUT, ' (A,13)')
 1 '!! ERROR IN DATA STATEMENT NUMBER', ICARD
IF (IERR .EQ. 1)
 1 WRITE (LOUT, '  (A) ')'SYNTAX ERROR, OR ILLEGAL VALUE'
 2 ' TOO FEW DATA ITEMS. NUMBER FOUND = ', NFOUND,
 1 ' NUMBER EXPECTED = ', NEXPEC
IF (IERR .EQ. 2) WRITE (LOUT, '(A,12, A, 12)')
 1 ' TOO MANY DATA ITEMS. NUMBER EXPECTED = ', NEXPEC
END
SUBROUTINE IPPARR(STRING, ICARD, NEXPEC, RVAL, NFOUND, IERR, LOUT)
C BEGIN PROLOGUE IPPARR
C REFER TO IPGETR
C DATE WRITTEN 850625 (YYMMDD)
C REVISION DATE 851625 (YYMMDD)
C CATEGORY NO. J3.,J4.,M2.
C KEYWORDS PARSE
C AUTHOR CLARK,G.L., GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Parses real variables from a character variable. Called
C by IPGETR, the IOPAK routine used for interactive input.
C DESCRIPTION
C---------------------------------------------------------------
C IPPARR may be used for parsing an input record that contains real
C values, but was read into a character variable instead of directly
C into real variables.
C The following benefits are gained by this approach:
C - specification of only certain elements of the array is allowed,
C thus letting the others retain default values
C - variable numbers of values may be input in a record, up to a
C specified maximum
C - control remains with the calling program in case of an input
C error
C - diagnostics may be printed by IPPARR to indicate the nature
C of input errors
C The contents of STRING on input indicate which elements of RVAL
C are to be changed from their entry values, and values to which
C they should be changed on exit. Commas and blanks serve as
C delimiters, but multiple blanks are treated as a single delimiter.
C Thus, an input record such as:
C ' 1., 2.,4.e-5 , ,6.e-6'
C is interpreted as the following set of instructions by IPGETR:
C
C (1) set RVAL(1) = 1.0
C (2) set RVAL(2) = 2.0
C (3) leave RVAL(3) unchanged
C (4) set RVAL(4) = 4.0E-05
C (5) leave RVAL(5) unchanged
C (6) set RVAL(6) = 6.0E-06
C
IPPARR will print diagnostics on the default output device, if desired.

IPPARR is part of IOPAK, and is written in ANSI FORTRAN 77

Examples:

Assume RVAL = (0., 0., 0.) and NEXPEC = 3 on entry:

input string      RVAL on exit      IERR   NFOUND
-------------------  -------------------  ------  ------
' 2.34e-3, 3 45.1'  (2.34E-03, 3.0, 45.1)  0     3
'2,,3,-5'          (2.0, 0.0, 3.02E-05)  0     3
'1.4,0.028E4'      (0.0, 1.4, 280.0)    0     3
'1.0, 2.44, 3.0'   (1.0, 0.0, 0.0)      1     1
'1.0'              (1.0, 0.0, 0.0)      2     1

Assume RVAL = (0., 0., 0., 0.) and NEXPEC = -4 on entry:

input string      RVAL on exit      IERR   NFOUND
-------------------  -------------------  ------  ------
'1.,2.'            (1.0, 2.0)         0     2
','3 4.0'          (0.0, 0.0, 3.0, 4.0)  0     4
'1,,3,,5.0'       (0.0, 0.0, 3.0, 0.0)  3     4

arguments: (I=input,O=output)

STRING (I) - the character string to be parsed.

ICARD (I) - data statement number, and error processing flag
< 0 : no error messages printed
= 0 : print error messages, but not ICARD
> 0 : print error messages, and ICARD

NEXPEC (I) - number of real variables expected to be input. If < 0, the number is unknown, and any number of values between 0 and abs(nexpec) may be input. (see NFOUND)

PROMPT (I) - prompting string, character type. A question mark will be added to form the prompt at the screen.

RVAL (I,O) - the real value or values to be modified. On entry, the values are printed as defaults. The formal parameter corresponding to RVAL must be dimensioned at least NEXPEC in the calling program if NEXPEC > 1.

NFOUND (O) - the number of real values represented in STRING, only in the case that there were as many or less than NEXPEC.

IERR (O) - error flag:
  = 0 if no errors found
  = 1 syntax errors or illegal values found
  = 2 for too few values found (NFOUND < NEXPEC)
  = 3 for too many values found (NFOUND > NEXPEC)

REFERENCES (NONE)
Routines called: IFIRCH, ILASCH
END PROLOGUE IPPARR
C*****precision > double
IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
C*****END precision > double
C
C*****precision > single
C IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
C*****END precision > single
C
CHARACTER STRING*(*) , ITEMP*80
DIMENSION R VAL(*)
CHARACTER *8 FMT(22)
LOGICAL OKINCR
C
C FIRST EXECUTABLE STATEMENT IPPARR
IERR = 0
NFOUND = 0
NEXP = IABS(NEXPEC)
IE = ILASCH(STRING)
IF (IE .EQ. 0) GO TO 500
NC = 1
C
C--- OKINCR is a flag that indicates it's OK to increment
C--- NFOUND, the index of the array into which the value
C--- should be read. It is set negative when a space follows
C--- a real value substring, to keep incrementing from
C--- occurring if a comma should be encountered before the
C--- next value.
C
OKINCR = .TRUE.
C
C--- begin overall loop on characters in string
C
100 CONTINUE
C
IF (STRING(NC:NC) .EQ. ',') THEN
  IF (OKINCR) THEN
    NFOUND = NFOUND + 1
  ELSE
    OKINCR = .TRUE.
  ENDIF
C
GO TO 450
ENDIF
IF (STRING(NC:NC) .EQ. ' ') GO TO 450
C
C--- first good character (non-delimeter) found - now find
C--- last good character
C
160 CONTINUE
NC = NC + 1
IF (NC .GT. IE) GO TO 180
IF (STRING(NC:NC) .EQ. ')') THEN
  OKINCR = .FALSE.
ELSEIF (STRING(NC:NC) .EQ. ',') THEN
  OKINCR = .TRUE.
ELSE
  GO TO 160
ENDIF
C
C--- end of substring found - read value into real array
C
180 CONTINUE
NFOUND = NFOUND + 1
IF (NFOUND .GT. NEXP) THEN
  IERR = 3
  GO TO 500
ENDIF
C
DATA FMT/ '(E1.0)', '(E2.0)', '(E3.0)', '(E4.0)',
1 ' '(E5.0)', '(E6.0)', '(E7.0)', '(E8.0)', '(E9.0)',
2 ' '(E10.0)', '(E11.0)', '(E12.0)', '(E13.0)', '(E14.0)' ,
IES = NC - 1
NCH = IES - IBS + 1
ITEMP = " 
ITEMP = STRING(ibs:ies)
READ (ITEMP (1:NCH), FMT(NCH), ERR = 400) RVAL(NFOUND)
GO TO 450
400 CONTINUE
WRITE (LOUT, 555) STRING(ibs:ies)
555 FORMAT (A)
IERR = 1
GO TO 510
450 CONTINUE
NC = NC + 1
IF (NC .LE. IE) GO TO 100
C
500 CONTINUE
IF (NEXPEC .GT. 0 .AND. NFOUND .LT. NEXP) IERR = 2
510 CONTINUE
C
IF (IERR .EQ. 0 .OR. ICARD .LT. 0) RETURN
IF (ICARD .NE. 0) WRITE (LOUT, '(A,13)')
1 '!!! ERROR IN DATA STATEMENT NUMBER', ICARD
IF (IERR .EQ. 1)
1 WRITE (LOUT, '(A)')'SYNTAX ERROR, OR ILLEGAL VALUE'
IF (IERR .EQ. 2) WRITE (LOUT, '(A,12, A, 12)')
1 ' TOO FEW DATA ITEMS. NUMBER FOUND = ', NFOUND,
2 ' NUMBER EXPECTED = ', NEXPEC
IF (IERR .EQ. 3) WRITE (LOUT, '(A,12)')
1 ' TOO MANY DATA ITEMS. NUMBER EXPECTED = ', NEXPEC
END
C
FUNCTION IFIRCH(STRING)
C BEGIN PROLOGUE IFIRCH
C DATE WRITTEN 850626
C REVISION DATE 850626
C CATEGORY NO. M 4 .
C KEYWORDS CHARACTER STRINGS,SIGNIFICANT CHARACTERS
C AUTHOR CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Determines first significant (non-blank) character
C in character variable
C DESCRIPTION
C------------------------------------------------------------
C IFIRCH locates the first non-blank character in a string of
C arbitrary length. If no characters are found, IFIRCH is set = 0.
C When used with the companion routine ILASCH, the length of a string
C can be determined, and/or a concatenated substring containing the
C significant characters produced.
C------------------------------------------------------------
C REFERENCES (NONE)
C ROUTINES CALLED (NONE)
C END PROLOGUE IFIRCH
C*****precision > double
C IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
C*****END precision > double
C
C CHARACTER* (*)STRING
C
C FIRST EXECUTABLE STATEMENT IFIRCH
NLOOP  =  LEN(STRING)

IF (NLOOP .EQ. 0) THEN
   IFIRCH = 0
   RETURN
ENDIF

DO 100 I = 1, NLOOP
   IF (STRING(I:I) .NE. ' ') GO TO 120
100 CONTINUE

IFIRCH = 0
RETURN

120 CONTINUE
IFIRCH = I
END

FUNCTION ILASCH(STRING)

C BEGIN PROLOGUE ILASCH
C DATE WRITTEN 85062 6
C REVISION DATE 850626
C CATEGORY NO. M4.
C KEYWORDS CHARACTER STRINGS, SIGNIFICANT CHARACTERS
C AUTHOR CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Determines last significant (non-blank) character
C in character variable
C DESCRIPTION
C------------------------------------------------------------
C IFIRCH locates the last non-blank character in a string of
C arbitrary length. If no characters are found, ILASCH is set = 0.
C When used with the companion routine IFIRCH, the length of a string
C can be determined, and/or a concatenated substring containing the
C significant characters produced.
C Note that the FORTRAN intrinsic function LEN returns the length
C of a character string as declared, rather than as filled. The
C declared length includes leading and trailing blanks, and thus is
C not useful in generating 'significant' substrings.
C------------------------------------------------------------
C REFERENCES (NONE)
C ROUTINES CALLED (NONE)
C END PROLOGUE IFIRCH

C***FIRST EXECUTABLE STATEMENT ILASCH
NLOOP  =  LEN(STRING)
IF (NLOOP.EQ.0) THEN
   ILASCH = 0
   RETURN
ENDIF

DO 100 I = NLOOP, 1, -1
   IF (STRING(I:I) .NE. ' ') GO TO 120
100 CONTINUE

ILASCH = I
END
SUBROUTINE CKCOMP (1ST, IRAY, II, I)

C START PROLOGUE

C SUBROUTINE CKCOMP (1ST, IRAY, II, I)*
C Returns the index of an element of a reference character
C string array which corresponds to a character string;
C leading and trailing blanks are ignored.

C INPUT
C 1ST - A character string.
C Data type - CHARACTER(*)
C IRAY - An array of character strings; dimension IRAY(*) at least II
C Data type - CHARACTER(*)
C II - The length of IRAY.
C Data type - integer scalar.

C OUTPUT
C I - The first integer location in IRAY in which 1ST
C corresponds to IRAY(I); if 1ST is not also an
C entry in IRAY, I=0.

END PROLOGUE

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 CHARACTER(*) 1ST, IRAY(*)

I = 0
DO 10 N = II, 1, -1
  IS1 = IFRICH(1ST)
  IS2 = ILASCH(1ST)
  IR1 = IFRICH(IRAY(N))
  IR2 = ILASCH(IRAY(N))
  IF ( IS2.GE.IS1 .AND. IS2.GT.0 .AND.
       1 IR2.GE.IR1 .AND. IR2.GT.0 .AND.
       2 1ST(IS1:IS2).EQ.IRAY(N)(IR1:IR2) ) I=N
10 CONTINUE
RETURN
END

SUBROUTINE CKUNIT (LINE, AUNITS, EUNITS, IUNITS)

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 CHARACTER(*) LINE, AUNITS, EUNITS
CHARACTER*4 UPCASE

AUNITS = '  ' EUNITS = '  ' IUNITS = '  '
DO 85 N = 1, ILASCH(LINE)-3
IND = ILASCH(IUNITS)
IF (EUNITS .EQ. ' ') THEN
IF (UPCASE(LINE(N:), 4) .EQ. 'CAL/') THEN
EUNITS = 'CAL/
ELSE
IUNITS = 'E units cal/mole'
ENDIF
ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KCAL') THEN
EUNITS = 'KCAL'
ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'JOUL') THEN
EUNITS = 'JOUL'
ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KJOU') THEN
EUNITS = 'KJOU'
ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KELV') THEN
EUNITS = 'KELV'
ELSEIF (UPCASE(LINE(N+4:), 4) .EQ. 'MOLE') THEN
IF (UPCASE(LINE(N+4:), 1) .EQ. 'S') THEN
AUNITS = 'MOLE'
ELSE
IUNITS = 'A units mole-cm-sec-K'
ENDIF
ELSEIF (UPCASE(LINE(N+4:), 4) .EQ. 'MOLC') THEN
AUNITS = 'MOLC'
ELSE
IUNITS = 'A units molecules'
ENDIF
ENDIF
ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'C') THEN
AUNITS = 'MOLE'
IND = ILSCH(IUNITS) + 1
IF (IND .GT. 1) THEN
IUNITS(IND:) = 'A units mole-cm-sec-K'
ELSE
C IUNITS(IND:) = ' A units mole-cm-sec-K'
ENDIF
ENDIF
C IF (EUNITS .EQ. ' ') THEN
   EUNITS = 'CAL/'
   IND = ILASCH(IUNITS) + 1
   IF (IND .GT. 1) THEN
      IUNITS(IND:) = E units cal/mole'
   ELSE
      IUNITS(IND:) = E units cal/mole'
   ENDIF
ENDIF
CRETURN
END

C INTEGER FUNCTION IPPLEN (LINE)

C BEGIN PROLOGUE

C FUNCTION IPPLEN (LINE)
Returns the effective length of a character string, i.e.,
the index of the last character before an exclamation mark (!)
indicating a comment.

C INPUT
LINE - A character string.
C
C OUTPUT
IPPLEN - The effective length of the character string.
C
C END PROLOGUE
C
C*****precision > double
IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
C*****END precision > double
C
C*****precision > single
IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
C*****END precision > single
C
CHARACTER LINE*(*)
C
IN = IFIRCH(LINE)
IF (IN.EQ.0 .OR. LINE(IN:IN).EQ.'!') THEN
   IPPLEN = 0
ELSE
   IN = INDEX(LINE,'!')
   IF (IN .EQ. 0) THEN
      IPPLEN = ILASCH(LINE)
   ELSE
      IPPLEN = ILASCH(LINE(:IN-1))
   ENDIF
ENDIF
RETURN
END
C
CHARACTER*(*) FUNCTION UPPCASE(ISTR, ILEN)
CHARACTER ISTR*(*), LCASE(26)*1, UCASE(26)*1
DATA LCASE /'a','b','c','d','e','f','g','h','i','j','k','l','m'/
   /'n','o','p','q','r','s','t','u','v','w','x','y','z'/
   /'N','O','P','Q','R','S','T','U','V','W','X','Y','Z'/
   /'A','B','C','D','E','F','G','H','I','J','K','L','M'/
   /'N','O','P','Q','R','S','T','U','V','W','X','Y','Z'/
C
UPCASE = ' '
UPCASE = ISTR(:ILEN)
JJ = MIN (LEN (UPCASE), LEN (ISTR), ILEN)
DO 10 J = 1, JJ
    DO 10 N = 1, 26
        IF (ISTR(J:J) .EQ. LCASE(N)) UPCASE(J:J) = UCASE(N)
10 CONTINUE
RETURN
END

SUBROUTINE THERM_IO(ERROR)
C
IMPLICIT NONE
INCLUDE 'THM_PAR.FI'
C
CHARACTER FILE_NAME*20
INTEGER IDOT, ISTART, IFIRCH
LOGICAL ERROR
C
ERROR = .FALSE.
5 WRITE(*,10)
10 FORMAT(' Please enter the input file name : '
READ(*, '(A20)') FILE_INP
OPEN(INP, FILE=FILE_INP, STATUS='OLD', ERR=30)
    GO TO 50
30 CONTINUE
    WRITE(*,40) FILE_INP
40 FORMAT (' Can not find file : ',A20)
    GO TO 5
50 CONTINUE
    ISTART = IFIRCH(FILE_INP)
    IDOT = INDEX(FILE_INP, '.')
    FILE_NAME = FILE_INP(ISTART:IDOT-1)
    FILE_LOG = FILE_NAME + '.' + 'LOG'
C    FILE_LOG = 'THERMCAL.LOG'
    FILE_LOG(IDOT:IDOT) = '.
    FILE_LOG(IDOT+1:IDOT+3) = 'LOG'
C
FILE_LST = FILE_NAME
    FILE_LST = FILE_NAME + '.' + 'LST'
C
FILE_DAT = FILE_NAME
    FILE_DAT = FILE_NAME + '.' + 'DAT'
C
OPEN(LOG, FILE=FILE_LOG, STATUS='UNKNOWN', ERR=230)
    CLOSE(LOG, STATUS='DELETE')
OPEN(LOG, FILE=FILE_LOG, STATUS='NEW', ERR=230)
    GO TO 250
230 CONTINUE
    WRITE(*,240) FILE_LOG
240 FORMAT(' File I/O Error : ',A20)
    ERROR = .TRUE.
    GO TO 350
250 CONTINUE
C
OPEN(LST, FILE=FILE_LST, STATUS='UNKNOWN', ERR=330)
    CLOSE(LST, STATUS='DELETE')
OPEN(LST, FILE=FILE_LST, STATUS='NEW', ERR=330)
    GO TO 350
330 CONTINUE
    WRITE(*,340) FILE_LST
340 FORMAT(' File I/O Error : ',A20)
    ERROR = .TRUE.
    GO TO 350
350 CONTINUE
C
RETURN
END
D.2 THERMSRT

C
C                  PROGRAM THERMCLV
C
C-----------------------------------------------C
C**********changed W.Ing 06-15-94, 08-01-95**********C
C
C IMPLICIT NONE
C
C           include 'thm_par2.fi'
C           include 'thm_cfg.fi'
C
C CHARACTER specy_name*70, specy_formula*40,
1 specy_group(Max_Specy_Group)*14, KEY2(3)*4,
2 specy_element(Max_Element)*4, str_int*8,
3 specy_comment*70, specy_phase, line*100, KEY(5)*5,
4 SUB(20)*100, UPCASE*5, str*100, qty*100, NAME*200
C
C INTEGER specy_group_qty(Max_Specy_Group), specy_symmetry,
1 specy_element_qty(Max_Element), Ngroup, ILASCH,
2 ILEN, IPPLEN, NSUB, NKEY, Isymmetry(5), Nval, Ierr,
3 I, Istr, Igroup(5), Iifich, Islash, Iqty(5), Ifound,J,
4 Index, Itotal, Iname, NKEY2, Nloop
C
C REAL*8 specy_Hf, specy_S, specy_Cp300, specy_Cp400, specy_Cp500,
2 specy_Cp1500, specy_rotor, rotor(5)
C
C LOGICAL therm, nasa, err, io_err
C
C DATA KEY/'UNITS','RADIOLXXX','END','!'/
C DATA KEY2/'SYMM','NROT','ENDS'/
C
C Call Initialization subroutines, 02/95 ING
C
C CALL TITLE
C CALL THERM_IO(io_err)
C if (io_err) GO TO 6000
C CALL THERM_INIT(init_err)
C if (init_err) GO TO 6000
C
C WRITE(1LST,*)'THERM BATCH INPUT FILE'
C WRITE(1LST,50)
C 50 FORMAT(' SPECIES  Hf  S  Cp 300  400  500  600  1000  1500  DATE  ELEMENTS')
C
C 100 CONTINUE
specy_rotor = 0.0
specy_symmetry = 1
LINE = ' ' READ (INP, '(A)', END=5000) LINE
105 CONTINUE
ILEN = IPPLEN(LINE)
IF (ILEN .EQ. 0) GO TO 100
READ (INP, '(A)', END=5000) LINE
READ (INP, '(A)', END=5000) LINE
READ (INP, '(A)', END=5000) LINE
ILEN = IPPLEN(LINE)
CALL CKISUB (LINE(:,ILEN), SUB, NSUB)
specy_name = SUB(1)
specy_formula = SUB(2)
C READ (INP,'(A)',END=5000) LINE
ILEN = IPPLEN(LINE)
CALL CKISUB (LINE(:ILEN), SUB, NSUB)

C IS THERE A KEYWORD?
C CALL CKCOMP ( UPCASE(SUB(1), 5) , KEY, 2, NKEY)
C WRITE(*,*) UPCASE(SUB(1),3),KEY(1),NKEY
IF (NKEY .GT. 0) ITASK = 0
C WRITE(*,*) NKEY
C WRITE(*,*) NKEY
IF (NKEY.EQ.2) THEN
  READ (INP,'(A)',END=5000) LINE
  READ (INP,'(A)',END=5000) LINE
  READ (INP,'(A)',END=5000) LINE
ELSEIF (NKEY.EQ.1) THEN
  C DO NOTHING FOR MOLECULES
ELSE
  C ERROR FOR NEITHER MOLECULES NOR RADICALS
  GO TO 6000
ENDIF
READ (INP,'(A)',END=5000) LINE
ILEN = IPPLEN(LINE)
CALL CKISUB (LINE(:ILEN), SUB, NSUB)
call IPPARI (SUB(2), -1, 1, Igroup, nval, Ierr, LOG)
Ngroup = Igroup(1)
IF (Ierr.NE.0 .OR. Ngroup.EQ.0) THEN
  CONTINUE
WRITE('**','specy_name',**' group number error !!!'
  READ (INP,'(A)',END=5000) LINE
  ILEN = IPPLEN(LINE)
  CALL CKISUB (LINE(:ILEN), SUB, NSUB)
call IPPARI (SUB(5), -1, 1, Iqty, nval, Ierr, LOG)
specy_group(I)(3) = SUB(3)
specy_group_qty(I) = Iqty(1)
200 CONTINUE
ELSE
  Nloop = INT(Ngroup/2.0)
  write('**','Ngroup = ',Ngroup, 'Nloop = ',Nloop
  IF (MOD(Ngroup,2).EQ.0) THEN
    DO 210 I = 1, Nloop
      READ (INP,'(A)',END=5000) LINE
      ILEN = IPPLEN(LINE)
      CALL CKISUB (LINE(:ILEN), SUB, NSUB)
call IPPARI (SUB(5), -1, 1, Iqty, nval, Ierr, LOG)
specy_group(Nloop+I)(3) = SUB(3)
specy_group_qty(Nloop+I) = Iqty(1)
210 CONTINUE
ELSE
  Nloop = INT(Ngroup/2.0)
  write('**','Ngroup = ',Ngroup, 'Nloop = ',Nloop
  IF (MOD(Ngroup,2).EQ.0) THEN
    DO 210 I = 1, Nloop
      READ (INP,'(A)',END=5000) LINE
      ILEN = IPPLEN(LINE)
      CALL CKISUB (LINE(:ILEN), SUB, NSUB)
call IPPARI (SUB(5), -1, 1, Iqty, nval, Ierr, LOG)
specy_group(Nloop+I)(3) = SUB(3)
specy_group_qty(Nloop+I) = Iqty(1)
210 CONTINUE
C write(**,I(specy_group(I),specy_group_qty(I)
C write(**Nloop+I,specy_group(Nloop+I),specy_group_qty(Nloop+I)
ELSE
  DO 220 I = 1, Nloop
    READ (INP,'(A)',END=5000) LINE
    ILEN = IPPLEN(LINE)
    CALL CKISUB (LINE(:ILEN), SUB, NSUB)
    specy_group(I) = SUB(3)
    call IPPARI (SUB(5), -1, 1, Iqty, nval, Ierr, LOG)
    specy_group_qty(I) = Iqty(1)
    specy_group(Nloop+I+1) = SUB(9)
    call IPPARI (SUB(11), -1, 1, Iqty, nval, Ierr, LOG)
    specy_group_qty(Nloop+I+1) = Iqty(1)
  CONTINUE
  READ (INP,'(A)',END=5000) LINE
  ILEN = IPPLEN(LINE)
  CALL CKISUB (LINE(:ILEN), SUB, NSUB)
  specy_group(Nloop+1) = SUB(3)
  call IPPARI (SUB(5), -1, 1, Iqty, nval, Ierr, LOG)
  specy_group_qty(Nloop+1) = Iqty(1)
ENDIF
ENDIF
C WRITE(LST,*) LINE
C READ (INP,'(A)',END=5000) LINE
C READ (INP,'(A)',END=5000) LINE
C READ (INP,'(A)',END=5000) LINE
250 CONTINUE
  READ (INP,'(A)',END=5000) LINE
  ILEN = IPPLEN(LINE)
  CALL CKISUB (LINE(:ILEN), SUB, NSUB)
  CALL CKCOMP (UPCASE(SUB(1), 4), KEY2, 3, NKEY2)
  WRITE(*,*) UPCASE(SUB(1), 4)
  IF (NKEY2.EQ.0) GO TO 250
  IF (NKEY2.EQ.3) GO TO 280
  IF (NKEY2.EQ.2) THEN
    call IPPARR (SUB(2), -1, 1, rotor, nval, Ierr, LOG)
    IF (Ierr.NE.0) GO TO 6000
    specy_rotor = rotor(1)
  ENDIF
  READ (INP,'(A)',END=5000) LINE
  ILEN = IPPLEN(LINE)
  CALL CKISUB (LINE(:ILEN), SUB, NSUB)
  call IPPARI (SUB(2), -1, 1, Isymmetry, nval, Ierr, LOG)
  IF (Ierr.NE.0) GO TO 6000
  specy_symmetry = Isymmetry(1)
ELSE
  call IPPARI (SUB(2), -1, 1, Isymmetry, nval, Ierr, LOG)
  write(*,*) line
  write(*,*) sub(2)
  IF (Ierr.NE.0) GO TO 6000
  specy_symmetry = Isymmetry(1)
ENDIF
GO TO 250
C
280 CONTINUE
  write(LST,300) specy_name, specy_formula,
  specy_symmetry, specy_rotor
300 format(A12,' ',A12,' ',I4,' ',F4.1)
  write(LST,300) specy_name(:ILASCH(specy_name)),
  specy_formula(:ILASCH(specy_name))
  specy_symmetry, specy_rotor
  specy_symmetry = Isymmetry(1)
C
INDEX = 1
Itotal = 0
DO 350 J = 1, Ngroup
  Itotal = Itotal + 1
  NAME(Itotal:Itotal) = ' '
  Index = Itotal + 1
  Iname = IPPLEN(specy_group(J))
Itotal = Itotal + Iname
NAME(Index:Itotal) = specy_group(J)
NAME(Itotal+1:Itotal+1) = ''
CALL CKI2CH(specy_group_qty(J), str_int, Iname, err)
NAME(Itotal+2:Itotal+1+Iname) = str_int
Itotal = Itotal + 1 + Iname

350 CONTINUE
ILEN = IPPLEN(NAME)
write(*,*)'..Itotal = ',itotal
write(1st,360)NAME(:Itotal)
360 format(' ,A')
write(LST,310)({specy_group(J){:ILASCH(specy_GROUP(J))},J=1,Ngroup)
310 format(' '' ,A,"\n
c
DUMP USELESS SPECY-END INFO FOR MOLECULES AND RADICALS

C
C IF (NKEY.EQ.2) THEN
C READ (INP,'(A)',END=5000) LINE
C ELSE
C ENDIF
C
C go to 100
5000 write(*,*)'...end of conversion...'
WRITE(LST,5100)
5100 FORMAT('END')

C-------------------------------------------------------------C
SUBROUTINE CKISUB (LINE, SUB, NSUB)
C
C Generates an array of CHAR*(*) substrings from a CHAR*(*) string,
C using blanks or tabs as delimiters
C
C Input: LINE - a CHAR*(*) line
C Output: SUB - a CHAR*(*) array of substrings
C NSUB - number of substrings found
C A '!' will comment out a line, or remainder of the line.
C
F. Rupley, Div. 8245, 5/15/86
C-------------------------------------------------------------C
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
CHARACTER*(*) SUB(*), LINE
NSUB = 0
C
DO 5 N = 1, LEN(LINE)
   IF (ICHAR(LINE(N:N)) .EQ. 9) LINE(N:N) = ''
5 CONTINUE
C
   IF (IPPLEN(LINE) .LE. 0) RETURN
ILEN = ILASCH(LINE)

ISTART = IIFIRCH(LINE)
10 CONTINUE
ISTART = ISTART
NSUB = NSUB + 1
SUB(NSUB) = ' '

DO 100 I = ISTART, ILEN
ILAST = INDEX(LINE(ISTART:),' ') - 1
IF (ILAST .GT. 0) THEN
   ILAST = ISTART + ILAST - 1
ELSE
   ILAST = ILEN
ENDIF
SUB(NSUB) = LINE(ISTART:ILAST)
IF (ILAST .EQ. ILEN) RETURN

NSTART = ILAST + IIFIRCH(LINE(ILAST+1:))

Does SUB have any slashes?
I1 = INDEX(SUB(NSUB),'-')
IF (I1 .LE. 0) THEN
   IF (LINE(NSTART:NSTART) .NE. '-') GO TO 10
   NEND = NSTART + INDEX(LINE(NSTART+1:),'-')
   IND = INDEX(SUB(NSUB),'-')+1
   SUB(NSUB)(IND:) = LINE(NSTART:NEND)
   IF (NEND .EQ. ILEN) RETURN
   NSTART = NEND + IIFIRCH(LINE(NEND+1:))
   GO TO 10
ENDIF

Does SUB have 2 slashes?
I2 = INDEX(SUB(NSUB)(I1+1:),'-')
IF (I2 .GT. 0) GO TO 10

NEND = NSTART + INDEX(LINE(NSTART+1:),'-')
IND = INDEX(SUB(NSUB),'-')+1
SUB(NSUB)(IND:) = LINE(NSTART:NEND)
IF (NEND .EQ. ILEN) RETURN
NSTART = NEND + IIFIRCH(LINE(NEND+1:))
GO TO 10

100 CONTINUE
RETURN
END

C-------------------------------------------------------------C
SUBROUTINE IPNPAR (LINE, NPAR, IPAR, ISTART)
C
C Returns CHAR(*) IPAR substring of CHAR(*) string LINE which
C contains NPAR real parameters
C
C Input:  LINE - a CHAR(*) line
C         NPAR - number of parameters expected
C Output: IPAR - the substring of parameters only
C         ISTART - the starting location of IPAR substring
C A '!' will comment out a line, or remainder of the line.
C
F. Rupley, Div. 8245, 5/14/86
C-------------------------------------------------------------C
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
CHARACTER(*) LINE, IPAR
C
Find Comment String (! signifies comment)
C
ILEN = IPPLEN(LINE)
ISTART = 0
N = 0
IF (ILEN .GT. 0) THEN
  DO 40 I = ILEN, 1, -1
    ISTART = I
    IPAR = LINE(ISTART:ILEN)
    IF (LINE(I:I) .NE. '!') THEN
      IF (I .EQ. 1) RETURN
      IF (LINE(I-1) .EQ. ' ') THEN
        N = N + 1
      ELSEIF (N .EQ. NPAR) RETURN
      ENDIF
    ENDIF
  CONTINUE
ENDIF
RETURN
END

SUBROUTINE IPPARI(STRING, ICARD, NEXPEC, IVAL, NFOUND, IERR, LOUT)

BEGIN PROLOGUE IPPARI
REFER TO IPGETI
DATE WRITTEN  850625 (YYMDD)
REVISION DATE  851725 (YYMDD)
CATEGORY NO.  J3,,J4,,M2.
KEYWORDS  PARSE
AUTHOR  CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
PURPOSE Parses integer variables from a character variable. Called
by IPGETI, the IOPAK routine used for interactive input.
DESCRIPTION

IPPAR may be used for parsing an input record that contains integer
values, but was read into a character variable instead of directly
into integer variables.

The following benefits are gained by this approach:
- specification of only certain elements of the array is allowed,
  thus letting the others retain default values
- variable numbers of values may be input in a record, up to a
  specified maximum
- control remains with the calling program in case of an input
  error
- diagnostics may be printed by IPPARI to indicate the nature
  of input errors

The contents of STRING on input indicate which elements of IVAL
are to be changed from their entry values, and values to which
they should be changed on exit. Commas and blanks serve as
delimiters, but multiple blanks are treated as a single delimiter.
Thus, an input record such as:
' 1, 2,,40000 ,,60'
is interpreted as the following set of instructions by IPGETI:
(1) set IVAL(1) = 1
(2) set IVAL(2) = 2
(3) leave IVAL(3) unchanged
(4) set IVAL(4) = 40000
(5) leave IVAL(5) unchanged
(6) set IVAL(6) = 60
IPPARI will print diagnostics on the default output device, if desired.

IPPARI is part of IOPAK, and is written in ANSI FORTRAN 77

Examples:

Assume IVAL = (0, 0, 0) and NEXPEC = 3 on entry:

<table>
<thead>
<tr>
<th>Input string</th>
<th>IVAL on exit</th>
<th>IERR</th>
<th>NFOUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>'2, 3 45'</td>
<td>(2, 3, 45)</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>'2.15, 3'</td>
<td>(2, 0, 3)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>'3X, 25, 2'</td>
<td>(0, 0, 0)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>'10000'</td>
<td>(10000, 0, 0)</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Assume IVAL = (0, 0, 0, 0) and NEXPEC = -4 on entry:

<table>
<thead>
<tr>
<th>Input string</th>
<th>IVAL on exit</th>
<th>IERR</th>
<th>NFOUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>'1, 2'</td>
<td>(1, 2)</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>'1, 37, 400'</td>
<td>(0, 0, 37, 400)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>'1, -3, 5'</td>
<td>(1, 0, -3, 0)</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Arguments: (I=Input, O=Output)

STRING (I) - the character string to be parsed.

ICARD (I) - data statement number, and error processing flag

< 0: no error messages printed
= 0: print error messages, but not ICARD
> 0: print error messages, and ICARD

NEXPEC (I) - number of real variables expected to be input. If
< 0, the number is unknown, and any number of values
between 0 and abs(nexpec) may be input. (see NFOUND)

PROMPT (I) - prompting string, character type. A question
mark will be added to form the prompt at the screen.

IVAL (I, O) - the integer value or values to be modified. On entry,
the values are printed as defaults. The formal parameter
corresponding to IVAL must be dimensioned at least NEXPEC
in the calling program if NEXPEC > 1.

NFOUND (O) - the number of real values represented in STRING,
only in the case that there were as many or less than
NEXPEC.

IERR (O) - error flag:

= 0 if no errors found
= 1 syntax errors or illegal values found
= 2 for too few values found (NFOUND < NEXPEC)
= 3 for too many values found (NFOUND > NEXPEC)

References (None)
Routines Called IFIRCH, IALASCH
End Prologue IPPARI

C***** precision > double
IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
C***** End precision > double
C
C***** precision > single
C IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
C****END precision > single
C
C
CHARACTER STRING*(*), ITEMP*80
DIMENSION IVAL(*)
CHARACTER *8 FMT(14)
LOGICAL OKINCR
C
C FIRST EXECUTABLE STATEMENT IPPARI
IERR = 0
NFOUND = 0
NEXP = IABS(NEXPNC)
IE = IASCH(STRING)
IF (IE .EQ. 0) GO TO 500
NC = 1
C
C OKINCR is a flag that indicates it's OK to increment
C NFOUND, the index of the array into which the value
C should be read. It is set false when a space follows
C an integer value substring, to keep incrementing from
C occurring if a comma should be encountered before the
C next value.
C
OKINCR = .TRUE.
C
C--- begin overall loop on characters in string
C 100 CONTINUE
C
IF (STRING(NC:NC) .EQ. ',') THEN
IF (OKINCR .OR. NC .EQ. IE) THEN
   NFOUND = NFOUND + 1
ELSE
   OKINCR = .TRUE.
ENDIF
C
GO TO 450
ENDIF
IF (STRING(NC:NC) .EQ. ' ') GO TO 450
C
C--- first good character (non-delimiter) found - now find
C--- last good character
C
IBS = NC
160 CONTINUE
NC = NC + 1
IF (NC .GT. IE) GO TO 180
IF (STRING(NC:NC) .EQ. ',') THEN
   OKINCR = .FALSE.
ELSEIF (STRING(NC:NC) .EQ. ',') THEN
   OKINCR = .TRUE.
ELSE
   GO TO 160
ENDIF
C
ENDIF
C
C--- end of substring found - read value into integer array
C
180 CONTINUE
NFOUND = NFOUND + 1
IF (NFOUND .GT. NEXP) THEN
   IERR = 3
   GO TO 500
ENDIF
C
IES = NC - 1
NCH = IES - IBS + 1
DATA FMT/'(I1)', ' (I2)', ' (I3)', ' (I4)', ' (I5)',
1 ' (I6)', ' (I7)', ' (I8)', ' (I9)', ' (I10)',
2 ' (I11)', ' (I12)', ' (I13)', ' (I14)'/
ITEMP = '
ITEMP = STRING(IBS:IES)
READ (ITEMP(1:NCH), FMT(NCH), ERR = 400) IVAL(NFOUND)
GO TO 450
400 CONTINUE
IERR = 1
GO TO 510
450 CONTINUE
NC = NC + 1
IF (NC .LE. IE) GO TO 100
C
500 CONTINUE
IF (NEXPEC .GT. 0 .AND. NFOUND .LT. NEXP) IERR = 2
510 CONTINUE
C
IF (IERR .EQ. 0 .OR. ICARD .LT. 0) RETURN
IF (ICARD .NE. 0) WRITE (LOUT, '(A,I3)')
1 '!! ERROR IN DATA STATEMENT NUMBER', ICARD
IF (IERR .EQ. 1)
1 WRITE (LOUT, '(A)') 'SYNTAX ERROR, OR ILLEGAL VALUE'
IF (IERR .EQ. 2) WRITE (LOUT, '(A,12,A,12)')
1 ' TOO FEW DATA ITEMS. NUMBER FOUND = ', NFOUND,
2 ' NUMBER EXPECTED = ', NEXPEC
IF (IERR .EQ. 3) WRITE (LOUT, '(A,12)')
1 ' TOO MANY DATA ITEMS. NUMBER EXPECTED = NEXPEC
END

SUBROUTINE IPPARR(STRING, ICARD, NEXPEC, RVAL, NFOUND, IERR, LOUT)
C
BEGIN PROLOGUE IPPARR
C REFER TO IPGETR
C DATE WRITTEN 850625 (YMMDD)
C REVISION DATE 851625 (YMMDD)
C CATEGORY NO. J3,,J4,,M2.
C KEYWORDS PARSE
C AUTHOR CLARK,,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Parses real variables from a character variable. Called
C by IPGETR, the IOPAK routine used for interactive input.
C DESCRIPTION
C-----------------------------------------------------------------------------------------------------------------
C IPPARR may be used for parsing an input record that contains real
C values, but was read into a character variable instead of directly
C into real variables.
C The following benefits are gained by this approach:
C - specification of only certain elements of the array is allowed,
C   thus letting the others retain default values
C - variable numbers of values may be input in a record, up to a
C   specified maximum
C - control remains with the calling program in case of an input
C   error
C - diagnostics may be printed by IPPARR to indicate the nature
C   of input errors
C
C The contents of STRING on input indicate which elements of RVAL
C are to be changed from their entry values, and values to which
C they should be changed on exit. Commas and blanks serve as
C delimiters, but multiple blanks are treated as a single delimeter.
C Thus, an input record such as:
C ' 1., 2.,4.0-5, ,6.0-6'
C is interpreted as the following set of instructions by IPGETR:
C
C (1) set RVAL(1) = 1.0
C (2) set RVAL(2) = 2.0
IPPARR will print diagnostics on the default output device, if desired.

IPPARR is part of IOPAK, and is written in ANSI FORTRAN 77

Examples:

Assume RVAL = (0., 0., 0.) and NEXPEC = 3 on entry:

<table>
<thead>
<tr>
<th>input string</th>
<th>RVAL on exit</th>
<th>IERR</th>
<th>NFOUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>' 2.34e-3, 3 45.1'</td>
<td>(2.34E-03, 3.0, 45.1)</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>'2.,3.-5'</td>
<td>(2.0, 0.0, 3.0E-05)</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>'1.4,0.02E4'</td>
<td>(0.0, 1.4, 280.0)</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>'1.0, 2.a4, 3.0'</td>
<td>(1.0, 0.0, 0.0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>'1.0'</td>
<td>(1.0, 0.0, 0.0)</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Assume RVAL = (0., 0., 0., 0.) and NEXPEC = -4 on entry:

<table>
<thead>
<tr>
<th>input string</th>
<th>RVAL on exit</th>
<th>IERR</th>
<th>NFOUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>'1.,2.'</td>
<td>(1.0, 2.0)</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>'1.,3 4.0'</td>
<td>(0.0, 0.0, 3.0, 4.0)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>'1.,3,5.0'</td>
<td>(0.0, 0.0, 3.0, 0.0)</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

arguments: (I=input,O=output)

STRING (I) - the character string to be parsed.

ICARD (I) - data statement number, and error processing flag
  < 0 : no error messages printed
  = 0 : print error messages, but not ICARD
  > 0 : print error messages, and ICARD

NEXPEC (I) - number of real variables expected to be input. If
  < 0, the number is unknown, and any number of values
  between 0 and abs(nexpec) may be input. (see NFOUND)

PROMPT (I) - prompting string, character type. A question
  mark will be added to form the prompt at the screen.

RVAL (I,O) - the real value or values to be modified. On entry,
  the values are printed as defaults. The formal parameter
  corresponding to RVAL must be dimensioned at least NEXPEC
  in the calling program if NEXPEC > 1.

NFOUND (O) - the number of real values represented in STRING,
  only in the case that there were as many or less than
  NEXPEC.

IERR (O) - error flag:
  = 0 if no errors found
  = 1 syntax errors or illegal values found
  = 2 for too few values found (NFOUND < NEXPEC)
  = 3 for too many values found (NFOUND > NEXPEC)

REFERENCES (NONE)

ROUTINES CALLED IFIRCH,ILASCH

END PROLOGUE IPPARR

**precision > double**
IMPLICIT DOUBLE PRECISION (A-H,0-Z), INTEGER (I-N)
C*****END precision > double
C
C****precision > single
C IMPLICIT REAL (A-H,0-Z), INTEGER (I-N)
C*****END precision > single
C
CHARACTER STRING(*), ITEMP*80
DIMENSION RVAL(*)
CHARACTER *8 FMT(22)
LOGICAL OKINCR
C
C FIRST EXECUTABLE STATEMENT IPPARR
IERR = 0
NFOUND = 0
NEXP = IABS(NEXPEC)
IE = ILASCH(STRING)
IF (IE .EQ. 0) GO TO 500
NC = 1
C
C--- OKINCR is a flag that indicates it's OK to increment
C--- NFOUND, the index of the array into which the value
C--- should be read. It is set negative when a space follows
C--- a real value substring, to keep incrementing from
C--- occurring if a comma should be encountered before the
C--- next value.
C
OKINCR = .TRUE.
C
C--- begin overall loop on characters in string
C
100 CONTINUE
C IF (STRING(NC:NC) .EQ. ',') THEN
 IF (OKINCR) THEN
   NFOUND = NFOUND + 1
 ELSE
   OKINCR = .TRUE.
 ENDIF
 C GO TO 450
ENDIF
IF (STRING(NC:NC) .EQ. ' ') GO TO 450
C
C--- first good character (non-delimeter) found - now find
C--- last good character
C
IBS = NC
160 CONTINUE
NC = NC + 1
IF (NC .GT. IE) GO TO 180
IF (STRING(NC:NC) .EQ. ',') THEN
 OKINCR = .FALSE.
 ELSEIF (STRING(NC:NC) .EQ. ',') THEN
 OKINCR = .TRUE.
 ELSE
 GO TO 160
 ENDIF
C
C--- end of substring found - read value into real array
C
180 CONTINUE
NFOUND = NFOUND + 1
IF (NFOUND .GT. NEXP) THEN
 IERR = 3
 GO TO 500
C
ENDIF
C
DATA FMT/
1 ' (E5.0) ', ' (E6.0) ', ' (E7.0) ', ' (E8.0) ', ' (E9.0) ',
2 ' (E10.0) ', ' (E11.0) ', ' (E12.0) ', ' (E13.0) ', ' (E14.0) ',
3 ' (E15.0) ', ' (E16.0) ', ' (E17.0) ', ' (E18.0) ', ' (E19.0) ',
4 ' (E20.0) ', ' (E21.0) ', ' (E22.0) '/
IES = NC - 1
NCH = IES - IBS + 1
ITEMP = ' '
ITEMP = STRING(IFS:IES)
READ (ITEMP(1:NCH), FMT(NCH), ERR = 400) RVAL(NFOUND)
GO TO 450
400 CONTINUE
WRITE (LOUT, 555) STRING(IFS:IES)
555 FORMAT (A)
ERR = 1
GO TO 510
450 CONTINUE
NC = NC + 1
IF (NC .LE. IE) GO TO 100
C
500 CONTINUE
IF (NEXPEC .GT. 0 .AND. NFOUND .LT. NEXP) IERR = 2
510 CONTINUE
C
IF (IERR .EQ. 0 .OR. ICARD .LT. 0) RETURN
IF (ICARD .NE. 0) WRITE (LOUT, '(A,13)')
1 '!! ERROR IN DATA STATEMENT NUMBER', ICARD
IF (IERR .EQ. 1)
1 WRITE (LOUT, '(A)') 'SYNTAX ERROR, OR ILLEGAL VALUE'
IF (IERR .EQ. 2) WRITE (LOUT, '(A,I2, A, I2)')
1 ' TOO FEW DATA ITEMS. NUMBER FOUND = ', NFOUND,
2 ' NUMBER EXPECTED = ', NEXPEC
IF (IERR .EQ. 3) WRITE (LOUT, '(A,I2)')
1 ' TOO MANY DATA ITEMS. NUMBER EXPECTED = ', NEXPEC
END
C
FUNCTION IFIRCH(STRING)
C BEGIN PROLOGUE IFIRCH
C DATE WRITTEN  850626
C REVISION DATE  850626
C CATEGORY NO.  M4.
C KEYWORDS CHARACTER STRINGS,SIGNIFICANT CHARACTERS
C AUTHOR CLARK,G.L., GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Determines first significant (non-blank) character
C in character variable
C DESCRIPTION
C---------------------------------------------------------------------------------------------------------------
C IFIRCH locates the first non-blank character in a string of
C arbitrary length. If no characters are found, IFIRCH is set = 0.
C When used with the companion routine IILASCH, the length of a string
C can be determined, and/or a concatenated substring containing the
C significant characters produced.
C---------------------------------------------------------------------------------------------------------------
C REFERENCES (NONE)
C ROUTINES CALLED (NONE)
C END PROLOGUE IFIRCH
C*****precision > double
C IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
C*****END precision > double
C
C*****precision > single
C IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
C****END precision > single
C
CHARACTER* (*) STRING
C
FIRST EXECUTABLE STATEMENT IFIRCH

NLOOP = LEN(STRING)
C
IF (NLOOP .EQ. 0) THEN
  IFIRCH = 0
  RETURN
ENDIF
C
DO 100 I = 1, NLOOP
  IF (STRING(I:I) .NE. ' ') GO TO 120
100 CONTINUE
C
IFIRCH = 0
RETURN
120 CONTINUE
IFIRCH = I
END

FUNCTION ILASCH(STRING)
C  BEGIN PROLOGUE ILASCH
C  DATE WRITTEN 850626
C  REVISION DATE 850626
C  CATEGORY NO. M 4.
C  KEYWORDS CHARACTER STRINGS, SIGNIFICANT CHARACTERS
C  AUTHOR CLARK, G. L., GROUP C-3 LOS ALAMOS NAT'L LAB
C  PURPOSE Determines last significant (non-blank) character
C    in character variable
C  DESCRIPTION
C---------------------------------------------------------------------------------------------
C  IFIRCH locates the last non-blank character in a string of
C  arbitrary length. If no characters are found, ILASCH is set = 0.
C  When used with the companion routine IFIRCH, the length of a string
C  can be determined, and/or a concatenated substring containing the
C  significant characters produced.
C  Note that the FORTRAN intrinsic function LEN returns the length
C  of a character string as declared, rather than as filled. The
C  declared length includes leading and trailing blanks, and thus is
C  not useful in generating 'significant' substrings.
C---------------------------------------------------------------------------------------------
C  REFERENCES (NONE)
C  ROUTINES CALLED (NONE)
C  END PROLOGUE IFIRCH
C---------------------------------------------------------------------------------------------
C****END precision > double

IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
C
C****END precision > double
C
IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
C
CHARACTER* (*) STRING
C
FIRST EXECUTABLE STATEMENT ILASCH

NLOOP = LEN(STRING)
C
IF (NLOOP.EQ.0) THEN
  ILASCH = 0
  RETURN
ENDIF
C
DO 100 I = NLOOP, 1, -1
  IF (STRING(I:I) .NE. ' ') GO TO 120
100 CONTINUE
C
IFIRCH = 0
RETURN
120 CONTINUE
IFIRCH = I
END
SUBROUTINE CKCOMP (1ST, IRAY, II, I)

C
C START PROLOGUE
C
C SUBROUTINE CKCOMP (1ST, IRAY, II, I)*
C Returns the index of an element of a reference character string array which corresponds to a character string; leading and trailing blanks are ignored.
C
C INPUT
C IST - A character string.
Data type - CHARACTER*(*)
C IRAY - An array of character strings;
 dimension IRAY(*) at least II
Data type - CHARACTER*(*)
C II - The length of IRAY.
Data type - integer scalar.
C
C OUTPUT
C I - The first integer location in IRAY in which IST corresponds to IRAY(I); if IST is not also an entry in IRAY, I=0.
C
C END PROLOGUE
C
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
CHARACTER*(*) IST, IRAY(*)
C
I = 0
DO 10 N = II, 1, -1
 IS1 = IFIRCH(IST)
 IS2 = ILASCH(IST)
 IR1 = IFIRCH(IRAY(N))
 IR2 = ILASCH(IRAY(N))
 IF ( IS2.GE.IS1 .AND. IS2.GT.0 .AND.  
 1 IR2.GE.IR1 .AND. IR2.GT.0 .AND.  
 2 ISTAT(IS1:IS2).EQ.IRAY(N)(IR1:IR2) ) I=N
10 CONTINUE
RETURN
END

SUBROUTINE CKUNIT (LINE, AUNITS, EUNITS, IUNITS)
C
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
CHARACTER*(*) LINE, IUNITS, AUNITS, EUNITS
CHARACTER*4 UPCASE
AUNITS = ' '  
EUNITS = ' '  
IUNITS = ' '  
DO 85 N = 1, ILASCH(LINE)-3  
IND = ILASCH(IUNITS)  
IF (EUNITS .EQ. ' ') THEN  
IF (UPCASE(LINE(N:), 4) .EQ. 'CAL/' ) THEN  
EUNITS = 'CAL/'  
ELSE  
IUNITS(IND:) = ', E units cal/mole'  
ENDIF  
ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KCAL') THEN  
EUNITS = 'KCAL'  
ELSE  
IUNITS(IND:) = ', E units Kcal/mole'  
ENDIF  
ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'JOUL') THEN  
EUNITS = 'JOUL'  
ELSE  
IUNITS(IND:) = ', E units Joules/mole'  
ENDIF  
ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KJOU') THEN  
EUNITS = 'KJOU'  
ELSE  
IUNITS(IND:) = ', E units Kjoule/mole'  
ENDIF  
ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KELV') THEN  
EUNITS = 'KELV'  
ELSE  
IUNITS(IND:) = ', E units Kelvins'  
ENDIF  
ENDIF  
ENDIF  
IF (AUNITS .EQ. ' ') THEN  
IF (UPCASE(LINE(N:), 4) .EQ. 'MOLE') THEN  
ELSEIF (UPCASE(LINE(N+4:), 1) .EQ. 'S') THEN  
AUNITS = 'MOLE'  
ELSE  
IUNITS(IND:) = ', A units mole-cm-sec-K'  
ENDIF  
ELSEIF (UPCASE(LINE(N+4:), 4) .EQ. 'MOLC') THEN  
AUNITS = 'MOLC'  
ELSE  
IUNITS(IND:) = ', A units molecules'  
ENDIF  
ENDIF  
ENDIF  
85 CONTINUE  
C  
IF (AUNITS .EQ. ' ') THEN
AUNITS = 'MOLE'
IND = ILASCH(IUNITS) + 1
IF (IND .GT. 1) THEN
  IUNITS(IND:) = ', A units mole-cm-sec-K'
ELSE
  IUNITS(IND:) = ' A units mole-cm-sec-K'
ENDIF
ENDIF
C
IF (EUNITS .EQ. ' ') THEN
  EUNITS = 'CAL/
IND = ILASCH(IUNITS) + 1
IF (IND .GT. 1) THEN
  IUNITS(IND:) = 'E units cal/mole'
ELSE
  IUNITS(IND:) = ' E units cal/mole'
ENDIF
ENDIF
C
RETURN
END
C---------------------------------------------------------------C
C
INTEGER FUNCTION IPPLEN (LINE)
C
BEGIN PROLOGUE
C
FUNCTION IPPLEN (LINE)
C  Returns the effective length of a character string, i.e.,
C  the index of the last character before an exclamation mark (!)
C  indicating a comment.
C
INPUT
LINE - A character string.
C
OUTPUT
IPPLEN - The effective length of the character string.
C
END PROLOGUE
C
******precision > double
IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
******END precision > double
******precision > single
IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
******END precision > single
C
CHARACTER LINE(*)
C
IN = IFIRCH(LINE)
IF (IN.EQ.0 .OR. LINE(IN:IN).EQ.'!') THEN
  IPPLEN = 0
ELSE
  IN = INDEX(LINE,'!!')
  IF (IN .EQ. 0) THEN
    IPPLEN = ILASCH(LINE)
  ELSE
    IPPLEN = ILASCH(LINE(:IN-1))
  ENDIF
ENDIF
RETURN
END
C
CHARACTER*(*) FUNCTION UPCASE(ISTR, ILEN)
CHARACTER ISTR*, ILEN, LCASE(26)*1, UCASE(26)*1
DATA LCASE / 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
           'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z' /


C

UPCASE = ' '
UPCASE = ISTR(:ILEN)
JJ = MIN (LEN(UPCASE), LEN(ISTR), ILEN)
DO 10 J = 1, JJ
   DO 10 N = 1, 26
      IF (ISTR(J:J) .EQ. LCASE(N)) UPCASE(J:J) = UCASE(N)
10 CONTINUE
RETURN
END

SUBROUTINE CKI2CH (NUM, STR, I, KERR)

C
C  START PROLOGUE
C
C  SUBROUTINE CKI2CH (NUM, STR, I, KERR)
C  Returns a character string representation of an integer
C  and the effective length of the string.
C
C  INPUT
C  NUM - A number to be converted to a character string;
C         the maximum magnitude of NUM is machine-dependent.
C         Data type - integer scalar.
C
C  OUTPUT
C  STR - A left-justified character string representing NUM
C         Data type - CHARACTER(*)
C  I   - The effective length of the character string
C         Data type - integer scalar
C  KERR - Error flag; character length errors will result in
C         KERR=.TRUE.
C         Data type - logical
C
C  END PROLOGUE
C
C  IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
C  END
C
C  IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
C  END
C
C  CHARACTER STR(*),  IST(10)*(1)
C  LOGICAL KERR
C  DATA IST/'0', '1', '2', '3', '4', '5', '6', '7', '8', '9' /
C  BIGI = 2147483647.
C
C  I = 0
C  STR = ' '
C  ILEN = LEN(STR)
C  KERR = .FALSE.
C  IF (ILEN.LT.1 .OR. ABS(NUM).GT.BIGI) THEN
C     KERR = .TRUE.
C     RETURN
C  ENDIF
C
C  IF (NUM .EQ. 0) THEN
C     STR = '0'
C     I = 1
C     RETURN
C  ELSEIF (NUM .LT. 0) THEN
C     STR(1 :) = ' -'
C  ENDIF
INUM = ABS(NUM)
NCOL = NINT(LOG10(REAL(INUM))) + 1

DO 10 J = NCOL, 1, -1
   IDIV = INUM / 10.0**(J-1)
   IF (J.EQ.NCOL .AND. IDIV.EQ.0) GO TO 10
   LT = ILASCH(STR)
   IF (LT .EQ. ILEN) THEN
      STR = ' ';
      KERR = .TRUE.
      RETURN
   ENDIF
   STR(LT+1:) = IST(IDIV+1)
   INUM = INUM - IDIV*10.0**(J-1)
10 CONTINUE
   I = ILASCH(STR)
RETURN
END

SUBROUTINE THERM_IO(ERROR)
IMPLICIT NONE
INCLUDE 'THM_PAR3.FI'
CHARACTER FILE_NAME*20
INTEGER IDOT, ISTART, IFIRCH
LOGICAL ERROR
ERROR = .FALSE.
5 WRITE(*,10)
10 FORMAT(' Please enter the input file name : ') READ(*,'(A20)')FILE_INP
OPEN(INP,FILE=FILE_INP,STATUS='OLD',ERR=30)
GO TO 50
30 CONTINUE
WRITE(*,40) FILE_INP
40 FORMAT(' Can not find file : ',A20)
GO TO 5
50 CONTINUE
ERROR = .FALSE.
55 WRITE(*,110)
110 FORMAT(' Please enter the output sorted file name : ') READ(*,'(A20)')FILE_OUT
OPEN(LOUT,FILE=FILE_OUT,STATUS='UNKNOWN',ERR=130)
CLOSE(LOUT,status='delete')
OPEN(LOUT,FILE=FILE_OUT,STATUS='NEW',ERR=130)
GO TO 150
130 CONTINUE
WRITE(*,140)FILE_OUT
140 FORMAT(' File I/O Error : ',A20)
GO TO 350
150 CONTINUE
FILE_LOG = 'THERMSRT.LOG'
OPEN(LOG,FILE=FILE_LOG,STATUS='UNKNOWN',ERR=230)
CLOSE(LOG,status='delete')
OPEN(LOG,FILE=FILE_LOG,STATUS='NEW',ERR=230)
GO TO 450
230 CONTINUE
WRITE(*,240)FILE_LOG
240 FORMAT(' File I/O Error : ',A20)
ERROR = .TRUE.
GO TO 450
350 CONTINUE
ERROR = .TRUE.
450 CONTINUE
C
RETURN
END
D.3 THERMCVT

PROGRAM THERMCVT
C**************************************changed W.Ing 06-15-94, 08-01-95**************************************
C IMPLICIT NONE
C include 'thm_par2.fi'
C include 'thm_cfg.fi'
C CHARACTER specy_name*70, specy_formula*40,
  specy_group(Max_Specy_Group)*14,KEY2(3)*4,
  specy_element(Max_Element)*4, str_int*8,
  specy_comment*70, specy_phase, line*100, KEY(5)*5,
  SUB(20)*100, UPCASE*5, str*100, qty*100, NAME*200
C INTEGER specy_group_qty(Max_Specy_Group), specy_symmetry,
  specy_element_qty(Max_Element), Ngroup,ILASCH,
  ILEN, IPPLEN, NSUB, NKEY, Isymmetry(5), NVAL, Ierr,
  I, istr, Igroup(5), Ifirch, Islash, Igty(5), Ifound,J,
  Index, Itotal, Iname, NKEY2, Nloop
C REAL*8 specy_Hf, specy_S, specy_Cp300, specy_Cp400, specy_Cp500,
  specy_Cp600, specy_Cp800, specy_Cp1000, specy_Cp1500,
  specy_rotor, rotor(5)
C LOGICAL therm, nasa, err, io_err
C DATA KEY/'UNITS','RADIC','XXX','END','!'/
C DATA KEY2/'SYMM','NROT','ENDS'/
C Call Initialization subroutines, 02/95 ING
C CALL TITLE
CALL THERM_IO(io_err)
if (io_err) GO TO 6000
CALL THERM_INIT(init_err)
if (init_err) GO TO 6000
write(lst,*)'THERM BATCH INPUT FILE'
write(lst,50)' SPECIES Hf S Cp 300 400 500 600'
write(lst,10)10 800 1000 1500 DATE ELEMENTS'

CONTINUE
C
specy_rotor = 0.0
specy_symmetry = 1
LINE = ' '
READ (INF,'(A)',END=5000) LINE
CONTINUE
ILEN = IPPLEN(LINE)
IF (ILEN .EQ. 0) GO TO 100
READ (INF,'(A)',END=5000) LINE
READ (INF,'(A)',END=5000) LINE
READ (INF,'(A)',END=5000) LINE
ILEN = IPPLEN(LINE)
CALL CKISUB (LINE(:ILEN), SUB, NSUB)
specy_name = SUB(1)
specy_formula = SUB(2)

C
READ (INF,'(A)',END=5000) LINE
ILEN = IPPLEN(LINE)
CALL CKISUB (LINE(:ILEN), SUB, NSUB)

C
IS THERE A KEYWORD?

C
CALL CKCOMPO (UPCASE(SUB(1), 5), KEY, 2, NKEY)
WRITE('**',UPCASE(SUB(1),3),KEY(1),NKEY)
IF (NKEY GT 0) ITASK = 0
WRITE('**',NKEY)
IF (NKEY.EQ.2) THEN
READ (INF,'(A)',END=5000) LINE
READ (INF,'(A)',END=5000) LINE
READ (INF,'(A)',END=5000) LINE
ELSEIF (NKEY.EQ.1) THEN
   C DO NOTHING FOR MOLECULES
ELSE
   C ERROR FOR NEITHER MOLECULES NOR RADICALS
   GO TO 6000
ENDIF
READ (INF,'(A)',END=5000) LINE
ILEN = IPPLEN(LINE)
CALL CKISUB (LINE(:ILEN), SUB, NSUB)
call IPPARI (SUB(2), -1, 1, Igroup, nval, Ierr, LOG)
Ngroup = Igroup(1)
IF (Ierr.NE.0 .OR. Ngroup.EQ.0) THEN
   160 CONTINUE
   WRITE('**', specy ' ,specy_name ,'' group number error !!!'
   READ (INF,'(A)',END=5000) LINE
   ILEN = IPPLEN(LINE)
   CALL CKISUB (LINE(:ILEN), SUB, NSUB)
   CALL CKCOMP (UPCASE(SUB(1), 4), KEY2, 3, NKEY2)
   IF (NKEY2.NE.3) GO TO 160
   GO TO 100
ENDIF
C GO TO 6000
READ (INF,'(A)',END=5000) LINE
IF (Ngroup.LE.5) THEN
   DO 200 I = 1, Ngroup
      READ (INF,'(A)',END=5000) LINE
      ILEN = IPPLEN(LINE)
      CALL CKISUB (LINE(:ILEN), SUB, NSUB)
      specy_group(I) = SUB(3)
      call IPPARI (SUB(5), -1, 1, Iqty, nval, Ierr, LOG)
      specy_group_qty(I) = Iqty(1)
   200 CONTINUE
ELSE
   Nloop = INT(Ngroup/2.0)
   WRITE('**','Ngroup = ',Ngroup,' Nloop = ',Nloop)
   IF(MOD(Ngroup,2).EQ.0) THEN
      DO 210 I = 1, Nloop
         READ (INF,'(A)',END=5000) LINE
         ILEN = IPPLEN(LINE)
         CALL CKISUB (LINE(:ILEN), SUB, NSUB)
         specy_group(I) = SUB(3)
         call IPPARI (SUB(5), -1, 1, Iqty, nval, Ierr, LOG)
         specy_group_qty(I) = Iqty(1)
         specy_group(specy_group(I),specy_group_qty(I))
      210 CONTINUE
      IF(MOD(Nloop,2).EQ.0) THEN
         CALL CKISUB (LINE(:ILEN), SUB, NSUB)
         specy_group(I) = SUB(3)
         call IPPARI (SUB(5), -1, 1, Iqty, nval, Ierr, LOG)
         specy_group_qty(specy_group(I)) = Iqty(I)
      ELSE
         CALL CKISUB (LINE(:ILEN), SUB, NSUB)
         specy_group(I) = SUB(3)
         call IPPARI (SUB(5), -1, 1, Iqty, nval, Ierr, LOG)
         specy_group_qty(specy_group(I)) = Iqty(I)
      ENDIF
   ELSE
      CALL CKISUB (LINE(:ILEN), SUB, NSUB)
      specy_group(I) = SUB(3)
      call IPPARI (SUB(5), -1, 1, Iqty, nval, Ierr, LOG)
      specy_group_qty(specy_group(I)) = Iqty(I)
   ENDIF
C
C
CONTINUE

ELSE

DO 220 I = 1, Nloop

READ (INP, 'A', END=5000) LINE
ILEN = IPPLEN(LINE)
CALL CKISUB (LINE(:ILEN), SUB, NSUB)
spcyc_group(I) = SUB(3)
call IPPARI (SUB(5), -1, 1, Iqty, nval, ierr, LOG)
spcyc_group_qty(I) = Iqty(I)
spcyc_group(Nloop+I+1) = SUB(9)
call IPPARI (SUB(11), -1, 1, Iqty, nval, ierr, LOG)
spcyc_group_qty(Nloop+I+1) = Iqty(I)

220 CONTINUE

READ (INP, 'A', END=5000) LINE
ILEN = IPPLEN(LINE)
CALL CKISUB (LINE(:ILEN), SUB, NSUB)
spcyc_group(Nloop+1) = SUB(3)
call IPPARI (SUB(5), -1, 1, Iqty, nval, ierr, LOG)
spcyc_group_qty(Nloop+1) = Iqty(I)

ENDIF

ENDIF

WRITE(LST,*) LINE
READ (INP, 'A', END=5000) LINE
READ (INP, 'A', END=5000) LINE
READ (INP, 'A', END=5000) LINE

250 CONTINUE

READ (INP, 'A', END=5000) LINE
ILEN = IPPLEN(LINE)
CALL CKISUB (LINE(:ILEN), SUB, NSUB)
CALL CKCOMP (UPCASE(SUB(1), 4), KEY2, 3, NKEY2)
WRITE(*,*) UPCASE(SUB(1), 4)
IF (NKEY2.EQ.0) GO TO 250
IF (NKEY2.EQ.3) GO TO 280
IF (NKEY2.EQ.2) THEN
call IPPARI (SUB(2), -1, 1, rotor, nval, ierr, LOG)
IF (Ierr.NE.0) GO TO 6000
specy_rotor = rotor(1)
READ (INP, 'A', END=5000) LINE
CALL CKISUB (LINE(:ILEN), SUB, NSUB)
call IPPARI (SUB(2), -1, 1, Isymmetry, nval, ierr, LOG)
IF (Ierr.NE.0) GO TO 6000
specy_symmetry = Isymmetry(I)
ELSE
call IPPARI (SUB(2), -1, 1, Isymmetry, nval, ierr, LOG)
WRITE(*,*) line
WRITE(*,*) sub(2)
IF (Ierr.NE.0) GO TO 6000
specy_symmetry = Isymmetry(I)
ENDIF
GO TO 250

C

280 CONTINUE

write(LST,300) specy_name, specy_formula,
2 specy_symmetry, specy_rotor
300 format(A12, ',', A12, ',', I4, ',', F4.1)
write(LST,300) specy_name(:ILASCH(specy_name)),
1 specy_formula(:ILASCH(specy_name)),
2 specy_symmetry, specy_rotor
300 format(A, ',', A, ',', I3, ',', F4.1)

Index = 1
Itotal = 0
DO 350 J = 1, Ngroup
Itotal = Itotal + 1
NAME(Itotal:Itotal) = '
Index = Itotal + 1
Iname = IPPLEN(specy_group(J))
Itotal = Itotal + Iname
NAME(Index:Itotal) = specy_group(J)
NAME(Itotal+1:Itotal+1) = '
CALL CKI2CH(specy_group_qty(J), str_int, Iname, err)
NAME(Itotal+2:Itotal+Iname+1) = str_int
Itotal = Itotal + 1 + Iname

350 CONTINUE
ILEN = IPPLEN(NAME)
c
write(*,*).'..Itotal = ',Itotal
write(lst,360)NAME(:Itotal)
360 format(' ',A)
c
write(LST,310)(specy_group(J)(:ILASCH(specy_GROUP(J))),
c 1 specy_group_qty(J)(:ILASCH(specy_GROUP(J))),J=1,Ngroup)
310 format(' ',A,'*',I)

dump useless specy-end info for molecules and radicals

C IF (NKEY.EQ.2) THEN
C READ (INF,'(A)',END=5000) LINE
C READ (INF,'(A)',END=5000) LINE
C READ (INF,'(A)',END=5000) LINE
C READ (INF,'(A)',END=5000) LINE
C READ (INF,'(A)',END=5000) LINE
C ELSE
C READ (INF,'(A)',END=5000) LINE
C READ (INF,'(A)',END=5000) LINE
C ENDIF
C
go to 100
5000 write(*,*)'...end of conversion...'
WRITE(LST,5100)
5100 format('END')
c
write(*,*)' new ThermCal input file created as - ',File_lst
write(*,*)' go to 7000
6000 write(*,*)'...error in input file...'
7000 continue
end
C----------------------------------------------------------------------------------------------------------------C
SUBROUTINE CKI2CH(LINE, SUB, NSUB)
C
C Generates an array of CHAR(*) substrings from a CHAR(*) string,
C using blanks or tabs as delimiters
C
C Input: LINE - a CHAR(*) line
C Output: SUB - a CHAR(*) array of substrings
C NSUB - number of substrings found
C A '!' will comment out a line, or remainder of the line.
C F. Rupley, Div. 8245, 5/15/86
C----------------------------------------------------------------------------------------------------------------C
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
CHARACTER(*) SUB(*), LINE
NSUB = 0
C
DO 5 N = 1, LEN(LINE)
  IF (ICHAR(LINE(N:N)) .EQ. 9) LINE(N:N) = ' ' 
  5 CONTINUE
IF (IPPLEN(LINE) .LE. 0) RETURN

C    ILEN = ILASCH(LINE)
C
NSTART = IFIRCH(LINE)
10 CONTINUE
ISTART = NSTART
NSUB = NSUB + 1
SUB(NSUB) = ' '

DO 100 I = ISTART, ILEN
      ILAST = INDEX(LINE(ISTART:), ' ') - 1
      IF (ILAST .GT. 0) THEN
             ILAST = ISTART + ILAST - 1
      ELSE
             ILAST = ILEN
      ENDIF
      SUB(NSUB) = LINE(ISTART:ILAST)
      IF (ILAST .EQ. ILEN) RETURN

NSTART = ILAST + IFIRCH(LINE(ILAST+1:))

C Does SUB have any slashes?
C
I1 = INDEX(SUB(NSUB), '-')
IF (I1 .LE. 0) THEN
      IF (LINE(NSTART:NSTART) .NE. '-') GO TO 10
      NEND = NSTART + INDEX(LINE(NSTART+1:), '-')
      IND = INDEX(SUB(NSUB), ' ') + 1
      SUB(NSUB)(IND:) = LINE(NSTART:NEND)
      IF (NEND .EQ. ILEN) RETURN
      NSTART = NEND + IFIRCH(LINE(NEND+1:))
      GO TO 10
ENDIF

C Does SUB have 2 slashes?
C
I2 = INDEX(SUB(NSUB)(I1+1:), '-')
IF (I2 .GT. 0) GO TO 10

NEND = NSTART + INDEX(LINE(NSTART+1:), '-')
IND = INDEX(SUB(NSUB), ' ') + 1
SUB(NSUB)(IND:) = LINE(NSTART:NEND)
IF (NEND .EQ. ILEN) RETURN
NSTART = NEND + IFIRCH(LINE(NEND+1:))
GO TO 10

100 CONTINUE
RETURN
END

C-----------------------------------------------------------------------
SUBROUTINE IPNPAR (LINE, NPAR, IPAR, ISTART)

C Returns CHAR(*) IPAR substring of CHAR(*) string LINE which
contains NPAR real parameters

C Input: LINE - a CHAR(*) line
NPAR - number of parameters expected
C
C Output: IPAR - the substring of parameters only
ISTART - the starting location of IPAR substring
A '!' will comment out a line, or remainder of the line.

F. Rupley, Div. 8245, 5/14/86

C-----------------------------------------------------------------------

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)**

*******END precision > single*****

**CHARACTER(*) LINE,IPAR**

----Find Comment String (! signifies comment)----

**ILEN = IPPLEN(LINE)**

**ISTART = 0**

**N = 0**

**IF (ILEN.GT.0) THEN**

**DO 40 I = ILEN, 1, -1**

**ISTART = I**

**IPAR = ' '**

**IPAR = LINE(ISTART:ILEN)**

**IF (LINE(I:1).NE.' ') THEN**

**IF (I .EQ. 1) RETURN**

**IF (LINE(I-1:I-1) .EQ. ' ') THEN**

**N = N + 1**

**IF (N .EQ. NPAR) RETURN**

**ENDIF**

40 **CONTINUE**

**ENDIF**

**RETURN**

**END**

---SUBROUTINE IPPARI(STRING, ICARD, NEXPEC, IVAL, NFOUND, IERR, LOUT)---

**BEGIN PROLOGUE IPPARI**

**REFER TO IPGETI**

**DATE WRITTEN 850625 (YYMDDD)**

**REVISION DATE 851725 (YYMDDD)**

**CATEGORY NO. J3.,J4.,M2.**

**KEYWORDS PARSE**

**AUTHOR CLARK,G.L..GROUP C-3 LOS ALAMOS NAT'L LAB**

**PURPOSE Parses integer variables from a character variable. Called by IPGETI, the IOPAK routine used for interactive input.**

**DESCRIPTION**

---IPPARI may be used for parsing an input record that contains integer values, but was read into a character variable instead of directly into integer variables. The following benefits are gained by this approach:

- specification of only certain elements of the array is allowed, thus letting the others retain default values
- variable numbers of values may be input in a record, up to a specified maximum
- control remains with the calling program in case of an input error
- diagnostics may be printed by IPPARI to indicate the nature of input errors

The contents of STRING on input indicate which elements of IVAL are to be changed from their entry values, and values to which they should be changed on exit. Commas and blanks serve as delimiters, but multiple blanks are treated as a single delimiter.

Thus, an input record such as:

' 1, 2, 40000 ,60 '

is interpreted as the following set of instructions by IPGETI:

- (1) set IVAL(1) = 1
- (2) set IVAL(2) = 2
- (3) leave IVAL(3) unchanged
- (4) set IVAL(4) = 40000
- (5) leave IVAL(5) unchanged
IPPARI will print diagnostics on the default output device, if desired.

IPPARI is part of IOPAK, and is written in ANSI FORTRAN 77

Examples:

Assume IVAL = (0, 0, 0) and NEXPEC = 3 on entry:

<table>
<thead>
<tr>
<th>input string</th>
<th>IVAL on exit</th>
<th>IERR</th>
<th>NFOUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>'2, 3 45'</td>
<td>(2, 3, 45)</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>'2.15, 3'</td>
<td>(2, 0, 3)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>'3X, 25, 2'</td>
<td>(0, 0, 0)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>'10000'</td>
<td>(10000, 0, 0)</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Assume IVAL = (0, 0, 0, 0) and NEXPEC = -4 on entry:

<table>
<thead>
<tr>
<th>input string</th>
<th>IVAL on exit</th>
<th>IERR</th>
<th>NFOUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>'1, 2'</td>
<td>(1, 2)</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>'37, 400'</td>
<td>(0, 0, 37, 400)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>'1, -3, 5'</td>
<td>(1, 0, -3, 0)</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

arguments: (I=input, O=output)

STRING (I) - the character string to be parsed.

ICARD (I) - data statement number, and error processing flag
< 0 : no error messages printed
  = 0 : print error messages, but not ICARD
  > 0 : print error messages, and ICARD

NEXPEC (I) - number of real variables expected to be input. If
< 0, the number is unknown, and any number of values
between 0 and abs(nexpec) may be input. (see NFOUND)

PROMPT (I) - prompting string, character type. A question
mark will be added to form the prompt at the screen.

IVAL (I, O) - the integer value or values to be modified. On entry,
the values are printed as defaults. The formal parameter
corresponding to IVAL must be dimensioned at least NEXPEC
in the calling program if NEXPEC > 1.

NFOUND (O) - the number of real values represented in STRING,
only in the case that there were as many or less than
NEXPEC.

IERR (O) - error flag:
  = 0 if no errors found
  = 1 syntax errors or illegal values found
  = 2 for too few values found (NFOUND < NEXPEC)
  = 3 for too many values found (NFOUND > NEXPEC)

REFERENCES (NONE)

ROUTINES CALLED IFIRCH, ILASCH

END PROLOGUE IPPARI

* * * * * precision > double
IMPLICIT DOUBLE PRECISION (A-H, O-Z), INTEGER (I-N)
* * * END precision > double

* * * precision > single
C IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
C*****END precision > single
C
C CHARACTER STRING(*), ITEMP*80
DIMENSION IVAL(*)
CHARACTER *8 FMT(14)
LOGICAL OKINCR
C
C FIRST EXECUTABLE STATEMENT IPPARI
IERR = 0
NFOUND = 0
NEXP = IABS(NEXPEC)
IE = ILASCH(STRING)
IF (IE .EQ. 0) GO TO 500
NC = 1
C
C OKINCR is a flag that indicates it's OK to increment
C NFOUND, the index of the array into which the value
C should be read. It is set false when a space follows
C an integer value substring, to keep incrementing from
C occurring if a comma should be encountered before the
C next value.
C
OKINCR = .TRUE.
C
C begin overall loop on characters in string
C 100 CONTINUE
C
IF (STRING(NC:NC) .EQ. ',') THEN
  IF (OKINCR .OR. NC .EQ. IE) THEN
    NFOUND = NFOUND + 1
  ELSE
    OKINCR = .TRUE.
  ENDIF
C
GO TO 450
ENDIF
IIF (STRING(NC:NC) .EQ. ' ') GO TO 450
C
C first good character (non-delimeter) found - now find
C last good character
C
IBS = NC
160 CONTINUE
NC = NC + 1
IF (NC .GT. IE) GO TO 180
IF (STRING(NC:NC) .EQ. ' ') THEN
  OKINCR = .FALSE.
ELSEIF (STRING(NC:NC) .EQ. ',') THEN
  OKINCR = .TRUE.
ELSE
  GO TO 160
ENDIF
C
C end of substring found - read value into integer array
C
180 CONTINUE
NFOUND = NFOUND + 1
IF (NFOUND .GT. NEXP) THEN
  IERR = 3
  GO TO 500
ENDIF
C
IES = NC - 1
NCH = IES - IBS + 1
DATA FMT/'(I1)', 'I2', 'I3', 'I4', 'I5',
1 'I6', 'I7', 'I8', 'I9', 'I10',
2 'I11', 'I12', 'I13', 'I14'/
ITEMP = ' '
ITEMP = STRING(IBS:IES)
READ (ITEMP(1:NCH), FMT(NCH), ERR = 400) IVAL(NFOUND)
GO TO 450
400 CONTINUE
IERR = 1
GO TO 510
450 CONTINUE
NC = NC + 1
IF (NC .LE. IE) GO TO 100
C
500 CONTINUE
IF (NEXPEC .GT. 0 .AND. NFOUND .LT. NEXP) IERR = 2
510 CONTINUE
C
IF (IERR .EQ. 0 .OR. ICARD .LT. 0) RETURN
IF (ICARD .NE. 0) WRITE (LOUT, ' (A,L13) ')
1 ' ! ! ERROR IN DATA STATEMENT NUMBER', ICARD
IF (IERR .EQ. 1)
1 WRITE (LOUT, ' (A) ' ) ' SYNTAX ERROR, OR ILLEGAL VALUE'
IF (IERR .EQ. 2)
1 WRITE (LOUT, ' ( A , 1 2 ) ' ) ' TOO FEW DATA ITEMS. NUMBER FOUND = ', NFOUND,
2 ' NUMBER EXPECTED = ', NEXPEC
IF (IERR .EQ. 3) WRITE (LOUT, ' ( A ) ' ) ' TOO MANY DATA ITEMS. NUMBER EXPECTED = ', NEXPEC
END
C
SUBROUTINE IPPARR(STRING, ICARD, NEXPEC, RVAL, NFOUND, IERR, LOUT)
C  BEGIN PROLOGUE IPPARR
C  REFER TO IPGETR
C  DATE WRITTEN 850625 (YYMDD)
C  REVISION DATE 851625 (YYMDD)
C  KEYWORDS PARSE
C  AUTHOR CLARK, G.L., GROUP C-3 LOS ALAMOS NAT'L LAB
C  PURPOSE Parses real variables from a character variable. Called
C  by IPGETR, the IOPAK routine used for interactive input.
C  DESCRIPTION
C-----------------------------------------------------------------------------------------------------------------C
IPPARR may be used for parsing an input record that contains real
C values, but was read into a character variable instead of directly
C into real variables.
C The following benefits are gained by this approach:
C - specification of only certain elements of the array is allowed, 
C  thus letting the others retain default values 
C - variable numbers of values may be input in a record, up to a
C  specified maximum 
C - control remains with the calling program in case of an input
C  error 
C - diagnostics may be printed by IPPARR to indicate the nature
C  of input errors 
C
C The contents of STRING on input indicate which elements of RVAL
C are to be changed from their entry values, and values to which
C they should be changed on exit. Commas and blanks serve as
C delimiters, but multiple blanks are treated as a single delimiter.
C Thus, an input record such as:
C   ' 1., 2., 4.e-5 , 6.e-6'
C is interpreted as the following set of instructions by IPGETR:
C
C (1) set RVAL(1) = 1.0
(2) set RVAL(2) = 2.0
(3) leave RVAL(3) unchanged
(4) set RVAL(4) = 4.0E-05
(5) leave RVAL(5) unchanged
(6) set RVAL(6) = 6.0E-06

IPPARR will print diagnostics on the default output device, if desired.

IPPARR is part of IOPAK, and is written in ANSI FORTRAN 77

Examples:

Assume RVAL = (0., 0., 0.) and NEXPEC = 3 on entry:

input string  RVAL on exit    IERR  NFOUND
-------------  ---------------  -----  -----  
'2.34e-3,  3 45.1' (2.34E-03, 3.0, 45.1)  0  3
'2,,3.5'      (2.0, 0.0, 3.0E-05)  0  3
'1.4,0.028E4' (0.0, 1.4, 280.0)   0  3
'1.0, 2.84, 3.0' (1.0, 0.0, 0.0)   1  1
'1.0'         (1.0, 0.0, 0.0)   2  1

Assume RVAL = (0.,0.,0.,0.) and NEXPEC = -4 on entry:

input string  RVAL on exit    IERR  NFOUND
-------------  ---------------  -----  -----  
'1,2.'        (1.0, 2.0)      0  2
'1,3, 4.0'    (0.0, 0.0, 3.0, 4.0) 0  4
'1,3,,5.0'    (0.0, 0.0, 3.0, 0.0) 3  4

arguments: (I=Input,O=Output)
-------------
STRING (I) - the character string to be parsed.

ICARD (I) - data statement number, and error processing flag
< 0 : no error messages printed
= 0 : print error messages, but not ICARD
> 0 : print error messages, and ICARD

NEXPEC (I) - number of real variables expected to be input. If
< 0, the number is unknown, and any number of values
between 0 and abs(nexpec) may be input. (see NFOUND)

PROMPT (I) - prompting string, character type. A question
mark will be added to form the prompt at the screen.

RVAL (I,O) - the real value or values to be modified. On entry,
the values are printed as defaults. The formal parameter
receiving this parameter must be dimensioned at least NEXPEC
in the calling program if NEXPEC > 1.

NFOUND (0) - the number of real values represented in STRING,
only in the case that there were as many or less than
NEXPEC.

IERR (0) - error flag:
= 0 if no errors found
= 1 syntax errors or illegal values found
= 2 for too few values found (NFOUND < NEXPEC)
= 3 for too many values found (NFOUND > NEXPEC)

REFERENCES (NONE)

ROUTINES CALLED IFIRCH, IILASCH

END PROLOGUE IPPARR
C***** precision > double
IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
C***** END precision > double
C
C***** precision > single
C IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
C***** END precision > single
C
CHARACTER STRING(*), ITEMP*80
DIMENSION RVAL(*)
CHARACTER *8 FMT(22)
LOGICAL OKINCR
C
C FIRST EXECUTABLE STATEMENT IPPARR
IERR = 0
NFOUND = 0
NEXP = IABS(NEXPEC)
IE = ILASCH(STRING)
IF (IE .EQ. 0) GO TO 500
NC = 1
C
C OKINCR is a flag that indicates it's OK to increment
C NFOUND, the index of the array into which the value
C should be read. It is set negative when a space follows
C a real value substring, to keep incrementing from
C occurring if a comma should be encountered before the
C next value.
C
OKINCR = .TRUE.
C
C begin overall loop on characters in string
C
100 CONTINUE
C
IF (STRING(NC:NC) .EQ. ',') THEN
IF (OKINCR) THEN
NFOUND = NFOUND + 1
ELSE
OKINCR = .TRUE.
ENDIF
C
GO TO 450
ENDIF
IF (STRING(NC:NC) .EQ. ' ') GO TO 450
C
C first good character (non-delimiter) found - now find
C last good character
C
160 CONTINUE
NC = NC + 1
IF (NC .GT. IE) GO TO 180
IF (STRING(NC:NC) .EQ. ',') THEN
OKINCR = .FALSE.
ELSEIF (STRING(NC:NC) .EQ. ',') THEN
OKINCR = .TRUE.
ELSE
GO TO 160
ENDIF
C
C end of substring found - read value into real array
C
180 CONTINUE
NFOUND = NFOUND + 1
IF (NFOUND .GT. NEXP) THEN
IERR = 3
GO TO 500
ENDIF
C
DATA FMT/ ' (E1.0)' , ' (E2.0)' , ' (E3.0)' , ' (E4.0)' ,
1 ' (E5.0)' , ' (E6.0)' , ' (E7.0)' , ' (E8.0)' , ' (E9.0)' ,
2 ' (E10.0)' , ' (E11.0)' , ' (E12.0)' , ' (E13.0)' , ' (E14.0)' ,
3 ' (E15.0)' , ' (E16.0)' , ' (E17.0)' , ' (E18.0)' , ' (E19.0)' ,
4 ' (E20.0)' , ' (E21.0)' , ' (E22.0)' /
IES = NC - 1
NCH = IES - IBS + 1
ITEM = ''
ITEM = STRING(IBS:IES)
READ (ITEMP(1:NCH), FMT(NCH), ERR = 400) RVAL(NFOUND)
GO TO 450

400 CONTINUE
WRITE (LOUT, 555) STRING(IBS:IES)

555 FORMAT (A)
IERR = 1
GO TO 510

450 CONTINUE
NC = NC + 1
IF (NC .LE. IE) GO TO 100
C
500 CONTINUE
IF (NEXPEC .GT. 0 .AND. NFOUND .LT. NEXP) IERR = 2
510 CONTINUE
C
IF (IERR .EQ. 0 .OR. ICARD .LT. 0) RETURN
IF (ICARD .NE. 0) WRITE (LOUT, '(A,13)')
1 '!!  ERROR IN DATA STATEMENT NUMBER', ICARD
IF (IERR .EQ. 1)
1 WRITE (LOUT, '(A)') 'SYNTAX ERROR, OR ILLEGAL VALUE'
IF (IERR .EQ. 2)
1 WRITE (LOUT, '(A,12, A, 12)') 'TOO FEW DATA ITEMS. NUMBER FOUND = ', NFOUND,
2 ' NUMBER EXPECTED = ', NEXPEC
IF (IERR .EQ. 3)
1 WRITE (LOUT, '(A,12)') 'TOO MANY DATA ITEMS. NUMBER EXPECTED = ', NEXPEC
END
C
FUNCTION IFIRCH(STRING)
C BEGIN PROLOGUE IFIRCH
C DATE WRITTEN 850626
C REVISION DATE 850626
C CATEGORY NO. M4.
C KEYWORDS CHARACTER STRINGS,SIGNIFICANT CHARACTERS
C AUTHOR CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Determines first significant (non-blank) character
C in character variable
C DESCRIPTION
C------------------------------------------------------------------
C IFIRCH locates the first non-blank character in a string of
C arbitrary length. If no characters are found, IFIRCH is set = 0.
C When used with the companion routine ILASCH, the length of a string
C can be determined, and/or a concatenated substring containing the
C significant characters produced.
C------------------------------------------------------------------
C C REFERENCES (NONE)
C C ROUTINES CALLED (NONE)
C C END PROLOGUE IFIRCH
C*****precision > double
IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
C*****END precision > double
C C*****precision > single

C
**IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)**

**END precision > single**

**CHARACTER* (*) STRING**

**FIRST EXECUTABLE STATEMENT IFIRCH**

```
NLOOP = LEN(STRING)
```

**IF** (NLOOP .EQ. 0) **THEN**

```
IFIRCH = 0
RETURN
ENDIF
```

**DO 100 I = 1, NLOOP**

```
IF (STRING(I:I) .NE. ' ') GO TO 120
```

**CONTINUE**

```
IFIRCH = 0
RETURN
```

**CONTINUE**

```
IFIRCH = I
END
```

**FUNCTION ILASCH(STRING)**

**BEGIN PROLOGUE ILASCH**

**DATE WRITTEN** 850626

**REVISION DATE** 850626

**CATEGORY NO.** M4.

**KEYWORDS** CHARACTER STRINGS, SIGNIFICANT CHARACTERS

**AUTHOR** CLARK,G.L., GROUP C-3 LOS ALAMOS NAT'L LAB

**PURPOSE** Determines last significant (non-blank) character

**in character variable**

**DESCRIPTION**

```
----------------------------------------------------------------------------------------------------------------
C IFIRCH locates the last non-blank character in a string of
C arbitrary length. If no characters are found, ILASCH is set = 0.
C When used with the companion routine IFIRCH, the length of a string
C can be determined, and/or a concatenated substring containing the
C significant characters produced.
C Note that the FORTRAN intrinsic function LEN returns the length
C of a character string as declared, rather than as filled. The
C declared length includes leading and trailing blanks, and thus is
C not useful in generating 'significant' substrings.
----------------------------------------------------------------------------------------------------------------
```

**REFERENCES** (NONE)

**ROUTINES CALLED** (NONE)

**END PROLOGUE ILASCH**

**IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)**

**END precision > double**

**IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)**

**END precision > single**

**CHARACTER* (*) STRING**

**FIRST EXECUTABLE STATEMENT ILASCH**

```
NLOOP = LEN(STRING)
```

**IF** (NLOOP .EQ. 0) **THEN**

```
ILASCH = 0
RETURN
ENDIF
```

**DO 100 I = NLOOP, 1, -1**
IF (STRING(I:I) .NE. ' ') GO TO 120

100 CONTINUE
C
120 CONTINUE
ILASCH = I
END

SUBROUTINE CKCOMP (IST, IRAY, II, I)

START PROLOGUE

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 - A character string.
Data type - CHARACTER(*)
IRAY - An array of character strings;
dimension IRAY(*) at least II
Data type - CHARACTER(*)
II - The length of IRAY.
Data type - integer scalar.

OUTPUT
I - The first integer location in IRAY in which IST
corresponds to IRAY(I); if IST is not also an
entry in IRAY, I=0.

END PROLOGUE

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
CHARACTER(*) IST, IRAY(*)
C
I = 0
DO 10 N = II, 1, -1
IS1 = IFIRCH(IST)
IS2 = ILASCH(IST)
IR1 = IFIRCH(IRAY(N))
IR2 = ILASCH(IRAY(N))
1 IF ( IS2.GE.IS1 .AND. IS2.GT.0 .AND.
1 IR2.GE.IR1 .AND. IR2.GT.0 .AND.
2 IST(IS1:IS2).EQ.IRAY(N)(IR1:IR2) ) I=N
10 CONTINUE
RETURN
END

SUBROUTINE CKUNIT (LINE, AUNITS, EUNITS, IUNITS)

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
CHARACTER(*) LINE, AUNITS, EUNITS
CHARACTER*4 UPCase

AUNITS = ' '
EUNITS = ' '
IUNITS = ' '
DO 85 N = 1, ILASCH(LINE)-3
   IND = ILASCH(IUNITS)
   IF (EUNITS .EQ. ' ') THEN
       IF (UPCASE(LINE(N:), 4) .EQ. 'CAL/') THEN
           EUNITS = 'CAL'/
           IF (IUNITS .EQ. ' ') THEN
               IUNITS = 'E units cal/mole'
           ELSE
               IUNITS(IND:) = ', E units cal/mole'
           ENDIF
       ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KCAL') THEN
           EUNITS = 'KCAL'
           IF (IUNITS .EQ. ' ') THEN
               IUNITS = 'E units Kcal/mole'
           ELSE
               IUNITS(IND:) = ', E units Kcal/mole'
           ENDIF
       ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'JOUL') THEN
           EUNITS = 'JOUL'
           IF (IUNITS .EQ. ' ') THEN
               IUNITS = 'E units Joules/mole'
           ELSE
               IUNITS(IND:) = ', E units Joules/mole'
           ENDIF
       ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KJOUL') THEN
           EUNITS = 'KJOUL'
           IF (IUNITS .EQ. ' ') THEN
               IUNITS = 'E units Kjoule/mole'
           ELSE
               IUNITS(IND:) = ', E units Kjoule/mole'
           ENDIF
       ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KELV') THEN
           EUNITS = 'KELV'
           IF (IUNITS .EQ. ' ') THEN
               IUNITS = 'E units Kelvins'
           ELSE
               IUNITS(IND:) = ', E units Kelvins'
           ENDIF
       ENDIF
   ENDIF
   IF (AUNITS .EQ. ' ') THEN
       IF (UPCASE(LINE(N:), 4) .EQ. 'MOLE') THEN
           IF (UPCASE(LINE(N+4:), 1) .EQ. 'S') THEN
               AUNITS = 'MOLE'
               IF (IUNITS .EQ. ' ') THEN
                   IUNITS = 'A units mole-cm-sec-K'
               ELSE
                   IUNITS(IND:) = ', A units mole-cm-sec-K'
               ENDIF
           ELSEIF (UPCASE(LINE(N+4:), 4) .EQ. 'CULE') THEN
               AUNITS = 'MOLC'
               IF (IUNITS .EQ. ' ') THEN
                   IUNITS = 'A units molecules'
               ELSE
                   IUNITS(IND:) = ', A units molecules'
               ENDIF
           ENDIF
       ENDIF
   ENDIF
85 CONTINUE

C
IF (AUNITS .EQ. ' ') THEN
  AUNITS = 'MOLE'
  IND = ILASCH(IUNITS) + 1
  IF (IND .GT. 1) THEN
    IUNITS(IND:) = ', A units mole-cm-sec-K'
  ELSE
    IUNITS(IND:) = ' A units mole-cm-sec-K'
  ENDIF
ENDIF

IF (EUNITS .EQ. ' ') THEN
  EUNITS = 'CAL/'
  IND = ILASCH(IUNITS) + 1
  IF (IND .GT. 1) THEN
    IUNITS(IND:) = ', E units cal/mole'
  ELSE
    IUNITS(IND:) = ' E units cal/mole'
  ENDIF
ENDIF

RETURN
END

C---------------------------------------------------------------------------------------------------C

INTEGER FUNCTION IPPLEN (LINE)

BEGIN PROLOGUE

FUNCTION IPPLEN (LINE)

Returns the effective length of a character string, i.e.,
the index of the last character before an exclamation mark (!)
indicating a comment.

INPUT
  LINE - A character string.

OUTPUT
  IPPLEN - The effective length of the character string.

END PROLOGUE

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

CHARACTER LINE(*)

IN = IFIRCH(LINE)
IF (IN.EQ.0 .OR. LINE(IN:IN).EQ.'!') THEN
  IPPLEN = 0
ELSE
  IN = INDEX(LINE,'!')
  IF (IN .EQ. 0) THEN
    IPPLEN = ILASCH(LINE)
  ELSE
    IPPLEN = ILASCH(LINE(:IN-1))
  ENDIF
ENDIF

RETURN
END

C

CHARACTER*(*) FUNCTION UPCASE(ISTR, ILEN)
CHARACTER ISTR(*), LCASE(26)*1, UCASE(26)*1
DATA LCASE /'a'/'b'/'c'/'d'/'e'/'f'/'g'/'h'/'i'/'j'/'k'/'l'/'m'/',
  'n'/'o'/'p'/'q'/'r'/'s'/'t'/'u'/'v'/'w'/'x'/'y'/'z'/,
  'A'/'B'/'C'/'D'/'E'/'F'/'G'/'H'/'I'/'J'/'K'/'L'/'M'/',
  'N'/'O'/'P'/'Q'/'R'/'S'/'T'/'U'/'V'/'W'/'X'/'Y'/'Z'/
UPCASE = ' '
UPCASE = ISTR(:ILEN)
JJ = MIN (LEN(UPCASE), LEN(ISTR), ILEN)
DO 10 J = 1, JJ
  DO 10 N = 1, 26
    IF (ISTR(J:J) .EQ. LCASE(N))  UPCASE(J:J) = UCASE(N)
  10 CONTINUE
RETURN
END
SUBROUTINE CKI2CH (NUM, STR, I, KERR)
C
C SUBROUTINE CKI2CH (NUM, STR, I, KERR)
C Returns a character string representation of an integer
C and the effective length of the string.
C
C INPUT
C NUM - A number to be converted to a character string:
     the maximum magnitude of NUM is machine-dependent.
     Data type - integer scalar.
C OUTPUT
C STR - A left-justified character string representing NUM
     Data type - CHARACTER(*)
C I - The effective length of the character string
     Data type - integer scalar
C KERR - Error flag; character length errors will result in
     KERR=.TRUE.
     Data type - logical
C
C END PROLOGUE
C
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
CHARACTER STR(*), IST(10)*(1)
LOGICAL KERR
DATA IST/'0','1','2','3','4','5','6','7','8','9'/
BIGI = 2147483647.
C
I = 0
STR = ' '
ILEN = LEN(STR)
KERR = .FALSE.
IF (ILEN.LT.1 .OR. ABS(NUM).GT.BIGI) THEN
  KERR = .TRUE.
  RETURN
ENDIF
C
IF (NUM .EQ. 0) THEN
  STR = '0'
  I = 1
  RETURN
ELSEIF (NUM .LT. 0) THEN
  STR(1:1) = '-'
  I = 1
ENDIF
INUM = ABS(NUM)
NCOL = NINT(LOG10(REAL(INUM))) + 1

DO 10 J = NCOL, 1, -1
IDIV = INUM / 10.0**(J-1)
IF (J.EQ.NCOL .AND. IDIV.EQ.0) GO TO 10
LT = ILASCH(STR)
IF (LT .EQ. ILEN) THEN
   STR = ' '
   KERR = .TRUE.
   RETURN
ENDIF
STR(LT+1:) = IST(IDIV+1)
INUM = INUM - IDIV*10.0**(J-1)
10 CONTINUE
I = ILASCH(STR)

RETURN
END

SUBROUTINE THERM_IO(ERROR)

IMPLICIT NONE
INCLUDE 'THM_PAR2.FI'

CHARACTER FILE_NAME*20
INTEGER IDOT, ISTART, IFIRCH
LOGICAL ERROR

ERROR = .FALSE.
5 WRITE(*,10)
10 FORMAT(' Enter the file name for conversion : ')
READ(*,'(A20)')FILE_INP
OPEN(INP,FILE=FILE_INP, STATUS='OLD', ERR=30)
GO TO 50
30 CONTINUE
WRITE(*,40)FILE_INP
40 FORMAT(' Can not find file : ',A20)
GO TO 5
50 CONTINUE
ISTART = IFIRCH(FILE_INP)
IDOT = INDEX(FILE_INP, '.')
FILE_NAME = FILE_INP(ISTART:IDOT-1)
FILE_LOG = FILE_NAME + '.' + 'LOG'
FILE_LOG = 'THERMCVT.LOG'
FILE_LOG(IDOT:IDOT) = '. '
FILE_LOG(IDOT+1:IDOT+3) = 'LOG'
FILE_LST = FILE_NAME + '.' + 'LST'
FILE_LST = FILE_NAME + '.' + 'LST'
FILE_LST(IDOT:IDOT) = '. '
FILE_LST(IDOT+1:IDOT+3) = 'THM'
FILE_DAT = FILE_NAME + '.' + 'DAT'
OPEN(LOG,FILE=FILE_LOG, STATUS='UNKNOWN', ERR=230)
GO TO 250
230 CONTINUE
WRITE(*,240)FILE_LOG
240 FORMAT(' File I/O Error : ',A20)
ERROR = .TRUE.
GO TO 350
OPEN(LST, FILE=FILE_LST, STATUS='UNKNOWN', ERR=330)
GO TO 350
CONTINUE
WRITE(*,340)FILE_LST
FORMAT(' File I/O Error : ',A20)
ERROR = .TRUE.
GO TO 350
CONTINUE
RETURN
END
D.4 INTPE

PROGRAM CKINTP

--------------------------------------------------C

VERSION 4.0
W. ING

CHEBYSHEV POLYNOMIALS

VERSION 3.0

CHANGES FROM VERSION 1.0
1. Changed from REAL*8 to DOUBLE PRECISION

CHANGES FROM VERSION 1.1
1. Changed CHARACTER*100 to CHARACTER*80
2. Added THERMO "ALL" option
3. Write LENICK, LENRCK, LENCK to linking file
4. Allow reaction species to end in '=' or '_'
5. Allow real values of elemental composition in THERMO cards
6. Allow upper/lower case input

CHANGES FROM VERSION 1.2
1. Reaction delimiters are now "=" or "<=>" if reversible,
   "=>" if irreversible.
2. Fixed an error with IFIRCH(LINE) in IPPLEN

CHANGES FROM VERSION 1.3
1. Add "unix" change blocks

CHANGES FROM VERSION 1.4
1. Modify OPEN statements

CHANGES FROM VERSION 1.5
1. Correct molecules to moles unit conversion
2. Correct UPCASE to avoid dimensioning errors

CHANGES FROM VERSION 1.7
1. Further correction of molecules conversion for fall-off
   and third-body reactions

CHANGES FOR VERSION 1.8
1. Change Subroutine CKUNIT to parse LINE instead of SUB(*)
   in order to correct misinterpretation of unit strings
   with slashes.

CHANGES FOR VERSION 1.9
1. First record of linking file now consists of a character
   string version, precision, and logical error flag

CHANGES FOR VERSION 2.0
1. Error in UPCASE could cause interpreter to ignore some
   keywords.

CHANGES FOR VERSION 2.1
1. 10/18/90 (F. Rupley):
   Error in scaling the pre-exponential constants RPAR(3,*)
   where REV is declared, and FPAL(3,*) for fall-off reactions,
   as RPAR(3,II)*EFAC should read RPAR(3,NREV), and
   FPAL(3,II)*EFAC should read FPAL(3,NFAL).
   This error was introduced in CKINTERP.15 during refinement
   Dof units conversion routines.
2. Subroutine CKDUP modified to recognize that two reactions
   may be duplicate except for a third-body species in a
   fall-off reaction.

CHANGES FOR VERSION 2.2
1. 11/14/90 (F. Rupley per M. Coltrin):
   Initialize variable NCHRG

CHANGES FOR VERSION 2.3
1. In CKPREAC, error correction of 10/18/90 (above, V2.1).

CHANGES FOR VERSION 2.4
1. Additional checking of TLO,TMID,THI for species -
   a) set initial values at -1.
   b) if user has not provided a TLO,TMID, or THI, use the
values provided by THERMO.D AT.

C c) check that TLO < THI, TLO <= TMID <= THI

CHANGES FOR VERSION 2.5
1. Need to get TLO,THI,TMID from database BEFORE reading
   user's THERMO data (unless THERMO ALL option is used)

CHANGES FOR VERSION 2.6
1. LRENCK lengthened by II+NREV to reflect additional
   work space needed by CKRAT for a 4th parameter
   (perturbation factor).

CHANGES FOR VERSION 2.7
1. Two otherwise duplicate reactions are unique if one
   is a third body reaction and the other not.

CHANGES FOR VERSION 2.8
1. Change output format to print all 16 characters for
   a species name.

CHANGES FOR VERSION 2.9 (2/24/92 F. Rupley)
1. Check that reverse (REV) parameters were given when
   RTL reverse Teller-Landauer parameters are given.
2. Add 2*II to length of real work space

CHANGES FOR VERSION 3.0 (4/13/92 F. Rupley per M. Coltrin)
1. Correct logic in CKGROUP, add argument to call list.

CKINTP interprets a formatted ASCII representation of a
chemical reaction mechanism and creates the binary file LINK
required by CHEMKIN. CKINTP is dimensioned as follows:

******************** changed W. Ingl 06-15-94, 08-01-95 **********************

MDIM = maximum number of elements in a problem; (20)
KDIM = maximum number of species in a problem; (500)
MAXTP = maximum number of temperatures used to fit
thermodynamic properties of species
NPC = number of polynomial coefficients to fits (5)
NPCP2 = number of fit coefficients for a temperature range (7)
IDIM = maximum number of reactions in a mechanism; (1500)
NPAR = number of Arrhenius parameters in a reaction; (3)
NLAR = number of Landau-Teller parameters in a reaction; (2)
NFAR = number of fall-off parameters in a reaction; (70)
MAXSP = maximum number of species in a reaction (6)
MAXTB = maximum number of third bodies for a reaction (10)
LSYM = character string length of element and species names (16)

User input is read from LIN (Unit15), a thermodynamic database
is read from LTHRM (Unit17), printed output is assigned to LOUT
(Unit16), and binary linking data is written to LINC (Unit25).

REQUIRED ELEMENT INPUT: (Subroutine CKCHAR) (DIMENSION)

The word 'ELEMENTS' followed by a list of element
names, terminated by the word 'END';

The resulting element data stored in LINK is:
MM - integer number of elements found
ENAME(*) - CHARACTER(*) array of element names (MDIM)
AWT(*) - real array of atomic weights; (MDIM)
   default atomic weights are those on
   atomic weight charts; if an element
   is not on the periodic chart, or if
   it is desirable to alter its atomic
   weight, this value must be included
   after the element name, enclosed by
   slashed, i.e., D/2.014/

REQUIRED SPECIES INPUT: (Subroutine CKCHAR)
The word 'SPECIES' followed by a list of species names, terminated by the word 'END';

The resulting species data stored in LINK is:

- **KK** integer number of species found
- **KNAME(*)** - CHARACTER(*) array of species names (KDIM)

Optional thermodynamic data: (Subroutine CKTHRM)
(If this feature is not used, thermodynamic properties are obtained from a CHEMKIN database.) The format for this option is the word 'THERMO' followed by any number of 4-line data sets:

1. **Line 1**: species name, optional comments, elemental composition, phase, \( T(\text{low}) \), \( T(\text{high}) \), \( T(\text{mid}) \), additional elemental composition, card number (col. 80);
   - **format** (A10,A14,4(A2,I3),A1,E10.0,E10.0,E8.0,(A2,I3),I1)

2. **Line 2**: coefficients \( a(1-5) \) for upper temperature range, card number (col. 80);
   - **format** (5(e15.0),I1)

3. **Line 3**: coefficients \( a(6-7) \) for upper temperature range, coefficients \( a(1-3) \) for lower temperature range, card number (col. 80);
   - **format** (5(e15.0),I1)

4. **Line 4**: coefficients \( a(4-7) \) for lower temperature range, card number (col. 80);
   - **format** (4(e15.0),I1)

End of THERMO data is indicated by 'END' line or new keyword.

The resulting thermodynamic data stored in LINK are:

- **WTM(*)** - real array of molecular weights (KDIM)
- **KNCF(*)** - integer composition of species (MDIM,KDIM)
- **KPHSE(*)** - integer phase of a species; (KDIM)
- **KCHRG(*)** - ionic charge of a species; (KDIM)
  - = 0 except in presence/absence of electrons
  - = +n in absence of n electrons
  - = -n in presence of n electrons
- **NCHRG** - integer number of species with \( K \neq 0 \)
- **NT(*)** - array of number of temperatures used (KDIM) in fits
- **T(*,*)** - array of temperatures used in fits (MAXTP,KDIM)
- **A(N,L,K)** - Thermodynamic properties for species \( K \) consists of polynomial coefficients for fits to
  - \( CP/R = \sum A(N,L,K)*Temperature**(N-1) \), \( N=1,NPC+2 \)
  - where \( T(L,K) \leq Temperature < T(L+1,K) \), and,
  - \( N=NPC+1 \) is formation enthalpy \( HO/R = A(NPC+1,L,K) \), \( N=NPC+2 \) is formation entropy \( SO/R = A(NPC+2,L,K) \)

Optional reaction input:
Reaction data is input after all ELEMENT, SPECIES and THERMO data in the following format:

1) (Subroutine CKREAC)

- The first line contains the keyword 'REATIONS' and an optional description of units:
  - 'MOLES' - (default), pre-exponential units are moles-sec-K;
  - 'MOLECULES' - pre-exponential units are molecules and will be converted to moles.
  - 'KELVINS' - activation energies are Kelvins, else the activation energies are converted to Kelvins;
  - 'CAL/MOLE' - (default), activation energies are cal/mole;
  - 'KCAL/MOLE' - activation energies are Kcal/mole;
A description of each reaction is expected to follow. Required format for a reaction is a list of '+'-delimited reactants, followed by a list of '+'-delimited reactants, each preceded by its stoichiometric coefficient if greater than 1; separating the reactants from the products is a '=' if reversible reaction, else a '=>'. Following the reaction string on the same line are the space-delimited Arrhenius coefficients.

If the reaction contains a third body, this is indicated by the presence of an 'M' as a reactant or product or both, and enhancement factors for third-bodies may be defined on additional lines as described in (2).

If the reaction contains a radiation wavelength, this is indicated by the presence of an 'HV' either as a reactant or as a product. Unless otherwise defined on additional lines as described in (2), the value of the wavelength is -1.0 if a reactant or +1.0 if a product.

If the reaction is a fall-off reaction, this is indicated either by a '(+M)' or a '+(KNAME(K))', and there must be additional lines as described in (2) to define fall-off parameters.

2) (Subroutine CKAUXL)

Additional information for a reaction is given on lines immediately following the reaction description; this data will consist of a 'keyword' to denote the type of data, followed by a '/ ', then the required parameters for the keyword, followed by another '/'. There may be more than one keyword per line, and there may be any number of lines. The keywords and required parameters are as follows:

KNAME(K)/efficiency value/ - species (K) is an enhanced third body in the reaction
HV/wavelength/ - radiation wavelength parameter
LT/val1 val2/ - Landau-Teller coefficients
LOW/val1 val2 val3/ - low fall-off parameters
TROE/val1 val2 val3 val4/ - Troe fall-off parameters;
if val4 is omitted, a default parameter will be used

ADDED W.ING, 06/15/95***************************************************************************

EXTROE/val1 val2 val3 val4 val5/ - Extended Troe fall-off parameters;
val1-val4 = Troe parameters
val5 = well # (for cal. ext reduced pressure)

***************************************************************************

SRI/val1 val2 val3 val4/ - SRI fall-off parameters;
if val4 is omitted, a default parameter will be used

(it is an error to have both LT and Fall-off defined)
REV/par1 par2 par3/ - reverse parameters given
RLT/val1 val2/ - Landau-Teller coefficients for reverse
(it is an error if REV given and not RLT)

The end of all reaction data is indicated by an 'END' card or
Resulting reaction data stored in LINC are:

- **II**: integer number of reactions found
- **PAR(*,*)**: array of real Arrhenius coefficients (NPAR, IDIM)
- **NSPEC(*)**: total number of species in a reaction (IDIM)
  - if NSPEC < 0, reaction is irreversible
- **NREAC(*)**: number of reactants only (IDIM)
- **NUNK(*,*)**: array of species numbers for reaction (MAXSP, IDIM)
  - array of stoichiometric coefficients (MAXSP, IDIM)
  - of species in a reaction, negative=reactant, positive=product
- **NR**: number of reactions with radiation wavelength
- **IWL(*,*)**: integer reaction numbers (IDIM)
- **WL(*)**: real radiation wavelengths (IDIM)
- **NTTHB(*)**: number of reactions with third bodies (IDIM)
- **ITHB**: integer reaction numbers (IDIM)
- **NTBS(*)**: total number of enhanced species for NTTHB (IDIM)
- **NKTB(*,*)**: species numbers of enhanced species (MAXTB, IDIM)
  - enhancement factors (MAXTB, IDIM)
- **NFAL**: number of fall-off reactions
- **IFAL(*)**: integer reaction numbers (IDIM)
- **KFAL(*)**: integer species number for which concentrations are a factor in fall-off calculation
- **IFOP(*)**: integer fall-off type number
  - = 0 if fall-off reaction is found
  - = 1 for Lindemann form
  - = 2 for 6-parameter Troe form
  - = 3 for 7-parameter Troe form
  - = 4 for SRI form
  - = 5 for 8-parameter Extended Troe form
- **PFAL(*,*)**: fall-off parameters (NFAR, IDIM)
- **NLAN**: number of reactions with Landau-Teller
- **ILAN**: integer reaction numbers (IDIM)
- **PLAN**: Landau-Teller parameters (NLAR, IDIM)
- **NREV**: number of reactions with reverse parameters
- **IREV(*)**: integer reaction numbers (IDIM)
- **RPAR(*,*)**: parameters (NPAR, IDIM)
- **NRLT**: number of reactions with reverse parameters and Landau-Teller parameters
- **IRLT(*)**: integer reaction numbers (IDIM)
- **RLAN(*,*)**: reverse Teller-Laudauer parameters (NLAR, IDIM)

-----------------------------------------------

```
PARAMETER (MDIM=20, KDIM=500, MKDIM=MDIM*KDIM, IDIM=1500, LSYM=16,
  1 NPAR=3, NPIDIM=IDIM*NPAR, NPC=5, NPCP2=NPC+2, MAXTP=3,
  2 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
  3 NLAR=2, NSIDIM=MAXSP*IDIM, NTIDIM=MAXTB*IDIM,
  4 NLDIM=NLAR*IDIM, NFAR=70, NFDIM=NFA*IDIM,
  5 NTDIM=KDIM*MAXTP, NIDIM=9*IDIM, LIN=5, LOUT=6,
  6 LTHRM=17, LINC=25)
```
CHARACTER KNAME(KDIM)*(LSYM), ENAME(MDIM)*(LSYM), SUB(80)*80,
KEY(5)*4, LINE*80, IUNITS*80, AUNITS*4, EUNITS*4,
UPCASE*4, VERS*(LSYM), PREC*(LSYM)

DIMENSION AWT(MDIM), KNCF(MDIM,KDIM), WTM(KDIM), KPHSE(KDIM),
KCHRG(KDIM), A(NPCP2,NTR,KDIM), T(MAXTP,KDIM), NT(KDIM),
NSPEC(IDIM), NREAC(IDIM), NU(MAXSP,IDIM),
NUK(MAXSP,IDIM), PAR(NPAR,IDIM), IDUP(IDIM), IREV(IDIM),
RPAR(NPAR,IDIM), ILAN(NLAR,IDIM), IWL(IDIM), WL(IDIM),
IFAL(IDIM), IFOP(IDIM), KFAL(IDIM), PFAL(NFAR,IDIM),
ITHB(IDIM), NTBS(IDIM), AIK(MAXTB,IDIM), NKTB(MAXTB,IDIM)
DIMENSION VALUE(5)

LOGICAL KERR, THERMO, ITHRM(KDIM)

Initialize variables

DATA KEY/'ELEM', 'SPEC', 'THER', 'REAC', 'END'/, KERR/.FALSE./,
ITASK, NCHRG, MM, KK, 11, NLAN, NFAL, NTHB, NREV, NRTL, NWL/11*0/,
ENAME, AWT/MDIM*0.0/, THERMO/.TRUE./,
T/NTDIM*-1.0/, KNAME, WTM, NT, KPHSE, KCHRG, ITHRM,
/KDIM*0.0, KDIM*3, KDIM*0, KDIM*0, KDIM*.FALSE./,
/WL, IFOP, NTBS, IDUP/IDIM*0.0, IDIM*0, IDIM*1, IDIM*0, IDIM*0/,
NSPEC, NREAC, IREV, ILAN, IRTL, IWL, IFAL, KFAL, ITHB/NDIM*0/

DATA NUNK, NU/NSIDIM*0, NSIDIM*0/, NKTB, AIK/NTIDIM*0, NTIDIM*-1.0/,
DATA PAR, RPAR/NPIDIM*0.0, NPIDIM*0.0/,
DATA PLAN, RLAN/NLIDIM*0.0, NLIDIM*0.0/,
DATA PFAL/NFIDIM*0.0/, KNCF/MKDIM*0.0/, A/NKTDIM*0.0/

ADD FILEIO SUBROUTINE 02/94 ING

CALL FILEIO_INT

VERS = '3.0'
WRITE (LOUT, 15) VERS:(3)

C*****precision > double
2/' DOUBLE PRECISION'/)
PREC = 'DOUBLE'
C*****END precision > double
C*****precision > single
C 2'/ SINGLE PRECISION'/)
PREC = 'SINGLE'
C*****END precision > single

START OF MECHANISM INTERPRETATION

100 CONTINUE
LINE = '.',
READ (LIN, '(A)', END=5000) LINE
105 CONTINUE
ILEN = IPPLEN(LINE)
IF (ILEN .EQ. 0) GO TO 100

CALL CKISUB (LINE(:ILEN), SUB, NSUB)

IS THERE A KEYWORD?
CALL CKCOMP (UPCASE(SUB(1), 4), KEY, 5, NKEY)
IF (NKEY .GT. 0) ITASK = 0

IF (NKEY.EQ.1 .OR. NKEY.EQ.2) THEN
ELEMENT OR SPECIES DATA
ITASK = NKEY
IF (NSUB .EQ. 1) GO TO 100
DO 25 N = 2, NSUB
SUB(N-1) = ' '
SUB(N-1) = SUB(N)
25 CONTINUE
NSUB = NSUB-1
ELSEIF (NKEY .EQ. 3) THEN
THERMODYNAMIC DATA
IF (NSUB .GT. 1) THEN
   IF (UPCASE(SUB(2), 3) .EQ. 'ALL') THEN
      THERMO = .FALSE.
      READ (LIN, '(A)') LINE
      CALL IPPARR (LINE, -1, 3, VALUE, NVAL, IER, LOUT)
      IF (NVAL .NE. 3 .OR. IER.NE.0) THEN
         KERR = .TRUE.
         WRITE (LOUT, 333)
      ELSE
         TLO = VALUE(1)
         TMID = VALUE(2)
         THI = VALUE(3)
      ENDIF
   ENDIF
ELSE
   USE THERMODYNAMIC DATABASE FOR DEFAULT TLO, TMID, THI
ENDIF
ELSE
USE THERMODYNAMIC DATABASE FOR DEFAULT TLO, TMID, THI
C*****OPEN statement > vms
OPEN (LTHRM, STATUS='OLD', FORM='FORMATTED', READONLY, 1 SHARED)
C*****END OPEN statement > vms
C*****OPEN statement > unix
OPEN (LTHRM, FORM='FORMATTED', FILE='thermdat')
C*****END OPEN statement > unix
READ (LTHRM, '(A)') LINE
READ (LTHRM, '(A)') LINE
CALL IPPARR (LINE, -1, 3, VALUE, NVAL, IER, LOUT)
IF (NVAL .NE. 3 .OR. IER.NE.0) THEN
   KERR = .TRUE.
   WRITE (LOUT, 333)
ELSE
   TLO = VALUE(1)
   TMID = VALUE(2)
   THI = VALUE(3)
ENDIF
REWIND (LTHRM)
CLOSE (LTHRM)
ENDIF
CALL CKTHRM (LIN, MDIM, ENAME, MM, AWT, KNAME, KK, KNCF, 1 KPHSE, KCHRG, WTM, MAXTF, NT, NTR, TLO, TMID, 2 THI, T, NFCP2, A, LTHRM, KERR, LOUT, LINE)
IF (.NOT. THERMO)
CALL CKPRNT (MDIM, MAXTP, MM, ENAME, KK, KNAME, WTM, KPHSE, KCHRG, NT, T, TLO, TMID, THI, KNCF, ITHRM, LOUT, KERR)

II = IFIRCH(LINE)
IF (UPCASE(LINE(II:), 4) .EQ. 'REA C') GO TO 105
ELSEIF (NKEY .EQ. 4) THEN
   ITASK = 4
   START OF REACTIONS; ARE UNITS SPECIFIED?
   CALL CKUNIT (LINE(:ILEN), AUNITS, EUNITS, IUNITS)
   IF (THERMO) THEN
      THERMODYNAMIC DATA
      ******OPEN statement > vms
      OPEN (LTHRM, STATUS='OLD', FORM='FORMATTED', READONLY, SHARED)
      ******END OPEN statement > vms
      ******OPEN statement > unix
      OPEN (LTHRM, FORM='FORMATTED', FILE='thermdat')
      ******END OPEN statement > unix
      READ (LTHRM, '(A)') LINE
      READ (LTHRM, '(A)') LINE
      CALL IPPARR (LINE, -1, 3, VALUE, NVAL, IER, LOUT)
      IF (NVAL .NE. 3 .OR. IER.NE.0) THEN
         KERR = .TRUE.
         WRITE (LOUT, 333)
      ELSE
         TLO = VALUE(1)
         TMID = VALUE(2)
         THI = VALUE(3)
      ENDIF
      CALL CKTHRM (LTHRM, MDIM, ENAME, MM, AWT, KNAME, KK, KNCF, KPHSE, KCHRG, WTM, MAXTP, NT, NTR, TLO, TMID, THI, T, NPCP2, A, ITHRM, KERR, LOUT, LINE)
      CALL CKPRNT (MDIM, MAXTP, MM, ENAME, KK, KNAME, WTM, KPHSE, KCHRG, NT, T, TLO, TMID, THI, KNCF, ITHRM, LOUT, KERR)
      THERMO = .FALSE.
   ENDIF
   WRITE (LOUT, 1800)
   GO TO 100
ENDIF

IF (ITASK .EQ. 1) THEN
   ELEMENT DATA
   IF (MM .EQ. 0) THEN
      WRITE (LOUT, 200)
      WRITE (LOUT, 300)
      WRITE (LOUT, 200)
   ENDIF
   IF (NSUB .GT. 0) THEN
      M1 = MM + 1
      CALL CKCHAR (SUB, NSUB, MDIM, ENAME, AWT, MM, KERR, LOUT)
      DO 110 M = M1, MM
         IF (AWT(M) .LE. 0) CALL CKAWTM (ENAME(M), AWT(M))
         WRITE (LOUT, 400) M, ENAME(M)(:4), AWT(M)
      IF (AWT(M) .LE. 0) THEN
         KERR = .TRUE.
      ENDIF
      WRITE (LOUT, 1000) ENAME(M)
   ENDIF
ENDIF
ELSEIF (ITASK .EQ. 2) THEN
   PROCESS SPECIES DATA
   IF (KK .EQ. 0) WRITE (LOUT, 200)
   IF (NSUB .GT. 0)
      CALL CKCHAR (SUB, NSUB, KDIM, KNAME, WTM, KK, KERR, LOUT)
   ELSEIF (ITASK .EQ. 4) THEN
   PROCESS REACTION DATA
   IND = 0
   DO 120 N = 1, NSUB
      IND = MAX(IND, INDEX(SUB(N),'/'))
      IF (UPCASE(SUB(N), 3) .EQ. 'DUP') IND = MAX(IND,1)
   CONTINUE
   IF (IND .GT. 0) THEN
      AUXILIARY REACTION DATA
      CALL CKAUXL (SUB, NSUB, II, KK, KNAME, LOUT, MAXSP, NPAR,
      NSPEC, NTHB, NTHB, MAXTB, NKTB, AIK, NPAR, PAR, RPAR,
      NFAL, IFAL, IDUP, NFAR, IFOP, KFAL, NLAN,
      ILAN, NLAN, PLAN, NREV, IREV, RPAR,
      NRLT, IRLT, RLAR, NLAR, PLAN, NREV, IREV, RPAR,
      NWL, IWL, WL, KERR)
   ELSE
      THIS IS A REACTION STRING
      IF (II .LT. IDIM) THEN
         CHECK PREVIOUS REACTION FOR COMPLETENESS
         CALL CPREAC (II, MAXSP, NSPEC, NPAR, PAR, RPAR,
         AUNITS, EUNITS, NSPEC, NPAR, PAR, RPAR,
         NMH, MM, NNCF, IDUP, NFAL, IFOP, KFAL,
         NFAR, IFAL, IFOP, NREV, IREV, NTB, INT,
         NLAN, ILAN, RLAN, NLAN, PLAN, NREV, IREV, RPAR,
         NRLT, IRLT, WRN, WRK, KERR, LOUT)
      ELSE
         WRITE (LOUT, 1070)
         KERR = .TRUE.
      ENDIF
   ENDIF
   ELSEIF (ITASK .EQ. 4) THEN
   PROCESS REACTION DATA
   ENDIF
C 5000 CONTINUE
END OF INPUT

IF (II .GT. 0) THEN

CHECK FINAL REACTION FOR COMPLETENESS

CALL CPREAC (II, MAXSP, NSPEC, NPAR, PAR, RPAR, AUNITS,
1 EUNITS, NREAC, NUNK, NU, KCHRG, MDIM, MM,
2 KNCF, IDUP, NFAL, IFAL, NFAR, FPAR, IFOP,
3 NREV, IREV, NTHB, ITHB, NLAN, ILAN, NALT,
4 IRTL, KERR, LOUT)

CHECK REACTIONS DECLARED AS DUPLICATES

DO 500 I = 1, II
  IF (IDUP(I) .LT. 0) THEN
    KERR = .TRUE.
    WRITE (LOUT, 1095) I
  ENDIF
500 CONTINUE

WRITE (LOUT, '(/IX,A)') '  NOTE: //IUNITS(//ILASCH(IUNITS))

ELSEIF (THERMO) THEN

THERE WAS NO REACTION DATA, MAKE SURE SPECIES DATA IS COMPLETE

C*****OPEN statement > vms
OPEN (LTHRM, STATUS='OLD', FORM='FORMATTED', READONLY,
1 SHARED)
C*****END OPEN statement > vms
C*****OPEN statement > unix
OPEN (LTHRM, FORM='FORMATTED', FILE='thermdat')
C*****END OPEN statement > unix
C
READ (LTHRM, '(A)') LINE
READ (LTHRM, '(A)1) LINE
CALL IPPARR (LINE, -1, 3, VALUE, NVAL, IER, LOUT)
  IF (NVAL .NE. 3 .OR. IER.NE.0) THEN
    KERR = .TRUE.
    WRITE (LOUT, 333)
  ELSE
    TLO = VALUE(1)
    TMID = VALUE(2)
    THI = VALUE(3)
  ENDFID
CALL CKTHRM (LTHRM, MDIM, ENAME, MM, AWT, KNAME, KK, KNCF,
1 KPHSE, KCHRG, WTM, MAXTP, NT, NTR, TLO, TMID,
2 THI, T, NPCP2, A, ITHRM, KERR, LOUT, LINE)
CALL CKPRNT (MDIM, MAXTP, MM, ENAME, KK, KNAME, WTM, KPHSE,
1 KCHRG, NT, T, TLO, TMID, THI, KNCF, ITHRM,
2 LOUT, KERR)
ENDIF

DO 1150 K = 1, KK
  IF (KCHRG(K) .NE. 0) NCHRG = NCHRG+1
1150 CONTINUE

LENICK = 1 + (3 + MM)*KK + (2 + 2*MAXSP)*II + NLAN + NRTL
  + 3*NFAL + (2 + MAXTB)*NTHB + NREV + NWL
LENCKK = MM + KK
LENCKKK = 3 + MM + KK*(5 + MAXTP + NTR*NPCP2) + II*7 + NREV
  + NPAR*(II + NREV) + NLAR*(NLAN + NRTL)
  + NFAR*NFAL + MAXTB*NTHB + NWL

OPEN LINKING FILE
OPEN (LINC, STATUS='NEW', FORM='UNFORMATTED')

OPEN (LINC, FORM='UNFORMATTED', FILE='cklink')

WRITE (LINC) VERS, PREC, KERR
WRITE (LINC) LENICK, LENRCK, LENCKK, MM, KK, II, MAXSP,
1      MAXTB, MAXTP, NPC, NFAR, NLAR, NREV, NFAL,
2      NTHB, NLAN, NRLT, NWL, NCHRG
WRITE (LINC) (ENAME(M), AWT(M), M = 1, MM)
WRITE (LINC) (KNNAME(K), (KNCF(M, K), M=1, MM), KPHSE(K),
1      KCHRG(K), WT(M), NT(K), (T(L, K), L=1, MAXTP),
2      ((A(M, L, K), M=1,NPCP2), L=1,NTR), K = 1, KK)

IF (II .GT. 0) THEN
WRITE (LINC) (NSPEC(I), NREAC(I), (PAR(N, I), N = 1, NPAR),
1      (NU(M, I), NUNK(M, I), M = 1, MAXSP), I = 1, II)

IF (NREV .GT. 0) WRITE (LINC)
1      (IREV(N), (RPAR(L, N), L=1, NPAR), N=1, NREV)

IF (NFAL .GT. 0) WRITE (LINC)
1      (IFAL(N), IFOP(N), KFAL(N), (PFAL(L, N), L=1, NFAR), N = 1, NFAL)

IF (NTHB .GT. 0) WRITE (LINC)
1      (ITHB(N), NTBS(N), (NKTB(M, N), AIK(M, N), M=1, MAXTB), N=1, NTHB)

IF (NLAR .GT. 0) WRITE (LINC)
1      (ILAN(N), (PLAN(L, N), L = 1, NLAR), N = 1, NLAR)

IF (NRLT .GT. 0) WRITE (LINC)
1      (IRLT(N), (RLAN(L, N), L = 1, NLAR), N = 1, NRLT)

IF (NWL .GT. 0) WRITE (LINC) (IWL(N), WL(N), N = 1, NWL)
ENDIF

IF (KERR) THEN
  WRITE (LOUT, '(//A)')
1  ' WARNING...THERE IS AN ERROR IN THE LINKING FILE'
ELSE
  IF (II .LE. 0) WRITE (LOUT, '(/A)')
1    ' WARNING...NO REACTION INPUT FOUND; ',
2    ' LINKING FILE HAS NO REACTION INFORMATION ON IT.'

WRITE (LOUT, '(()/A)')
1  ' NO ERRORS FOUND ON INPUT...CHEMKIN LINKING FILE WRITTEN.'

WRITE (LOUT, '(()/A,3(()/A,16))')
1  ' WORKING SPACE REQUIREMENTS ARE',
2  ' INTEGER: ', LENICK,
3  ' REAL: ', LENRCK,
4  ' CHARACTER: ', LENCKK
ENDIF

FORMATS

C 200 FORMAT (26X,20('-*'))
300 FORMAT (26X,'ELEMENTS',5X,'ATOMIC',/26X,'CONSIDERED',3X,'WEIGHT')
333 FORMAT (/6X,'Error...no TLO,TMID,THI given for THERMO ALL...'/)
400 FORMAT (25X,13,'. ',A4,G15.6)
C
1000 FORMAT (6X,'Error...no atomic weight for element ',A)
1070 FORMAT (6X,'Error...more than IDIM reactions...')
1095 FORMAT (6X,'Error...no duplicate declared for reaction no. ',I3)
1800 FORMAT (///54X, ' (k = A T**b exp(-E/RT))',/,
1       6X,'REACTIONS CONSIDERED',30X,'A',8 X , 'b ',8 X , 'E ',/)
C
CLOSE(LOUT)
CLOSE(LINC)
C
STOP
END
C-----------------------------------------------C
SUBROUTINE CKCHAR (SUB, NSUB, NDIM, STRAY, RAY, NN, KERR, LOUT)
C
Extracts names and real values from an array of CHAR*(*)
substrings; stores names in STRAY array, real values in RAY;
i.e. can be used to store element and atomic weight data,
species names, etc.
C
Input:  SUB(N),N=1,NSUB  - array of CHAR*(*) substrings
        NSUB     - number of substrings
        NDIM     - size of STRAY,RAY arrays
        NN      - actual number of STRAY found
        STRAY(N),N=1,NN - CHAR*(*) array
        RAY(N),N=1,NN  - Real array
        LOUT    - output unit for error messages
C
Output: NN       - incremented if more STRAY found
        STRAY(N),N=1,NN - incremented array of STRAY
        RAY(N),N=1,NN  - incremented array of reals
        KERR   - logical, .TRUE. = error in data
C
F. Rupley, Div. 8245, 2/5/88
C-----------------------------------------------C
C*****precision > double                      C
C***END precision > double
C*****precision > single                     C
C***END precision > single
C
DIMENSION RAY(*), PAR(1)
CHARACTER SUB(*), STRAY(*), ISTR*80, UPCASE*4
LOGICAL KERR
C
ILEN = LEN(STRAY(1))
C
DO 200 N = 1, NSUB
   IF ( UPCASE(SUB(N), 3) .EQ. 'END') RETURN
   ISTR = '
   IL = INDEX(SUB(N),'/')
   IF (IL .EQ .1) THEN
      KERR = .TRUE.
      WRITE (LOUT, 130) SUB(N)(:ILASCH(SUB(N)))
   ELSE
      IF (IL .LE. 0) THEN
         ISTR = SUB(N)
      ELSE
         ISTR = SUB(N)(:IL-1)
   ENDIF
200 CONTINUE
C
CALL CKCOMP (ISTR, STRAY, NN, INUM)

C IF (INUM .GT. 0) THEN
   WRITE (LOUT, 100) SUB(N)(:ILASCH(SUB(N)))
ELSE
   IF (NN .LT. NDIM) THEN
      IF (ISTR(ILLEN+1:) .NE. ' ') THEN
         WRITE (LOUT, 120) SUB(N)(:ILASCH(SUB(N)))
         KERR = .TRUE.
      ELSE
         NN = NN + 1
         STRAY(NN) = ' '
         STRAY(NN) = ISTR(ILLEN)
         IF (I1 .GT. 0) THEN
            I2 = I1 + INDEX(SUB(N)(I1+1:),'/')
            ISTR = SUB(N)(I1+1:I2-1)
            CALL IPPARR (ISTR, 1, 1, PAR, NVAL, IER, LOUT)
            IF (IER .EQ. 0) THEN
               RAY(NN) = PAR(1)
            ELSE
               KERR = .TRUE.
            ENDIF
         ELSE
            WRITE (LOUT, 110) SUB(N)(:ILASCH(SUB(N)))
            KERR = .TRUE.
         ENDIF
      ENDIF
   ELSE
      WRITE (LOUT, 110) SUB(N)(:ILASCH(SUB(N)))
      KERR = .TRUE.
   ENDIF
ENDIF
ENDIF
ELSE
   WRITE (LOUT, 110) SUB(N)(:ILASCH(SUB(N)))
   KERR = .TRUE.
ENDIF
ENDIF
200 CONTINUE
C
100 FORMAT (6X,'Warning...duplicate array element ignored...',A)
110 FORMAT (6X,'Error...character array size too small for ...',A)
120 FORMAT (6X,'Error...character array element name too long...',A)
130 FORMAT (6X,'Error...misplaced value...',A)
END
C--------------------------------------------------------------------------C
SUBROUTINE CKAWTM (ENAME, AWT)
C
C Returns atomic weight of element ENAME.
C Input:  ENAME - CHAR(**) element name
C Output: AWT - real atomic weight
C
F. Rupley, Div. 8245, 11/11/86
C--------------------------------------------------------------------------C
PARAMETER (NATOM = 102)
DIMENSION ATO M(NATOM)
CHARACTER ENAME(*), IATOM(NATOM)*2, UPCASE*2
C
DATA (IATOM(I),ATOM(I),I=1,102) /
   *'F ', 18.99840, 'NE', 20.18300, 'NA', 22.98980, 'MG', 24.31200,
   *'CL', 35.45300, 'AR', 39.94800, 'K ', 39.10200, 'CA', 40.08000,
   *'SC', 44.95600, 'TI', 47.90000, 'V ', 50.94200, 'CR', 51.99600,
   *'MN', 54.93800, 'FE', 55.84700, 'CO', 58.93320, 'NI', 58.71000,
DATA (IATOM(I),ATOM(I),I=1,80) /
** Cu', 63.54000, 'Zn', 65.37000, 'Ga', 69.72000, 'Ge', 72.59000,
** As', 74.92160, 'Se', 78.96000, 'Br', 79.90900, 'Kr', 83.80000,
** Rb', 85.47000, 'Sr', 87.62000, 'Y', 88.90500, 'Zr', 91.22000/

DATA (IATOM(I),ATOM(I),I=81,NATOM) /
** Nb', 92.90600, 'Mo', 95.94000, 'Tc', 99.00000, 'Ru', 101.07000,
** Rh', 102.90500, 'Pd', 106.40000, 'Ag', 107.87000, 'Cd', 112.40000,
** In', 114.82000, 'Sn', 118.69000, 'Sb', 121.75000, 'Te', 127.60000,
** I', 126.90440, 'Xe', 131.30000, 'Cs', 132.90500, 'Ba', 137.34000,
** La', 138.91000, 'Ce', 140.12000, 'Pr', 140.90700, 'Nd', 144.24000,
** Pm', 145.00000, 'Sm', 145.00000, 'Eu', 147.27000, 'Gd', 151.96000,
** Tb', 152.54000, 'Dy', 158.92400, 'Ho', 158.92400, 'Er', 162.50000,
** Tm', 168.93400, 'Yb', 173.04000, 'Lu', 174.99700, 'Hf', 178.49000,
** Er', 180.94800, 'W', 183.85000, 'Re', 186.20000, 'Os', 190.20000,
** Ir', 192.20000, 'Pt', 195.09000, 'Au', 196.96700, 'Hg', 200.59000/

CALL CKCOMP (UPCASE(ENAME, 2), IATOM, NATOM, L)
IF (L .GT. 0) AWT = ATOM(L)
RETURN
END

SUBROUTINE CKTHRM (LUNIT, MDIM, ENAME, MM, AWT, KNAME, KK, KNCF,
1 KPHSE, KCHRG, WTM, MAXTP, NT, NTR, TLO, TMID,
2 THI, T, NPCP2, A, ITHRM, KERR, LOUT, ISTR)

C Finds thermodynamic data and elemental composition for species.
C
C Input:  LUNIT - unit number for input of thermo properties
C          MDIM - maximum number of elements allowed
C          ENAME(M),M=1,MM - array of CHAR(*) element names
C          MM - total number of elements declared
C          AWT(M),M=1,MM - array of atomic weights for elements
C Output: KNCF(M,K) - elemental composition of species
C          KPHSE(K) , K=1, KK - integer array, species phase
C          KCHRG(K), K=1, KK - integer array of species charge
C          WTM(K),K=1, KK - array of molecular weights of species
C          A(M,L,K)- array of thermodynamic coefficients
C          T(N),N=1,NT - array of temperatures
C          KERR - logical error flag

C-------------------------------------------------------------------------C
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
DIMENSION WTM(*), NT(*), T(MAXTP,*), KPHSE(*), KNCF(MDIM,*),
1 KCHRG(*), A(NPCP2,NTR,*), AWT(*), VALUE(5)
CHARACTER ENAME(*)*(*) , KNAME(*)*(*) , LINE(4)*80, ELEM*16
CHARACTER UPCASE*(4), ISTR*80, SUB(80)*80
LOGICAL KERR, ITHRM(*)

C IF (MM.LE.0 .OR. KK.LE.0) WRITE (LOUT, 80)
C GO TO 20
10 CONTINUE
ISTR = ' ',
READ (LUNIT, '(A)', END=40) ISTR
20 CONTINUE
ILEN = IPPLEN(ISTR)
IF (ILEN .LE. 0) GO TO 10
C CALL CKISUB (ISTR(:ILEN), SUB, NSUB)
CALL CKCOMP (SUB(1), KNAME, KK, K)
IF (K .EQ. 0) THEN
IF (UPCASE(SUB(1), 3) .EQ. 'END' .OR.
1   UPCASE(SUB(1), 4) .EQ. 'REAC') RETURN
GO TO 10
ENDIF
C IF (ITHRM(K)) GO TO 10
ITHRM(K) = .TRUE.
LINE(1) = ' ',
LINE(1) = ISTR
DO 25 L = 2, 4
LINE(L) = ' ',
READ (LUNIT, '(A)', END=40) LINE(L)
25 CONTINUE
C ICOL = 20
DO 60 I = 1, 5
ICOL = ICOL + 5
IF (I .EQ. 5) ICOL = 74
ELEM = LINE(1)(ICOL:ICOL+1)
IELEM = 0
C IF (LINE(1)(ICOL+2:ICOL+4) .NE. ' ') THEN
CALL IPPARR
1   1   (LINE(1)(ICOL+2:ICOL+4), 0, 1, VALUE, NVAL, IER, LOUT)
   IELEM = VALUE(1)
ENDIF
C IF (ELEM.NE. ' ' .AND. IELEM.NE.0) THEN
1   IF (UPCASE(ELEM, 1) .EQ. 'E')
KCHRG(K) = KCHRG(K)+IELEM*(-1)
CALL CKCOMP (ELEM, ENAME, MM, M)
IF (M .GT. 0) THEN
   KNCF(M,K) = IELEM
   WTM(K) = WTM(K) + AWT(M)*FLOAT(IELEM)
ELSE
   WRITE (LOUT, 100) ELEM,KNAME(K)(:10)
   KERR = .TRUE.
ENDIF
60 CONTINUE
C IF (UPCASE(LINE(1)(45:),1) .EQ. 'L') KPHSE(K)=1
C IF (UPCASE(LINE(1)(45:),1) .EQ. 'S') KPHSE(K)=-1
C------Currently allows for three temperatures, two ranges;
C in future, NT(K) may vary, NTR = NT(K)-1
C T(1,K) = TLO
IF (LINE(1)(46:55) .NE. ' ') CALL IPPARR
1   1   (LINE(1)(46:55), 0, 1, T(1,K), NVAL, IER, LOUT)
C T(2,K) = TMID
IF (LINE(1)(66:73) .NE. ' ') CALL IPPARR
1  (LINE(1)(66:73), 0, 1, T(2,K), NVAL, IER, LOUT)
C
T(NT(K),K) = THI
IF (LINE(1)(56:65) .NE. ' ') CALL IPPARR
1  (LINE(1)(56:65), 0, 1, T(NT(K),K), NVAL, IER, LOUT)
C
READ (LINE(2):75), '(',5E15.8')' (A(I,NTR,K),I=1,5)
READ (LINE(3):75), '(',5E15.8')'
1  (A(I,NTR,K),I=6,7), (A(I,1,K),I=1,3)
READ (LINE(4):60), '(',4E15.8')' (A(I,1,K),I=4,7)
GO TO 10
C
40 RETURN
80 FORMAT (6X,'Warning...THERMO cards misplaced will be ignored...')
100 FORMAT (6X,'Error...element...',A,'not declared for...',A)
END
C-----------------------------------------------------------------------------------------------
SUBROUTINE CKREAC (LINE, II, KK, KNAME, LOUT, MAXSP, NSPEC, NREAC, NUNK, NU, NPAR, PAR, NTHB, ITHB,
1  NFAL, IFAL, KFAL, NWL, IWL, WL, KERR)
C
CKREAC parses the main CHAR(*) line representing a gas-phase reaction; first, the real Arhenius parameters are located and stored in PAR[N,I],N=1,NPAR, where I is the reaction number; then a search is made over the reaction string:
C
'='/'=<=': reaction I is reversible;
C
'=': reaction I is irreversible;
C
'(+[n]KNAME(K))': reaction I is a fall-off reaction;
NFAL is incremented, the total number of fall-off reactions;
IFAL(NFAL)=I, KFAL(NFAL)=K;
this species is eliminated from consideration as a reactant or product in this reaction.
C
'+M' : reaction I is a fall-off reaction;
NFAL is incremented, IFAL(NFAL)=I, KFAL(NFAL)=0;
C
'+[n]KNAME(K)': NSPEC(I) is incremented, the total number of species for this reaction;
N is an optional stoichiometric coefficient of KNAME(K), if omitted, n=1;
if this string occurs before the =/=/-, NREAC(I) is incremented, the total number of reactants for this reaction, NUNK(N,I)=K, and NU(N,I) = -n, where N=1-3 is reserved for reactants;
if this string occurs after the =/=/-, NUNK(N,I) = K, and NU(N,I) = n, where N=4-6 is reserved for products;
C
'+M' : I is a third-body reaction; NTHB is incremented, the total number of third-body reactions, and ITHB(NTHB)=I.
C
Input: LINE - a CHAR(*) line (from data file)
II - the index of this reaction, and the total number of reactions found so far.
KK - actual integer number of species
KNAME(K),K=1,KK - array of CHAR(*) species names
OUT - output unit for error messages
MAXSP - maximum number of species allowed in reaction
NPAR - number of parameters expected
A '!' will comment out a line, or remainder of the line.
Output: NSPEC - total number of reactants+products in reaction
NREAC - number of reactants
NUNK - species numbers for the NSPEC species
NU - stoichiometric coefficients for the NSPEC spec.
NFAL - total number of fall-off reactions
IFAL - reaction numbers for the NFAL reactions
KFAL - 3rd body species numbers for the NFAL reactions
NTHB - total number of 3rd-body reactions
ITHB - reaction numbers for the NTHB reactions
NWL - number of radiation-enhanced reactions
IWL - reaction numbers for the NWL reactions
WL - radiation wavelengths for the NWL reactions
KERR - logical, .TRUE. = error in data file

---

C F. Rupley, Div. 8245, 5/13/86

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
DIMENSION NSPEC(*), NREAC(*), NUNK(MAXSP,*), NU(MAXSP,*),
1 PAR(NPAR,*), IFAL(*), KFAL(*), ITHB(*), IWL(*), WL(*),
2 IPLUS(20)
CHARACTER KNAME(*)*(*) , LINE(*)*(*) , INUM(10)*1, UPCASE*4
CHARACTER*80 ISTR, IREAC, IPROD, ISPEC, INAME, ITEMP
LOGICAL KERR, LTHB, LWL
DATA INUM/'0','1','2','3','4','5','6','7','8','9'/

LTHB = .FALSE.
LWL = .FALSE.
NSPEC(II) = 0
NREAC(II) = 0

C-------Find NPAR real parameters-----------------------------
C
CALL IFNPAR (LINE, NPAR, ISTR, ISTART)
CALL IFPARR (ISTR, 1, NPAR, PAR(1,II), NVAL, IER, LOUT)
IF (IER .NE. 0) KERR = .TRUE.
C
C Remove blanks from reaction string
C
INAME = ' 
ILEN = 0
DO 10 I = 1, ISTART-1
   IF (LINE(I:I) .NE. ' ') THEN
      ILEN = ILEN+1
      INAME(ILEN:ILEN) = LINE(I:I)
   ENDIF
10 CONTINUE
C
C Find reaction string, product string
C
II = 0
I2 = 0
DO 25 I = 1, ILEN
   IF (II .LE. 0) THEN
      IF (INAME(I:I+2) .EQ. '<=>') THEN
         II = I
         I2 = I+2
         IR = 1
      ELSEIF (INAME(I:I+1) .EQ. '=>') THEN
         II = I
         I2 = I+1
      ELSEIF (INAME(I:I+1) .EQ. '<=>') THEN
         II = I
         I2 = I+1
      ELSEIF (INAME(I:I+1) .EQ. '=>') THEN
         II = I
         I2 = I+1
      ELSE
         II = I
         I2 = I+1
      ENDIF
   ENDIF
25 CONTINUE
C
IR = -1
ELSEIF (I.GT.1 .AND. INAME(I:I).EQ.'=' .AND. INAME(I-I-I-1).NE.'=') THEN
  I1 = I
  I2 = I
  IR = 1
ENDIF
END IF
25 CONTINUE

C IF (ILASCH(INAME).GE.45 .AND. I1.GT.0) THEN
  WRITE (LOUT, 1900) I1,INAME(:I1-1), (PAR(N,I1),N=1,NPAR)
  WRITE (LOUT, 1920) INAME(I1:)
ELSE
  WRITE (LOUT, 1900) I1,INAME(:45), (PAR(N,I1),N=1,NPAR)
ENDIF

C IREAC = ' ' 
IPROD = '
IF (I1 .GT. 0) THEN 
  IREAC = INAME(:I1-1)
  IPROD = INAME(I1+1:)
ELSE
C----did not find delimiter
C
WRITE (LOUT, 660)
KERR = .TRUE.
RETURN
ENDIF
C
C IF (INDEX(IREAC,'=>').GT.0 .OR. INDEX(IPROD,'=>').GT.0) THEN 
C----more than one '=>'
C
WRITE (LOUT, 800)
KERR = .TRUE.
RETURN
ENDIF
C
C-----Is this a fall-off reaction?
C
IF (INDEX(IREAC,'(+').GT.0 .OR. INDEX(IPROD,'(+').GT.0) THEN
  KRTB = 0
  KPTB = 0
  DO 300 J = 1, 2
    ISTR = ' ' 
    KTB = 0
    IF (J .EQ. 1) THEN
      ISTR = IREAC
    ELSE 
      ISTR = IPROD
    ENDIF
    DO 35 N = 1, ILASCH(ISTR)-1
      IF (ISTR(N:N+1).EQ. '+') THEN
        I1 = N+2
        I2 = I1 + INDEX(ISTR(I1:),'+')-1
        IF (I2 .GT. I1) THEN
          IF (ISTR(I1:I2-1).EQ.'M' .OR. 
            ISTR(I1:I2-1).EQ.'m') THEN
            IF (KTB .NE. 0) THEN
              WRITE (LOUT, 630)
              KERR = .TRUE.
              RETURN
            ELSE
              WRITE (LOUT, 630)
              KERR = .TRUE.
              RETURN
            ENDIF
          ENDIF
        ENDIF
      ENDIF
    ENDDO
  ENDDO
ENDIF
KTB = -1
ENDIF
ELSE
CALL CKCOMP (ISTR(I1:I2-1), KNAME, KK, KNUM)
IF (KNUM .GT. 0) THEN
IF (KTB .NE. 0) THEN
WRITE (LOUT, 630)
KERR = .TRUE.
RETURN
ELSE
KTB = KNUM
ENDIF
ENDIF
ENDIF
IF (KTB .NE. 0) THEN
ITEMP = '  '
IF (II .EQ. 1) THEN
ITEMP = ISTR(I1+1:)
ELSE
ITEMP = ISTR(1:I1-3)/ISTR(I1+1:)
ENDIF
IF (J .EQ. 1) THEN
IREAC = '  '
IREAC = ITEMP
KRTB = KTB
ELSE
IPROD = '  '
IPROD = ITEMP
KPTB = KTB
ENDIF
ENDIF
ENDIF
ENDIF
35 CONTINUE
300 CONTINUE
C
IF (KRTB.NE.0 .OR. KPTB.NE.0) THEN
C


C
ELSEIF (KRTB .EQ. KPTB) THEN
NFAL = NFAL + 1
IFAL(NFAL) = II
KFAL(NFAL) = KRTB

C
ELSE
WRITE (LOUT, 640)
KERR = .TRUE.
RETURN
ENDIF
ENDIF
ENDIF
C
C----------Find reactants, products--------------------------
C
DO 600 J = 1, 2
ISTR = ' 
LTHB = .FALSE.
IF (J .EQ. 1) THEN
ISTR = IREAC
NS = 0
ELSE
ISTR = IPROD
NS = 3
ENDIF
C
C------------store pointers to '+'-signs
C
NPLUS = 1
IPLUS(NPLUS) = 0
DO 500 L = 2, ILASCH(ISTR)-1
   IF (ISTR(L:L).EQ.'+') THEN
      NPLUS = NPLUS + 1
      IPLUS(NPLUS) = L
   ENDIF
500 CONTINUE
NPLUS = NPLUS + 1
IPLUS(NPLUS) = ILASCH(ISTR)+1
C
NSTART = 1
505 CONTINUE
N1 = NSTART
DO 510 N = NPLUS, N1, -1
   ISPEC = ' 
   ISPEC = ISTR(IPLUS(N1)+1 : IPLUS(N)-1)
   IF (UPCASE(ISPEC, 1) .EQ. 'M') THEN
      IF (LTHB) THEN
         WRITE (LOUT, 900)
         KERR = .TRUE.
         RETURN
      ELSEIF (NFAL.GT.0 .AND. IFAL(NFAL).EQ.II) THEN
         WRITE (LOUT, 640)
         KERR = .TRUE.
         RETURN
      ELSE
         LTHB = .TRUE.
         NM = NM + 1
         IF (NM.EQ.NPLUS) GO TO 600
         NSTART = N
         GO TO 505
      ENDIF
   ELSEIF (UPCASE(ISPEC, 2) .EQ. 'HV') THEN
      IF (LWL) THEN
         WRITE (LOUT, 670)
         KERR = .TRUE.
         RETURN
      ELSE
         LWL = .TRUE.
         NWL = NWL + 1
         IWL(NWL) = II
         WL(NWL) = 1.0
         IF (J .EQ. 1) WL(NWL) = -1.0
         IF (N .EQ. NPLUS) GO TO 600
         NSTART = N
         GO TO 505
      ENDIF
   ELSEIF (UPCASE(ISPEC, 1) .EQ. 'M') THEN
      IF (LTHB) THEN
         WRITE (LOUT, 900)
         KERR = .TRUE.
         RETURN
      ELSEIF (NFAL.GT.0 .AND. IFAL(NFAL).EQ.II) THEN
         WRITE (LOUT, 640)
         KERR = .TRUE.
         RETURN
      ELSE
         LTHB = .TRUE.
         NM = NM + 1
         IF (NM.EQ.NPLUS) GO TO 600
         NSTART = N
         GO TO 505
      ENDIF
   ELSEIF (UPCASE(ISPEC, 2) .EQ. 'HV') THEN
      IF (LWL) THEN
         WRITE (LOUT, 670)
         KERR = .TRUE.
         RETURN
      ELSE
         LWL = .TRUE.
         NWL = NWL + 1
         IWL(NWL) = II
         WL(NWL) = 1.0
         IF (J .EQ. 1) WL(NWL) = -1.0
         IF (N .EQ. NPLUS) GO TO 600
         NSTART = N
         GO TO 505
   ENDIF
C
C---------does this string start with a one- or two-digit number?
C
IVAL = 0
CALL CKCOMP (ISPEC(1:1), INUM, 10, I1)
CALL CKCOMP (ISPEC(2:2), INUM, 10, I2)
IF (I1 .GT. 0) THEN
  ITEMP = ' ' 
  IF (I2 .GT. 0) THEN
    IVAL = 10*(I1-1) + (I2-1) 
    ITEMP = ISPEC(2:)
  ELSE
    IVAL = I1-1 
    ITEMP = ISPEC(2:)
  ENDIF
  ISPEC = ' ' 
  ISPEC = ITEMP
ENDIF

CALL CKCOMP (ISPEC, KNAME, KK, KNUM)
IF (KNUM .EQ. 0) THEN
  IF ((N-Nl) .GT. 1) GO TO 510
  WRITE (LOUT, 680) ISPEC(:,ILASCH(ISPEC))
  KERR = .TRUE.
ELSE
  C
  C----------------a species has been found
  C
  IVAL = MAX(IVAL,1)
  IF (J .EQ. 1) IVAL = -IVAL
  NNUM = 0
  DO 111 K = 1, NS
    IF (KNUM.EQ.NUNK(K,II) .AND.
      1 NU(K,II)/IVAL.GT.0) THEN
      C
      C-------------------increment species coefficient count
      C
      NNUM=K 
      NU(NNUM,II) = NU(NNUM,II) + IVAL
  ENDIF
  111 CONTINUE
  IF (NNUM .LE. 0) THEN
    C
    C--------------------are there too many species?
    C
    IF (J.EQ.1 .AND. NS.EQ.3) THEN
      WRITE (LOUT, 690)
      KERR = .TRUE.
      RETURN
    ELSEIF (J.EQ.2 .AND. NS.EQ.MAXSP) THEN
      WRITE (LOUT, 700)
      KERR = .TRUE.
      RETURN
    ELSE
      C
      C----------------------increment species count
      C
      NS = NS + 1
      NSPEC(II) = NSPEC(II)+1
      IF (J .EQ. 1) NREAC(II) = NS
      NUNK(NS,II) = KNUM 
      NU(NS,II) = IVAL
    ENDIF
  ENDIF
C
510 CONTINUE
600 CONTINUE
C
NSPEC(II) = IR*NSPEC(II)
C
630 FORMAT (6X,'Error...more than one fall-off declaration...')
640 FORMAT (6X,'Error in fall-off declaration...')
650 FORMAT (6X,'Error...reaction string not found...')
660 FORMAT (6X,'Error in reaction...')
670 FORMAT (6X,'Error in HV declaration...')
680 FORMAT (6X,'Error...undeclared species...','A)
690 FORMAT (6X,'Error...more than 3 reactants...')
700 FORMAT (6X,'Error...more than 3 products...')
800 FORMAT (6X,'Error in reaction delimiter...')
900 FORMAT (6X,'Error in third-body declaration...')
C
1900 FORMAT (I4,'. ',A,T53,E10.3,F7.3,F11.3)
1900 FORMAT (I4,'. ', A, T53, 1PE8.2, 2X, 0PF5.1, 2X, F9.1)
1920 FORMAT (6X,A)
RETURN
C
SUBROUTINE CKAUXL (SUB, NSUB, II, KK, KNAME, LOUT, MAXSP, NPAR,
1 NSPEC, NTHB, ITHB, NTBS, MAXTB, NKTB, AIK,
2 NFAL, IFAL, IDUP, NFAR, PFAL, IFOP, KFAL, NLAN,
3 ILAN, NLR, PLAN, NREV, IREV, RPAR,
4 NRLT, IRLT, RLAN, NWL, IWL, WL, KERR)
C
CKAUXL parses the auxiliary CHAR(*) lines representing
additional options for a gas-phase reaction; data is stored
based on finding a 'keyword' followed by its required
parameters:

KNAME(K)/val/: this is an enhanced third-body;

if ITHB(NTHB) <> I, this is an error, reaction I is not a
third-body reaction;
else NTBS(NTHB) is incremented,
AIK(NTBS(NTHB),NTHB) = K,
NKTB(NTBS(NTHB)),NTHB) = val;

(LOW,TROE, and SRI define fall-off data):

LOW/val1 val2 val3/: PFAL(N,NFAL) = val(N),N=1,3;

if IFAL(NFAL)<=I, this is an error, reaction I is not a
fall-off reaction;
if ILAN(NLAN)=I, this is an error, cannot have T-L numbers.
if IRLT(NRLT)=I, this is an error, 
if IREV(NREV)=I, this is an error, cannot declare reverse
parameters;
if IFOP(NFAL)>0, this is an error, LOW already declared;
else
IFOP(NFAL) = ABS(IFOP(NFAL))

TROE/val1 val2 val3 [val4]/: PFAL(N,NFAL) = val(N),N=4,7;

if IFAL(NFAL)<=I, this is an error, reaction I is not a
fall-off reaction;
if ILAN(NLAN)=I, this is an error, cannot have T-L numbers.
if IRLT(NRLT)=I, this is an error, 
if IREV(NREV)=I, this is an error, cannot declare reverse
parameters;
if ABS(IFOP(NFAL)).GT.1, this is an error, else if 3 TROE values, IFOP(NFAL) = 3*IFOP(NFAL); if 4 TROE values, IFOP(NFAL) = 4*IFOP(NFAL);
SRI/val1 val2 val3/: PFAL(N,NFAL) = val(N), N=4,6;
if IFAL(NFAL)<I, this is an error, reaction I is not a fall-off reaction; if ILAN(NLAN)=I, this is an error, cannot have T-L numbers. if IREV(NREV)=I, this is an error, cannot declare reverse parameters; if ABS(IFOP(NFAL))>1, this is an error; else if IFOP(NFAL)= 2*IFOP(NFAL);
LT/val1 val2/: if IFAL(NFAL)=I, this is an error, cannot have fall-off and T-L numbers; else increment NLAN, the number of T-L reactions, ILAN(NLAN)=I, PLAN(N,NLAN)=val(N), N=1,2 if IREV(NREV)=I, need IRLT(NRLT)=I.
REV[ERSE]/val1 val2 val3/ : if IFAL(NFAL)=I, this is an error; if IREV(NREV)=I, this is an error, REV already declared; if NSPEC(I)<0, this an error, as I is irreversible; else increment NREV, the number of reactions with reverse parameters given, IREV(NREV)=I, RPAR(N,NREV)=val(N), N=1,3; if ILAN(NLAN)=I, need IRLT(NRLT)=I; if IRLT(NRLT)=I, need ILAN(NLAN)=I.
RLT/val1 val2/: if IFAL(NFAL)=I, this is an error, cannot have fall-off and T-L numbers; if IRLT(NRLT)=I, this is an error, RLT already declared; else increment NRLT, the number of reactions with BOTH reverse parameters given, and T-L numbers; IRLT(NRLT)=I, PLAN(N,NRLT)=val(N), N=1,2; if IREV(NREV)<I, need IREV(NREV)=I; if ILAN(NREV)<I, need ILAN(NLAN)=I;
DUP[LICATE]: This reaction is allowed to be duplicated.
Input: LINE - CHAR*(*) auxiliary information string KK - total number of species declared KNAME- CHAR*(*) species names LOUT - output unit for error messages MAXSP- maximum third bodies allowed in a reaction Output: NTHB - total number of reactions with third bodies ITHB - array of third-body reaction numbers AIK - non-zero third body enhancement factors MKTB - array of species numbers for the third body enhancement factors NFAL - total number of fall-off reactions IFAL - array of fall-off reaction numbers IFOP - array of fall-off type PFAL - fall-off parameters NFAL - total number of Landau-Teller reactions ILAN - array of T-L reaction numbers NLAR - number of Landau-Teller numbers allowed PLAN - array of Landau-Teller numbers
NRLT - total number of 'reverse' T-L reactions
IRLT - array of 'reverse' T-L reaction numbers
RLAN - array of 'reverse' Landau-Teller numbers
NWL - total number of radiation-enhanced reactions
IWL - array of radiation-enhanced reaction numbers
WL - array of wavelengths
KERR - logical, = .TRUE. if error found.

F. Rupley, Div. 8245, 5/27/87

C*** + precision > double
IMPLICIT DOUBLE PRECISION (A-H,0-Z), INTEGER (I-N)
C*** + precision > double
IMPLICIT REAL (A-H,0-Z), INTEGER (I-N)
C*** + precision > single
C*** + precision > single
C
DIMENSION NSPEC(*), ITHB(*), NTBS(*), NKTB(MAXB, *), IDUP(*),
1 AIF(MAXB, *), IFAL(*), KFAL(*), IFOP(*), PFAL(NFAR, *),
2 ILAN(*), PLAN(NLAN, *), IREV(*), RPAR(NPAR, *), IRLT(*),
3 RLAN(NLAN, *), IWL(*), WL(*), VAL(I)
CHARACTER SUB(*)*(*), KNAME(*)*(*), KEY*80, RSTR*80, UPCASE*4
CHARACTER SUB.CH (80)*80,RSTR.CH*80,LINE.CH*80
LOGICAL KERR, LLAN, LRLT, LTHB, LFAL, LTRO, LSRI, LWL, LREV
C
LTHB = (NTHB.GT.0 .AND. ITHB(NTHB).EQ.II)
LFAL = (NFAL.GT.0 .AND. IFAL(NFAL).EQ.II)
LWL = (NWL.GT.0 .AND. IWL(NWL).EQ.II)
LREV = (NREV.GT.0 .AND. IREV(NREV).EQ.II)
LLAN = (NLAN.GT.0 .AND. ILAN(NLAN).EQ.II)
LRLT = (NRLT.GT.0 .AND. IRLT(NRLT).EQ.II)
LTRO = (NFAL.GT.0 .AND. IFAL(NFAL).EQ.II .AND. IFOP(NFAL).GT.2)
LSRI = (NFAL.GT.0 .AND. IFAL(NFAL).EQ.II .AND. IFOP(NFAL).EQ.2)
C
DO 500 N = 1, NSUB
ILEN = ILASCH(SUB(N))
KEY = ' ';
IF (UPCASE(SUB(N), 3).EQ. 'DUP') THEN
IDUP(II) = -1
WRITE (LOUT, 4000)
GO TO 500
ELSE
I1 = INDEX(SUB(N), '/')
I2 = INDEX(SUB(N)(I1+1:), '/')
IF (I1.LE.0 .OR. I2.LE.0) THEN
KERR = .TRUE.
WRITE (LOUT, 2090) SUB(N)(:ILEN)
GO TO 500
ENDIF
KEY = SUB(N)(:I1-1)
RSTR = ' ';
RSTR = SUB(N)(I1+I1+I2-1)
ENDIF
C
**********W.ING 6/16/95***********************
IF (UPCASE(KEY, 3).EQ.'LOW'.OR.
1 UPCASE(KEY, 4).EQ.'TROE'.OR.
2 UPCASE(KEY, 4).EQ.'EXTR'.OR.
3 UPCASE(KEY, 4).EQ.'CHEB'.OR.
4 UPCASE(KEY, 3).EQ.'SRI') THEN
C
FALL-OFF DATA
C
IF ((.NOT.LFAL).OR.LLAN. OR. LRLT .OR. LREV) THEN
KERR = .TRUE.
IF (.NOT. LFAL) WRITE (LOUT, 1050) SUB(N)(:ILEN)
IF (ILAN) WRITE (LOUT, 1060) SUB(N)(:ILEN)
IF (ILRT) WRITE (LOUT, 1070) SUB(N)(:ILEN)
C
IF (LREV) WRITE (LOUT, 1090) SUB(N)(:ILEN)
ELSE
IF (UPCASE(KEY, 3) .EQ. 'LOW') THEN
  IF (IFOP(NFAL) .GT. 0) THEN
    WRITE (LOUT, 2000) SUB(N)(:ILEN)
    KERR = .TRUE.
  ELSE
    IFOP(NFAL) = ABS(IFOP(NFAL))
    CALL IPPARR (RSTR, 1, 3, PFAL(1, NFAL), NVAL, IER, LOUT)
    IF (IER .NE. 0) KERR = .TRUE.
    WRITE (LOUT, 3050) (PFAL(L, NFAL), L=1, 3)
  ENDIF
ELSEIF (UPCASE(KEY, 4) .EQ. 'TROE') THEN
  IF (LTRO .OR. LSRI) THEN
    KERR = .TRUE.
    IF (LTRO) WRITE (LOUT, 2010) SUB(N)(:ILEN)
    IF (LSRI) WRITE (LOUT, 2030) SUB(N)(:ILEN)
  ELSE
    LTRO = .TRUE.
    CALL IPPARR (RSTR, 1, -4, PFAL(4, NFAL), NVAL, IER, LOUT)
    IF (NVAL .EQ. 3) THEN
      IFOP(NFAL) = 3*IFOP(NFAL)
      WRITE (LOUT, 2080) (PFAL(L, NFAL), L=4, 6)
    ELSEIF (NVAL .EQ. 4) THEN
      IFOP(NFAL) = 4*IFOP(NFAL)
      WRITE (LOUT, 2090) (PFAL(L, NFAL), L=4, 7)
    ELSE
      WRITE (LOUT, 2020) SUB(N)(:ILEN)
      KERR = .TRUE.
    ENDIF
  ENDIF
ELSEIF (UPCASE(KEY, 4) .EQ. 'CHEB') THEN
  LTRO = .TRUE.
  CALL IPPARR (RSTR, 1, -6, PFAL(4, NFAL), NVAL, IER, LOUT)
  N_CHEB = PFAL(4, NFAL)
  M_CHEB = PFAL(5, NFAL)
  NPARA_CHEB = N_CHEB*M_CHEB + 2
  INDEX_CHEB = 0
  IFOP(NFAL) = 6*IFOP(NFAL)
  write('(*,*)'..NFAL.IFOP...'NFAL.IFOP...',NFAL,IFOP(NFAL))
  WRITE('(*,*)'..NPARA_CHEX..')
  CALL IPPARR (RSTR, 1, -5, PFAL(4, NFAL), NVAL, IER, LOUT)
  INDEX_CHEB = INDEX_CHEB + NVAL
  WRITE (LOUT, 2080) SUB_CH(N)(:ILEN)
  WRITE(*,*)'..NPARA..INDEX..NVAL.',NPARA_CHEB,NVAL
  WRITE(*,*)'..N..M...',N_CHEB,M_CHEB
  WRITE (LOUT, 3098) (PFAL(L, NFAL), L=6, NVAL+3)
  CONTINUE
LINE_CH = ',
READ(5,'(A)') LINE_CH
WRITE(*,'..LINE_CH..')
WRITE(*,*)LINE_CH
C
ILEN = IPPLEN(LINE_CH)
CALL CKISUB (LINE_CH(:ILEN), SUB_CH, NSUB_CH)
C
KEY = '1'
DO 151 N_CH = 1, NSUB_CH
  ILEN = ILASCH(SUB_CH(NSUB_CH))
  WRITE(*,'(*,*)' NSUB_CH = ',NSUB_CH)
  WRITE(*,'(*,*)SUB_CH(NSUB_CH)
  I11 = INDEX(SUB_CH(NSUB_CH),'/')
  I12 = INDEX(SUB_CH(NSUB_CH)(I11+1:),'/')
WRITE(*,'(I11..112...)',I11,I12)
IF (I11.LE.0 .OR. I12.LE.0) THEN
  KERR = .TRUE.
  WRITE (LOUT, 2090) SUB_CH(N_CH)(:IILEN)
ENDIF
KEY = SUB(N)(:11-1)
RSTR_CH = ' '
RSTR_CH = SUB_CH(NSUB_CH)(II1+1:II1+II2-1)
WRITE(*,'RSTR',RSTR_CH)
CALL IPPARR (RSTR_CH,1,-8,PFAL(INDEX_CHEB+4,NFAL),NVAL,IER,LOUT)
WRITE(*,'INDEX..NVAL..',INDEX_CHEB, NVAL)
WRITE(*,'IER..= ',IER)
INDEX_CHEB = INDEX_CHEB + NVAL
WRITE(*,'INDEX...1,INDEX_CHEB')
CONTINUE
IF (INDEX_CHEB.LT.NPARA_CHEB) GOTO 101
CONTINUE
IF (INDEX_CHEB.EQ.NPARA_CHEB) THEN
ELSE
WRITE(LOUT, 2028) SUB_CH(N)(:ILEN)
KERR = .TRUE.
ENDIF
ELSEIF (UPCASE(KEY, 4) .EQ. 'EXTR') THEN
IF (LTRO .OR. LSRI) THEN
  KERR = .TRUE.
  IF (LTRO) WRITE (LOUT, 2015) SUB(N)(:ILEN)
  IF (LSRI) WRITE (LOUT, 2035) SUB(N)(:ILEN)
ELSE
  LTRO = .TRUE.
  CALL IPPARR (RSTR,1,-5,PFAL(4,NFAL),NVAL,IER,LOUT)
  IF (NVAL EQ. 5) THEN
    IFOP(NFAL) = 5*IFOP(NFAL)
    WRITE(*,'IFOP',IFOP(NFAL))
    WRITE (LOUT, 3095) (PFAL(L,NFAL),L = 4,8)
  ELSE
    WRITE (LOUT, 2025) SUB(N)(:ILEN)
    KERR = .TRUE.
  ENDIF
ELSEIF (UPCASE(KEY, 3) .EQ. 'SRI') THEN
ELSEIF (UPCASE(KEY, 3) .EQ. 'SRI') THEN
  IF (LTRO .OR. LSRI) THEN
    KERR = .TRUE.
    IF (LTRO) WRITE (LOUT, 2030) SUB(N)(:ILEN)
    IF (LSRI) WRITE (LOUT, 2040) SUB(N)(:ILEN)
ELSE
  LSRI = .TRUE.
  IFOP(NFAL) = 2*IFOP(NFAL)
  CALL IPPARR (RSTR,1,-5,PFAL(4,NFAL),NVAL,IER,LOUT)
  IF (NVAL EQ. 3) THEN
    PFAL(7,NFAL) = 1.0
    PFAL(8,NFAL) = 0.0
    WRITE (LOUT, 3060) (PFAL(L,NFAL),L = 4,8)
  ELSEIF (NVAL EQ. 5) THEN
    WRITE (LOUT, 3070) (PFAL(L,NFAL),L = 4,8)
ELSE

WRITE (LOUT, 2020) SUB(N)(:ILEN)
KERR = .TRUE.
ENDIF
ENDIF
ENDIF
ENDIF
ELSEIF (UPCASE(KEY, 3) .EQ. 'REV') THEN

REVERSE ARRHENIUS PARAMETERS

IF (LFAL .OR. LREV .OR. NSPEC(II) .LT. 0) THEN
  KERR = .TRUE.
  IF (LFAL) WRITE (LOUT, 1090) SUB(N)(:ILEN)
  IF (LREV) WRITE (LOUT, 2050) SUB(N)(:ILEN)
  IF (NSPEC(II) .LT. 0) WRITE (LOUT, 2060) SUB(N)(:ILEN)
ELSE
  LREV = .TRUE.
  NREV = NREV + 1
  IREV(NREV) = II
  CALL IPPARR (RSTR, 1, NPAR, RPAR(1, NREV), NVAL, IER, LOUT)
  IF (IER .NE. 0) KERR = .TRUE.
  WRITE (LOUT, 1900) ' Reverse Arrhenius coefficients:',
                    (RPAR(L, NREV), L = 1, 3)
ENDIF
ELSEIF (UPCASE(KEY, 3) .EQ. 'RLT') THEN

REVERSE LANDAU-TELLER PARAMETERS

IF (LFAL .OR. LRLT .OR. NSPEC(II) .LT. 0) THEN
  KERR = .TRUE.
  IF (LFAL) WRITE (LOUT, 1070) SUB(N)(:ILEN)
  IF (LRLT) WRITE (LOUT, 2080) SUB(N)(:ILEN)
  IF (NSPEC(II) .LT. 0) WRITE (LOUT, 1080) SUB(N)(:ILEN)
ELSE
  LRLT = .TRUE.
  NRLT = NRLT + 1
  IRLT(NRLT) = II
  CALL IPPARR (RSTR, 1, NLR, RLAN(1, NRLT), NVAL, IER, LOUT)
  IF (IER .NE. 0) KERR = .TRUE.
  WRITE (LOUT, 3040) (RLAN(L, NRLT), L = 1, 2)
ENDIF
ELSEIF (UPCASE(KEY, 2) .EQ. 'HV') THEN

RADIATION WAVELENGTH ENHANCEMENT FACTOR

IF (.NOT. LWL) THEN
  WRITE (LOUT, 1000) SUB(N)(:ILEN)
  KERR = .TRUE.
ELSE
  CALL IPPARR (RSTR, 1, 1, VAL, NVAL, IER, LOUT)
  IF (IER .EQ. 0) THEN
    WL(NWL) = WL(NWL) * VAL(1)
    WRITE (LOUT, 3020) ABS(WL(NWL))
  ELSE
    WRITE (LOUT, 1000) SUB(N)(:ILEN)
    KERR = .TRUE.
  ENDIF
ENDIF
ELSEIF (UPCASE(KEY, 2) .EQ. 'LT') THEN

LANDAU-TELLER PARAMETERS

IF (LFAL .OR. LLAN) THEN
    KERR = .TRUE.
    IF (LFAL) WRITE (LOUT, 1060) SUB(N)(:ILEN)
    IF (LLAN) WRITE (LOUT, 2070) SUB(N)(:ILEN)
ELSE
    LLAN = .TRUE.
    NLAN = NLAN + 1
    ILAN(NLAN) = II
    CALL IPPARR (RSTR, 1, NLAR, PLAN(1,NLAN), NVAL, IER, LOUT)
    IF (IER .NE. 0) THEN
        WRITE (LOUT, 1010) SUB(N)(:ILEN)
        KERR = .TRUE.
    ENDIF
    WRITE (LOUT, 3000) (PLAN(L,NLAN), L=1,2)
ENDIF
C
ELSE
ENHANCED THIRD BODIES

CALL CKCOMP (KEY, KNAME, KK, K)
IF (K .EQ. 0) THEN
    WRITE (LOUT, 1040) KEY(:ILASCH(KEY))
    KERR = .TRUE.
ELSE IF (.NOT.LTHB) THEN
    KERR = .TRUE.
    WRITE (LOUT, 1020) SUB(N)(:ILEN)
ELSE
    IF (NTBS(NTHB) .EQ. MAXTB) THEN
        KERR = .TRUE.
        WRITE (LOUT, 1030) SUB(N)(:ILEN)
    ELSE
        CALL IPPARR (RSTR, 1, 1, VAL, NVAL, IER, LOUT)
        IF (IER .EQ. 0) THEN
            WRITE (LOUT, 3010) KNAME(K), VAL(1)
            NTBS(NTHB) = NTBS(NTHB) + 1
            NKTB(NTBS(NTHB), NTHB) = K
            AIK(NTBS(NTHB), NTHB) = VAL(1)
        ELSE
            WRITE (LOUT, 1020) SUB(N)(:ILEN)
            KERR = .TRUE.
        ENDIF
    ENDIF
ENDIF
ENDIF
ENDIF
C
FORMATS
C
1000 FORMAT (6X,'Error in HV declaration...',A)
1010 FORMAT (6X,'Error in LT declaration...',A)
1020 FORMAT (6X,'Error in third body declaration...',A)
1030 FORMAT (6X,'Error...more than MAXTB third bodies...',A)
1040 FORMAT (6X,'Error...undeclared species...',A)
1050 FORMAT (6X,'Error...this is not a fall-off reaction...',A)
1060 FORMAT (6X,'Error...LT declared in fall-off reaction...',A)
1070 FORMAT (6X,'Error...RLT declared in fall-off reaction...',A)
1080 FORMAT (6X,'Error...RLT declared in irreversible reaction...',A)
1090 FORMAT (6X,'Error...REV declared in fall-off reaction...',A)
2000 FORMAT (6X,'Error...LOW declared more than once...',A)
2010 FORMAT (6X,'Error...TROE declared more than once...',A)
2015 FORMAT (6X,'Error...EXTROE declared more than once...',A)
2020 FORMAT (6X,'Error in fall-off parameters...',A)
2025 FORMAT (6X,'Error in EXTROE fall-off parameters...',A)
2028 FORMAT (6X,'Error in CHEB fall-off parameters...','A)
2030 FORMAT (6X,'Error...cannot use both TROE and SRI...','A)
2035 FORMAT (6X,'Error...cannot use TROE, EXTROE or SRI...','A)
2040 FORMAT (6X,'Error...SRI declared more than once...','A)
2050 FORMAT (6X,'Error...SRI declared more than once...','A)
2060 FORMAT (6X,'Error...REV declared for irreversible reaction...','A)
2070 FORMAT (6X,'Error...RLT declared more than once...','A)
2080 FORMAT (6X,'Error in auxiliary data...','A)
3000 FORMAT (9X,'Landau-Teller parameters: B=','E12.5',' C=','E12.5)
3010 FORMAT (9X,A16,' Enhanced by ',1PE12.3)
3020 FORMAT (9X,'Radiation wavelength (A): ',F10.2)
3030 FORMAT (6X,A,T51,E10.3,F7.3,F11.3)
3040 FORMAT (6X,'Error...REV declared more than once...','A)
3050 FORMAT (6X,'Error...cannot use both TROE and SRI...','A)
3060 FORMAT (6X,'SRI declared more than once...','A)
3070 FORMAT (6X,'SRI declared more than once...','A)
3080 FORMAT (6X,'TROE declared more than once...','A)
3090 FORMAT (6X,'TROE declared more than once...','A)
3095 FORMAT (6X,'EXTROE declared more than once...','A)
3098 FORMAT (6X,'CHEB Polynomials: ',8E13.5)
4000 FORMAT (6X,'Declared duplicate reaction...')
END

C---**precision > double**-----------------------------C
C DIMENSION WTM(*), KPHSE(*), KCHRG(*), T(MAXTP,*),
1 NT(*), KNCF(MDIM,*) , IPLUS(IO)
LOGICAL KERR, ITHRM(*)
CHARACTER ENAME(*)*(*) , KNAME(*)*(*) , IPHSE(3)*1, INUM(10)*1
DATA IPHSE/'S', 'G', 'L'/
DATA INUM/'O', '1', '2', '3', '4', '5', '6', '7', '8', '9'/

C WRITE (LOUT, 400) (ENAME(M), M = 1, MM)
WRITE (LOUT, 300)
C
DO 100 K = 1, KK
C
IF (T(1,K) .LT. 0.0) T(1,K) = TLO
IF (T(2,K) .LT. 0.0) T(2,K) = TMID
IF (T(3,K) .LT. 0.0) T(NT(K),K) = THI
WRITE (LOUT, 500) K, KNAME(K), IPHSE(KPHSE(K)+2) ,
1 WTM(K), T(1,K) , T(NT(K),K) , (KNCF(M,K),M=1,MM)
C
IF (T(1,K) .GE. T(NT(K),K) ) THEN
KERR = .TRUE.
WRITE (LOUT, 240)
ENDIF
IF (T(1,K) .GT. T(2,K)) THEN
WRITE (LOUT, 250)
KERR = .TRUE.
ENDIF
IF (T(NT(K),K) .LT. T(2,K)) THEN
WRITE (LOUT, 260)
ENDIF
each species must have thermodynamic data

IF (.NOT. ITHRM(K)) THEN
  KERR = .TRUE.
  WRITE (LOUT, 200)
ENDIF

a species cannot start with a number

CALL CKCOMP (KNAME(K)(:1), INUM, 10, I)
IF (I .GT. 0) THEN
  KERR = .TRUE.
  WRITE (LOUT, 210)
ENDIF

if '+' sign is used in a species name,
  examples of legal species symbols with + are:
  OH(+2), OH(+), OH++, OH(+), OH(++)
  OH(+OH), OH2+, OH+2
  examples of illegal species symbols with + are:
  +OH (symbol starts with a +, this will cause confusion in a reaction)
  OH(+OH) (symbol in parentheses is another species- this arrangement is reserved for a fall-off reaction)
  OH+OH (plus delimits other species names, this will cause confusion in a reaction)

NPLUS = 0
DO 50 N = 1, ILASCH(KNAME(K))
  IF (KNAME(K)(N:N) .EQ. '+') THEN
    NPLUS = NPLUS + 1
    IPLUS(NPLUS) = N
  ENDIF
50 CONTINUE
DO 60 N = 1, NPLUS
  II = IPLUS(N)
  IF (II .EQ. 1) THEN
    WRITE (LOUT, 220)
    KERR = .TRUE.
  ELSE
    IF (KNAME(K)(11-1:11-1) .EQ. '(') THEN
      11 = II + INDEX(KNAME(K)(II:),')')-1
      IF (12 .GT. II) THEN
        CALL CKCOMP (KNAME(K)(II:12-1), KNAME, KK, KNUM)
        IF (KNUM .GT. 0) THEN
          WRITE (LOUT, 230)
          KERR = .TRUE.
        ENDIF
      ENDIF
    ENDIF
  ENDIF
60 CONTINUE

is there another species name in parentheses

IF (KNAME(K)(I1-1:I1-1) .EQ. '()' THEN
  I1 = I1 + 1
  I2 = I1 + INDEX(KNAME(K)(I1:)',')'-1
  IF (I2 .GT. II) THEN
    CALL CKCOMP (KNAME(K)(II:I2-1), KNAME, KK, KNUM)
    IF (KNUM .GT. 0) THEN
      WRITE (LOUT, 230)
      KERR = .TRUE.
    ENDIF
  ENDIF
ENDIF

is there another species name after a +

I1 = I1 + 1
IF (N .LT. NPLUS) THEN
  DO 55 L = N+1, NPLUS
    I2 = IPLUS(L)
  END
IF (I2 .GT. II) THEN
  CALL CKCOMP (KNAME(K)(I2:I2-1), KNAME, KK, KNUM)
ENDIF

IF (KNUM .GT. 0) THEN
  WRITE (LOUT, 230)
  KERR = .TRUE.
ENDIF

CONTINUE

ENDIF

I2 = ILASCH(KNAME(K))

IF (I2 .GE. II) THEN
  CALL CKCOMP (KNAME(K)(I1:I2), KNAME, KK, KNUM)
  IF (KNUM .GT. 0) THEN
    WRITE (LOUT, 230)
    KERR = .TRUE.
  ENDIF
ENDIF

CONTINUE

100 CONTINUE

WRITE (LOUT, 300)
RETURN

200 FORMAT (6X,'Error...no thermodynamic properties for species')
210 FORMAT (6X,'Error...species starts with a number')
220 FORMAT (6X,'Error...species starts with a plus')
230 FORMAT (6X,'Error...illegal + in species name')
240 FORMAT (6X,'Error...High temperature must be < Low temperature')
250 FORMAT (6X,'Error...Low temperature must be <= Mid temperature')
260 FORMAT (6X,'Error...High temperature must be => Mid temperature')
300 FORMAT (6x,'Error...species starts with a number')

400 FORMAT (1X, 'SPECIES', 79('-'))

500 FORMAT (14,'. ',A10,2X, Al, T24,  Al, T2 6,11,T 2 8 ,F 9 .5, T 3 8 ,  F 6 .1,T4 5,F 6 .1,  
T51,15(13))

END
AUNITS - character string which describes the input units of A, the pre-exponential factor PAR(1,I)
EUNITS - character string which describes the input units of E, the activation energy PAR(3,I)
NREAC - array of the number of reactants in the reactions
NUNK - matrix of the species numbers of the reactants and products in the reactions
NU - matrix of the stoichiometric coefficients of the reactants and products in the reactions
KCHRG - array of the electronic charges of the species
MDIM - the maximum number of elements allowed
MM - the actual number of elements declared
KNCF - matrix of elemental composition of the species
IDUP - array of integer flags to indicate duplicate reactions
NFAL - total number of reactions with fall-off
IFAL - array of the NFAL reaction numbers
NFAR - maximum number of fall-off parameters allowed
PFAL - matrix of fall-off parameters for the NFAL reactions
IFOP - array of integer fall-off types for the NFAL reactions
NREV - total number of reactions with reverse parameters
IREV - array of the NREV reaction numbers
NTHB - total number of reactions with third-bodies
ITHB - array of the NTHB reaction numbers
NLAN - total number of reactions with Landauer-Teller parameters
ILAN - array of the NLAN reaction numbers
NRLT - total number of reactions with reverse Landauer-Teller parameters
ILRT - array of the NRLT reaction numbers
KERR - logical error flag
LOUT - unit number for output messages

---
C
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
DIMENSION NSPEC(*), PAR(NPAR,*), RFAR(NPAR,*), NREAC(*),
1 NUNK(MAXSP,*), NU(MAXSP,*), KCHRG(*), KNCF(MDIM,*),
2 IDUP(*), IFAL(*), KFAL(*), PFAL(NPAR,*), IFOP(*),
3 IREV(*), ITHB(*), ILAN(*), ILRT(*)
CHARACTER(*) AUNITS, EUNITS
LOGICAL IERR, KERR, LREV, LLAN, LRLT
C
CALL CK3AL (MAXSP, NUNK(1,I), NU(1,I), MDIM, MM, KCHRG, KNCF, 1
1 IERR)
C
IF (IERR) THEN
KERR = .TRUE.
WRITE (LOUT, 1060)
ENDIF
C
CALL CKDUP (II, MAXSP, NSPEC, NREAC, NU, NUNK, NFAL, IFAL, KFAL, 1
1 ISAME)
C
IF (ISAME .GT. 0) THEN
IF (IDUP(ISAME).NE.0 .AND. IDUP(II).NE.0) THEN
IDUP(ISAME) = ABS(IDUP(ISAME))
IDUP(II) = ABS(IDUP(II))
ENDIF
ENDIF
ELSE
  N1 = 0
  N2 = 0
  IF (NTHB .GT. 1) THEN
    DO 150 N = 1, NTHB
      IF (ITHB(N) .EQ. ISAME) N1 = 1
      IF (ITHB(N) .EQ. II) N2 = 1
    150 CONTINUE
  ENDIF
  IF (N1 .EQ. N2) THEN
    KERR = .TRUE.
    WRITE (LOUT, 1050) ISAME
  ENDIF
ENDIF
ENDIF
C
IF (NFAL.GT.0 .AND. IFAL(NFAL).EQ.II .AND. IFOP(NFAL).LT.0) THEN
  KERR = .TRUE.
  WRITE (LOUT, 1020)
ENDIF
C
LREV = (NREV.GT.0 .AND. IREV(NREV).EQ.II)
LLAN = (NLAN.GT.0 .AND. ILAN(NLAN).EQ.II)
LRLT = (NRLT.GT.0 .AND. IRLT(NRLT).EQ.II)
IF (LREV .AND. LLAN .AND. (.NOT.LRLT)) THEN
  KERR = .TRUE.
  WRITE (LOUT, 1030)
ENDIF
IF (LRLT .AND. (.NOT.LLAN)) THEN
  KERR = .TRUE.
  WRITE (LOUT, 1040)
ENDIF
IF (LRLT .AND. (.NOT.LREV)) THEN
  KERR = .TRUE.
  WRITE (LOUT, 1045)
ENDIF
C
IF (EUNITS .EQ. 'KELV') THEN
  EFAC = 1.0
ELSEIF (EUNITS .EQ. 'CAL') THEN
  convert E from cal/mole to Kelvin
  EFAC = 1.0 / 1.987
ELSEIF (EUNITS .EQ. 'KCAL') THEN
  convert E from kcal/mole to Kelvin
  EFAC = 1000.0 / 1.987
ELSEIF (EUNITS .EQ. 'JOUL') THEN
  convert E from Joules/mole to Kelvin
  EFAC = 1.0 / 8.314
ELSEIF (EUNITS .EQ. 'KJOU') THEN
  convert E from Kjoules/mole to Kelvin
  EFAC = 1000.0 / 8.314
ENDIF
PAR(3,II) = PAR(3,II) * EFAC
C
IF (NREV.GT.0 .AND. IREV(NREV).EQ.II)  RPAR(3,II)=RPAR(3,II)*EFAC
C
IF (NFAL.GT.0 .AND. IFAL(NFAL).EQ.II)  PPAL(3,II)=PPAL(3,II)*EFAC

C
IF (AUNITS .EQ. 'MOLC') THEN
  NSTOR = 0
  NSTOP = 0
  DO 50 N = 1, MAXSP
C IF (NU(N,II) .LT. 0) THEN
sum of stoichiometric coefficients of reactants
NSTOR = NSTOR + ABS(NU(N,II))
ENDIF
ELSEIF (NU(N,II) .GT. 0) THEN
sum of stoichiometric coefficients of products
NSTOP = NSTOP + NU(N,II)
ENDIF
CONTINUE
C AVAG = 6.023E23
C IF (NFAL.GT.0 .AND. IFAL(NFAL).EQ.II) THEN
fall-off reaction, "(+M)" or "(+species name)" does not count except in "LOW" A-factor;
reverse-rate declarations are not allowed
IF (NSTOR.GT.0) PAR(1,II) = PAR(1,II) * AVAG**(NSTOR-1)
NSTOR = NSTOR + 1
IF (NFAL.GT.0) PFAL(1,NFAL) = PFAL(1,NFAL)*AVAG**(NSTOR-1)
ENDIF
C ELSEIF (NTHB.GT.0 .AND. ITHB(NTHB).EQ.II) THEN
third body reaction, "+M" counts as species in forward and reverse A-factor conversion
NSTOR = NSTOR + 1
NSTOP = NSTOP + 1
IF (NSTOR.GT.0) PAR(1,II) = PAR(1,II) * AVAG**(NSTOR-1)
IF (NREV.GT.0 .AND. IREV(NREV).EQ.II .AND. NSTOP.GT.0)
RPAR(1,NREV) = RPAR(1,NREV) * AVAG**(NSTOP-1)
ENDIF
C ELSE
not third-body or fall-off reaction, but may have reverse rates.
IF (NSTOR .GT. 0) PAR(1,II) = PAR(1,II) * AVAG**(NSTOR-1)
IF (NREV.GT.0 .AND. IREV(NREV).EQ.II .AND. NSTOP.GT.0)
RPAR(1,NREV) = RPAR(1,NREV) * AVAG**(NSTOP-1)
ENDIF
C SUBROUTINE CKBAL (NSPEC, KSPEC, KCOEF, MDIM, MM, KCHRG, KNCF, KERR)
Checks elemental balance of reactants vs. products.
Checks charge balance of reaction.
Input: NSPEC - total number of species in this reaction
KSPEC(N),N=1,NSPEC- array of species numbers in reaction
KCOEF(N) - stoichiometric coefficients of the species
MDIM - maximum number of elements allowed
MM - actual integer number of elements
KCHRG(K) - ionic charge Kth species
KNCF(M,K)- integer elemental composition of Kth species
Output: KERR - logical, =.TRUE. if reaction does not balance
F. Ruple, Div. 8245, 5/13/86

C---------------------------------------------------------------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
DIMENSION KSPEC(*), KCOEF(*), KNCF(MDIM,*), KCHRG(*)
LOGICAL IERR
C
IERR = .FALSE.
C
charge balance
C
KBAL = 0
DO 50 N = 1, ABS(NSPEC)
   KBAL = KBAL + KCOEF(N)*KCHRG(KSPEC(N))
50 CONTINUE
IF (KBAL .NE. 0) IERR = .TRUE.
C
element balance
C
DO 100 M = 1, MM
   MBAL = 0
   DO 80 N = 1, ABS(NSPEC)
      MBAL = MBAL + KCOEF(N)*KNCF(M, KSPEC(N))
80 CONTINUE
IF (MBAL .NE. 0) IERR = .TRUE.
100 CONTINUE
RETURN
END

C---------------------------------------------------------------C

SUBROUTINE CKDUP (I, MAXSP, NS, NR, NU, NUNK, NFAL, IFAL, KFAL, ISAME)
C
Checks reaction I against the (I-1) reactions for duplication
C---------------------------------------------------------------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
DIMENSION NS(*), NR(*), NU(MAXSP,*), NUNK(MAXSP,*), IFAL(*), KFAL(*)
C
ISAME = 0
NRI = NR(I)
NPI = ABS(NS(I)) - NR(I)
C
DO 500 J = 1, I-I
C
   NRJ = NR(J)
   NPJ = ABS(NS(J)) - NR(J)
C
   IF (NRJ.EQ.NRI .AND. NPJ.EQ.NPI) THEN
      NSAME = 0
      DO 20 N = 1, MAXSP
         KI = NUNK(N,I)
         NI = NU(N,I)
      20 CONTINUE
      DO 15 L = 1, MAXSP
      15 CONTINUE

   END IF
500 CONTINUE

RETURN
END
KJ = NUNK(L,J)
NJ = NU(L,J)
IF (NJ.NE.0 .AND. KJ.EQ.KI .AND. NJ.EQ.NI)
    NSAME = NSAME + 1
1
15 CONTINUE
20 CONTINUE

IF (NSAME.EQ.ABS(NS(J))) THEN
    same products, reactants, coefficients, check fall-off
    third body
    IF (NFAL.GT.0 .AND. IFAL(NFAL).EQ.I) THEN
        DO 22 N = 1, NFAL-1
            IF (J.EQ.IFAL(N) .AND. KFAL(N).EQ.KFAL(NFAL)) THEN
                ISAME = J
                RETURN
            ENDIF
        22 CONTINUE
        RETURN
    ENDIF
    ISAME = J
    RETURN
ENDIF

IF (NPI.EQ.NRJ .AND. NPJ.EQ.NRI) THEN
    NSAME = 0
    DO 30 N = 1, MAXSP
        KI = NUNK(N,I)
        NI = NU(N,I)
    C
    DO 25 L = 1, MAXSP
        KJ = NUNK(L,J)
        NJ = NU(L,J)
        IF (NJ.NE.0 .AND. KJ.EQ.KI .AND. -NJ.EQ.NI)
            NSAME = NSAME + 1
        25 CONTINUE
    30 CONTINUE
    C
    IF (NSAME.EQ.ABS(NS(J)) .AND.
        (NS(J).GT.0 .OR. NS(I).GT.0)) THEN
        C
        same products as J reactants, and vice-versa
        C
        IF (NFAL.GT.0 .AND. IFAL(NFAL).EQ.I) THEN
            DO 32 N = 1, NFAL-1
                IF (J.EQ.IFAL(N) .AND. KFAL(N).EQ.KFAL(NFAL)) THEN
                    ISAME = J
                    RETURN
                ENDIF
            32 CONTINUE
            RETURN
        ENDIF
        C
        ISAME = J
        RETURN
    ENDIF
C
500 CONTINUE
RETURN
END
SUBROUTINE CKISUB (LINE, SUB, NSUB)
C Generates an array of CHAR(*) substrings from a CHAR(*) string,
C using blanks or tabs as delimiters
C C Input: LINE - a CHAR(*) line
C C Output: SUB - a CHAR(*) array of substrings
C C NSUB - number of substrings found
C A '!' will comment out a line, or remainder of the line.
C F. Rupley, Div. 8245, 5/15/86
C----------------------------------------
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 C CHARACTER(*) SUB(*), LINE
NSUB = 0
C DO 5 N = 1, LEN(LINE)
   IF (ICHAR(LINE(N:N)) .EQ. 9) LINE(N:N) = '  '
5 CONTINUE
C IF (IPPLEN(LINE) .LE. 0) RETURN
C ILEN = ILASCH(LINE)
C NSTART = IFIRCH(LINE)
10 CONTINUE
ISTART = NSTART
NSUB = NSUB + 1
SUB(NSUB) = '  '
C DO 100 I = ISTART, ILEN
   ILAST = INDEX(LINE(ISTART:), '  *  )  - 1
   IF (ILAST .GT. 0) THEN
      ILAST = ISTART + ILAST - 1
   ELSE
      ILAST = ILEN
   ENDIF
   SUB(NSUB) = LINE(ISTART:ILAST)
   IF (ILAST .EQ. ILEN) RETURN
C NSTART = ILAST + IFIRCH(LINE(ILAST+1:))
C Does SUB have any slashes?
C I1 = INDEX(SUB(NSUB), '/')
   IF (I1 .LE. 0) THEN
      IF (LINE(NSTART:NSTART) .NE. '/') GO TO 10
      NEND = NSTART + INDEX(LINE(NSTART+1:),'/')
      IND = INDEX(SUB(NSUB), '  ')
      SUB(NSUB)(IND:) = LINE(NSTART:NEND)
      IF (NEND .EQ. ILEN) RETURN
      NSTART = NEND + IFIRCH(LINE(NEND+1:))
   GO TO 10
   ENDIF
C Does SUB have 2 slashes?
C I2 = INDEX(SUB(NSUB)(I1+1:], '/')
   IF (I2 .GT. 0) GO TO 10
   NEND = NSTART + INDEX(LINE(NSTART+1:),'/')
IND = INDEX(SUB(NSUB),'') + 1
SUB(NSUB)(IND:) = LINE(NSTART:NEND)
IF (NEND .EQ. ILEN) RETURN
NSTART = NEND + IFIRCH(LINE(NEND+1:))
GO TO 10
100 CONTINUE
RETURN
END

SUBROUTINE IPNPAR (LINE, NPAR, IPAR, ISTART)
C
C Returns CHAR*(*) IPAR substring of CHAR*(*) string LINE which
C contains NPAR real parameters
C
C Input: LINE - a CHAR(*) line
C NPAR - number of parameters expected
C Output: IPAR - the substring of parameters only
C A '!' will comment out a line, or remainder of the line.
C
F. Rupley, Div. 8245, 5/14/86
-------------------------------------------------------------
C*****precision > double
IMPLICIT DOUBLE PRECISION (A-H,0-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
CHARACTER*(*) LINE,IPAR
-----------Find Comment String ( !  signifies comment)
ILEN = IPPLEN(LINE)
ISTART = 0
N = 0
IF (ILEN.GT.0) THEN
  DO 40 I = ILEN, 1, -1
    ISTART = I
    IPAR = '  '
    IPAR = LINE(ISTART:ILEN)
    IF (LINE(I:I).NE.'  ') THEN
      IF (I .EQ. 1) RETURN
      IF (LINE(I-1:I-1) .EQ. '  ' ) THEN
        N = N + 1
      ENDIF
    ENDIF
 40 CONTINUE
ENDIF
RETURN
END

SUBROUTINE IPPARI(STRING, ICARD, NEXPEC, IVAL, NFOUND, IERR, LOUT)
C BEGIN PROLOGUE IPPARI
C REFER TO IPGETI
C DATE WRITTEN 850625 (YYMMDD)
C REVISION DATE 851725 (YYMMDD)
C KEYWORDS PARSE
C AUTHOR CLARK,G.L., GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Parses integer variables from a character variable. Called
C by IPGETI, the IOPAK routine used for interactive input.
C DESCRIPTION
C
IPPARI may be used for parsing an input record that contains integer
values, but was read into a character variable instead of directly into integer variables.

The following benefits are gained by this approach:
- specification of only certain elements of the array is allowed,
- thus letting the others retain default values
- variable numbers of values may be input in a record, up to a specified maximum
- control remains with the calling program in case of an input error
- diagnostics may be printed by IPPARI to indicate the nature of input errors

The contents of STRING on input indicate which elements of IVAL are to be changed from their entry values, and values to which they should be changed on exit. Commas and blanks serve as delimiters, but multiple blanks are treated as a single delimiter.

Thus, an input record such as:
' 1, 2, ,40000, ,60'
is interpreted as the following set of instructions by IPGETR:

(1) set IVAL(1) = 1
(2) set IVAL(2) = 2
(3) leave IVAL(3) unchanged
(4) set IVAL(4) = 40000
(5) leave IVAL(5) unchanged
(6) set IVAL(6) = 60

IPPARI will print diagnostics on the default output device, if desired.

IPPARI is part of IOPAK, and is written in ANSI FORTRAN 77.

Examples:
Assume IVAL = (0, 0, 0) and NEXPEC = 3 on entry:

input string  IVAL on exit  IERR  NFOUND
--------------  ---------------  ----  ----
' 2 , 3 45 '    (2, 3, 45)      0   3
'2.15,,3'       (2, 0, 3)      1   0
'3X, 25, 2'     (0, 0, 0)      1   0
'10000'         (10000, 0, 0)  2   1

Assume IVAL = (0, 0, 0, 0) and NEXPEC = -4 on entry:

input string  IVAL on exit  IERR  NFOUND
--------------  ---------------  ----  ----
' 1, 2 '       (1, 2)         0   2
',,37 400 '     (0, 0, 37, 400) 0   4
' 1,, -3, 5 '  (1, 0, -3, 0)  3   4

arguments: (I=input, O=output)

STRING (I) - the character string to be parsed.

ICARD (I) - data statement number, and error processing flag
< 0 : no error messages printed
= 0 : print error messages, but not ICARD
> 0 : print error messages, and ICARD

NEXPEC (I) - number of real variables expected to be input. If < 0, the number is unknown, and any number of values between 0 and abs(nexpec) may be input. (see NFOUND)

PROMPT (I) - prompting string, character type. A question mark will be added to form the prompt at the screen.
IVAL (I,0) - the integer value or values to be modified. On entry, the values are printed as defaults. The formal parameter corresponding to IVAL must be dimensioned at least NEXPEC in the calling program if NEXPEC > 1.

NFOUND (0) - the number of real values represented in STRING, only in the case that there were as many or less than NEXPEC.

IERR (0) - error flag:
  = 0 if no errors found
  = 1 for syntax errors or illegal values found
  = 2 for too few values found (NFOUND < NEXPEC)
  = 3 for too many values found (NFOUND > NEXPEC)

-----------------------------------------------

REFERENCES (NONE)
ROUTINES CALLED IFIRCH,ILASCH
END PROLOGUE IPPARI
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
CHARACTER STRING(*), ITEMP*80
DIMENSION IVAL(*)
CHARACTER *8 FMT(14)
LOGICAL OKINCR
C
FIRST EXECUTABLE STATEMENT IPPARI
IERR = 0
NFOUND = 0
NEXP = IABS(NEXPEC)
IE = ILASCH(STRING)
IF (IE .EQ. 0) GO TO 500
NC = 1
C
C--- OKINCR is a flag that indicates it's OK to increment
C--- NFOUND, the index of the array into which the value
C--- should be read. It is set false when a space follows
C--- an integer value substring, to keep incrementing from
C--- occurring if a comma should be encountered before the
C--- next value.
C
OKINCR = .TRUE.
C
C--- begin overall loop on characters in string
C
100 CONTINUE
C
IF (STRING(NC:NC) .EQ. ',') THEN
  IF (OKINCR .OR. NC .EQ. IE) THEN
    NFOUND = NFOUND + 1
  ELSE
    OKINCR = .TRUE.
  ENDIF
GO TO 450
ENDIF
IF (STRING(NC:NC) .EQ. ' ') GO TO 450
C
C--- first good character (non-delimiter) found - now find
C   last good character
C
   IBS = NC
   CONTINUE
   NC = NC + 1
   IF (NC .GT. IE) GO TO 180
   IF (STRING(NC:NC) .EQ. ',') THEN
      OKINCR = .FALSE.
   ELSEIF (STRING(NC:NC) .EQ. ',') THEN
      OKINCR = .TRUE.
   ELSE
      GO TO 160
   ENDIF
C
C--- end of substring found - read value into integer array
C
   NFOUND = NFOUND + 1
   IF (NFOUND .GT. NEXP) THEN
      IERR ■ =  3
      GO TO 500
   ENDIF
C
   IES = NC - 1
   NCH = IES - IBS + 1
   DATA FMT/' (II)', '  (12)', '  (13)', *  (14)', '  (15)',
   1 '  (16)', '  (17)', '  (18)', '  (19)', '  (110)',
   2 '  (111)', '  (112)', '  (113)', '  (114)','
   ITEMP = ';
   ITEMP = STRING(IBS:IES)
   READ (ITEMP (l.-NCH) ,  FMT (NCH) ,  ERR = 400) IVAL(NFOUND)
   GO TO 450
   400 CONTINUE
   IERR = 1
   GO TO 510
   450 CONTINUE
   NC = NC + 1
   IF (NC .LE. IE) GO TO 100
C
   500 CONTINUE
   IF (NEXPEC .GT. 0 .AND. NFOUND .LT. NEXP) IERR = 2
   510 CONTINUE
C
   IF (IERR .EQ. 0 .OR. ICARD .LT. 0)RETURN
   IF (ICARD .NE. 0) WRITE (LOUT, '(A,13)')
   1 ' !! ERROR IN DATA STATEMENT NUMBER', ICARD
   IF (IERR .EQ. 1)
   1 WRITE (LOUT, ' (A)')'SYNTAX ERROR, OR ILLEGAL VALUE'
   IF (IERR .EQ. 2) WRITE (LOUT, ' (A,I2, A, I2)')
   1 ' TOO FEW DATA ITEMS. NUMBER FOUND = ', NFOUND,
   2 ' NUMBER EXPECTED = ', NEXPEC
   IF (IERR .EQ. 3) WRITE (LOUT, ' (A,I2)')
   1 ' TOO MANY DATA ITEMS. NUMBER EXPECTED = ', NEXPEC
   END
C
SUBROUTINE IPPARR(STRING, ICARD, NEXPEC, RVAL, NFOUND, IERR, LOUT)
C BEGIN PROLOGUE IPPARR
C REFER TO IPGETR
C DATE WRITTEN 850625 (YYMMDD)
C REVISION DATE 851625 (YYMMDD)
C CATEGORY NO. J3.,J4.,M2.
C KEYWORDS PARSE
C AUTHOR CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Parses real variables from a character variable. Called
C by IPGETR, the IOPAK routine used for interactive input.

DESCRIPTION

IPPARR may be used for parsing an input record that contains real values, but was read into a character variable instead of directly into real variables.

The following benefits are gained by this approach:

- specification of only certain elements of the array is allowed, thus letting the others retain default values
- variable numbers of values may be input in a record, up to a specified maximum
- control remains with the calling program in case of an input error
- diagnostics may be printed by IPPARR to indicate the nature of input errors

The contents of STRING on input indicate which elements of RVAL are to be changed from their entry values, and values to which they should be changed on exit. Commas and blanks serve as delimiters, but multiple blanks are treated as a single delimiter.

Thus, an input record such as:

'1., 2., 4.e-5, 6.e-6'

is interpreted as the following set of instructions by IPGETR:

(1) set RVAL(1) = 1.0
(2) set RVAL(2) = 2.0
(3) leave RVAL(3) unchanged
(4) set RVAL(4) = 4.0E-05
(5) leave RVAL(5) unchanged
(6) set RVAL(6) = 6.0E-06

IPPARR will print diagnostics on the default output device, if desired.

IPPARR is part of IOPAK, and is written in ANSI FORTRAN 77.

Examples:

Assume RVAL = (0., 0., 0.) and NEXPEC = 3 on entry:

<table>
<thead>
<tr>
<th>input string</th>
<th>RVAL on exit</th>
<th>IERR</th>
<th>NFOUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>'2.34e-3, 3 45.1'</td>
<td>(2.34E-03, 3.0, 45.1)</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>'2., 3.-5'</td>
<td>(2.0, 0.0, 3.0E-05)</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>',1.4,0.028E4'</td>
<td>(0.0, 1.4, 280.0)</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>'1.0, 2.4, 3.0'</td>
<td>(1.0, 0.0, 0.0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>'1.,'</td>
<td>(1.0, 0.0, 0.0)</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Assume RVAL = (0., 0., 0., 0.) and NEXPEC = -4 on entry:

<table>
<thead>
<tr>
<th>input string</th>
<th>RVAL on exit</th>
<th>IERR</th>
<th>NFOUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>'1.,2.'</td>
<td>(1.0, 2.0)</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>',3 4.0'</td>
<td>(0.0, 0.0, 3.0, 4.0)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>'1,,3,,5.0'</td>
<td>(0.0, 0.0, 3.0, 0.0)</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

arguments: (I=input,O=output)

STRING (I) - the character string to be parsed.

ICARD (I) - data statement number, and error processing flag
< 0 : no error messages printed
  = 0 : print error messages, but not ICARD
> 0 : print error messages, and ICARD

NEXPEC (I) - number of real variables expected to be input. If
< 0, the number is unknown, and any number of values
between 0 and abs(nexpec) may be input. (see NFOUND)

PROMPT (I) - prompting string, character type. A question
mark will be added to form the prompt at the screen.

RVAL (I,0) - the real value or values to be modified. On entry,
the values are printed as defaults. The formal parameter
corresponding to RVAL must be dimensioned at least NEXPEC
in the calling program if NEXPEC > 1.

NFOUND (0) - the number of real values represented in STRING,
only in the case that there were as many or less than
NEXPEC.

IERR (O) - error flag:
= 0 if no errors found
= 2 for too few values found (NFOUND < NEXPEC)
= 3 for too many values found (NFOUND > NEXPEC)

OKINCR is a flag that indicates it's OK to increment
NFIND, the index of the array into which the value
should be read. It is set negative when a space follows
a real value substring, to keep incrementing from
occurring if a comma should be encountered before the
next value.

OKINCR = .TRUE.

begin overall loop on characters in string
CONTINUE
IF (STRING(NC:NC) .EQ. ',') THEN
IF (OKINCR) THEN
NFIND = NFIND + 1
ELSE
OKINCR = .TRUE.
ENDIF

GO TO 450
ENDIF
IF (STRING(NC:NC) .EQ. ' ') GO TO 450
C
C--- first good character (non-delimeter) found - now find
C--- last good character
C
IBS = NC
CONTINUE
160 NC = NC + 1
IF (NC .GT. IE) GO TO 180
IF (STRING(NC:NC) .EQ. ' ') THEN
OKINCR = .FALSE.
ELSEIF (STRING(NC:NC) .EQ. ',') THEN
OKINCR = .TRUE.
ELSE
GO TO 160
ENDIF
C
C  end of substring found - read value into real array
C
180 CONTINUE
NFOUND = NFOUND + 1
IF (NFOUND .GT. NEXP) THEN
IERR = 3
GO TO 500
ENDIF
C
DATA FMT/ ' (E1.0)', ' (E2.0)', ' (E3.0)', ' (E4.0)',
1 ' (E5.0)', ' (E6.0)', ' (E7.0)', ' (E8.0)', ' (E9.0)',
2 ' (E10.0)', ' (E11.0)', ' (E12.0)', ' (E13.0)', ' (E14.0)',
3 ' (E15.0)', ' (E16.0)', ' (E17.0)', ' (E18.0)', ' (E19.0)',
4 ' (E20.0)', ' (E21.0)', ' (E22.0)/
IES = NC - 1
NCH = IES - IBS + 1
ITEMP = ' '
ITEMP = STRING(IBS:IES)
READ (ITEMP(1:NCH), FMT(NCH), ERR = 400) RVAL(NFOUND)
GO TO 450
400 CONTINUE
WRITE (LOUT, 555) STRING(IBS:IES)
555 FORMAT (A)
IERR = 1
GO TO 510
450 CONTINUE
NC = NC + 1
IF (NC .LE. IE) GO TO 100
C
500 CONTINUE
IF (NEXPEC .GT. 0 .AND. NFOUND .LT. NEXP) IERR = 2
510 CONTINUE
C
IF (IERR .EQ. 0 .OR. ICARD .LT. 0) RETURN
IF (ICARD .NE. 0) WRITE (LOUT, '(A,I3)')
1 '!! ERROR IN DATA STATEMENT NUMBER', ICARD
IF (IERR .EQ. 1)
1 WRITE (LOUT, '(A)')'SYNTAX ERROR, OR ILLEGAL VALUE'
IF (IERR .EQ. 2) WRITE (LOUT, '(A,I2, A, I2)')
1 ' TOO FEW DATA ITEMS. NUMBER FOUND = ', NFOUND,
2 ' NUMBER EXPECTED = ', NEXPEC
IF (IERR .EQ. 3) WRITE (LOUT, '(A,I2)')
1 ' TOO MANY DATA ITEMS. NUMBER EXPECTED = ', NEXPEC
END
C
FUNCTION IFIRCH(STRING)
C BEGIN PROLOGUE IFIRCH
IFIRCH locates the first non-blank character in a string of arbitrary length. If no characters are found, IFIRCH is set = 0. When used with the companion routine ILASCH, the length of a string can be determined, and/or a concatenated substring containing the significant characters produced.

REFERENCES (NONE)
ROUTINES CALLED (NONE)
END PROLOGUE IFIRCH

**precision > double**
IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
END precision > double

**precision > single**
IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
END precision > single

CHARACTER* (*)STRING

FIRST EXECUTABLE STATEMENT IFIRCH
NLOOP = LEN(STRING)

IF (NLOOP .EQ. 0) THEN
  IFIRCH = 0
  RETURN
ENDIF

DO 100 I = 1, NLOOP
  IF (STRING(I) .NE. ' ') GO TO 120
100 CONTINUE

IFIRCH = 0
RETURN

FUNCTION ILASCH(STRING)

BEGIN PROLOGUE ILASCH
DATE WRITTEN 850626
REVISION DATE 850626
CATEGORY NO. M4.
KEYWORDS CHARACTER STRINGS, SIGNIFICANT CHARACTERS
AUTHOR CLARK, G.L., GROUP C-3 LOS ALAMOS NAT'L LAB
PURPOSE Determines last significant (non-blank) character in character variable
DESCRIPTION

IFIRCH locates the last non-blank character in a string of arbitrary length. If no characters are found, ILASCH is set = 0. When used with the companion routine IFIRCH, the length of a string can be determined, and/or a concatenated substring containing the significant characters produced. Note that the FORTRAN intrinsic function LEN returns the length of a character string as declared, rather than as filled. The
C declared length includes leading and trailing blanks, and thus is
C not useful in generating 'significant' substrings.
C-----------------------------------------------
C REFERENCES (NONE)
C ROUTINES CALLED (NONE)
C END PROLOGUE IFIRCH
C****precision > double
C IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
C****END precision > double
C
C****precision > single
C IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
C****END precision > single
C
CHARACTER*(*) STRING
C
C***FIRST EXECUTABLE STATEMENT ILASCH
NLOOP = LEN(STRING)
   IF (NLOOP.EQ.0) THEN
      ILASCH = 0
      RETURN
   ENDIF
C
DO 100 I = NLOOP, 1, -1
   IF (STRING(1:1) .NE. ' ') CONTINUE
   CONTINUE
   ILASCH = I
END
C-----------------------------------------------
C
SUBROUTINE CKCOMP (1ST, IRAY, II, I)
C
C START PROLOGUE
C
SUBROUTINE CKCOMP (1ST, IRAY, II, I)*
C Returns the index of an element of a reference character
C string array which corresponds to a character string;
C leading and trailing blanks are ignored.
C
C INPUT
C   IST  - A character string.
C      Data type - CHARACTER*(*)
C   IRAY - An array of character strings;
C            dimension IRAY(*) at least II
C      Data type - CHARACTER*(*)
C   II   - The length of IRAY.
C      Data type - integer scalar.
C
C OUTPUT
C   I    - The first integer location in IRAY in which IST
C            corresponds to IRAY(I); if IST is not also an
C            entry in IRAY, I=0.
C
C END PROLOGUE
C
C****precision > double
C 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
CHARACTER*(*) IST, IRAY(*)

I = 0
DO 10 N = 11, 1, -1
IS1 = IIFIRCH(IST)
IS2 = ILASCH(IST)
IR1 = IIFIRCH(IRAY(N))
IR2 = ILASCH(IRAY(N))
IF (IS2.GE.IS1 .AND. IS2.GT.0 .AND. IR2.GE.IRI .AND. IR2.GT.0 .AND. IST(IS1:IS2).EQ.IRAY(N)(IR1:IR2) ) I=N
10 CONTINUE
RETURN
END

SUBROUTINE CKUNIT (LINE, AUNITS, EUNITS, IUNITS)
*****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
CHARACTER*4 LINE, AUNITS, EUNITS, IUNITS
CHARACTER*4 UPCASE

AUNITS = '  '
EUNITS = '  '
IUNITS = '  '
DO 85 N = 1, ILASCH(LINE)-3
IND = ILASCH(IUNITS)
IF (EUNITS .EQ. '  ' ) THEN
  IF (UPCASE(LINE(N:), 4) .EQ. 'CAL/' ) THEN
    EUNITS = 'CAL/
  ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KCAL' ) THEN
    EUNITS = 'KCAL
  ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'JOUL' ) THEN
    EUNITS = 'JOUL
  ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KJOU' ) THEN
    EUNITS = 'KJOU
  ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KELV' ) THEN
    EUNITS = 'KELV
  ELSE
    EUNITS = '  '
ENDIF
ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'CAL' ) THEN
  IF (IUNITS .EQ. '  ' ) THEN
    IUNIT(IND:)) = ' , E units cal/mole
  ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KCAL' ) THEN
    IUNIT = 'E units Kcal/mole
  ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'JOUL' ) THEN
    IUNITS = 'E units Joules/mole
  ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KJOU' ) THEN
    IUNITS = 'E units Kjoule/mole
  ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KELV' ) THEN
    IUNITS = 'E units Kelvins
  ELSE
IUNITS(IND:) = ', E units Kelvins'
ENDIF
ENDIF
ENDIF
IF (AUNITS .EQ. ' ') THEN
  IF (UPCASE(LINE(N+1:), 4) .EQ. 'MOLE') THEN
    AUNITS = 'MOLE'
    IF (IUNITS .EQ. ' ') THEN
      IUNITS = 'A units mole-cm-sec-K'
    ELSE
      IUNITS(IND:) = ' , A units mole-cm-sec-K'
    ENDIF
  ELSEIF (UPCASE(LINE(N+4:), 4) .EQ. 'CULE') THEN
    AUNITS = 'MOLC'
    IF (IUNITS .EQ. ' ') THEN
      IUNITS = 'A units molecules'
    ELSE
      IUNITS(IND:) = ' , A units molecules'
    ENDIF
  ELSE
    IUNITS(IND:) = ', A units mole-sec-K'
  ENDIF
ENDIF
ENDIF
ENDIF
C
IF (AUNITS .EQ. ' ') THEN
  AUNITS = 'MOLE'
  IND = ILASCH(IUNITS) + 1
  IF (IND .GT. 1) THEN
    IUNITS(IND:) = ' , A units mole-cm-sec-K'
  ELSE
    IUNITS(IND:) = ' A units mole-cm-sec-K'
  ENDIF
ENDIF
C
EUNITS = 'CAL/
  IND = ILASCH(EUNITS) + 1
  IF (IND .GT. 1) THEN
    IUNITS(IND:) = ' , E units cal/mole'
  ELSE
    IUNITS(IND:) = ' E units cal/mole'
  ENDIF
ENDIF
C
RETURN
END
C
INTEGER FUNCTION IPPLEN (LINE)
C
BEGIN PROLOGUE
C
FUNCTION IPPLEN (LINE)
C Returns the effective length of a character string, i.e.,
C the index of the last character before an exclamation mark (!) indicating a comment.
C
INPUT
C LINE - A character string.
C
OUTPUT
C IPPLEN - The effective length of the character string.
C
END PROLOGUE
C
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
CHARACTER LINE(*)
C
IN = IFIRCH(LINE)
IF (IN.EQ.0 .OR. LINE(IN:IN).EQ.'!') THEN
  IPPELEN = 0
ELSE
  IN = INDEX(LINE,'!')
  IF (IN.EQ.0) THEN
    IPPELEN = ILASCH(LINE)
  ELSE
    IPPELEN = ILASCH(LINE(:IN-1))
  ENDIF
ENDIF
ENDIF
RETURN
END
C
CHARACTER*(*) FUNCTION UPCASE(ISTR, ILEN)
CHARACTER ISTR(*), LCASE(26)*1, UCASE(26)*1
DATA LCASE /'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
1 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'/,
C
UPCASE = ISTR
UPCASE = ISTR(:ILEN)
JJ = MIN (LEN(UPCASE), LEN(ISTR), ILEN)
DO 10 J = 1, JJ
  DO 10 N = 1, 26
    IF (ISTR(J:J) .EQ. LCASE(N)) UPCASE(J:J) = UCASE(N)
  10 CONTINUE
END
SUBROUTINE FILEIO_INT

IMPLICIT DOUBLE PRECISION (A-H,O-Z)
PARAMETER (LIN=5, LOUT=6, LTHRM=17, LINC=25)
CHARACTER FILE_MECH*20, FILE_THM*20, FILE_INP*20,
1 FILE_MOUT*20, FILE_OUT*20, FILE_LINK*20
COMMON/FILE_IO/FILE_MECH, FILE_THM, FILE_INP,
1 FILE_MOUT, FILE_OUT, FILE_LINK

5 WRITE(*,10)
10 FORMAT('Please enter the input mechanism file name : ')
READ(*,'(A20)')FILE_MECH
OPEN(UNIT=5,FILE=FILE_MECH,STATUS='OLD', ERR=30)
GO TO 50
30 CONTINUE
WRITE(*,40)FILE_MECH
40 FORMAT('Can not find file : ',A20)
GO TO 5
50 CONTINUE
105 WRITE(*,110)
110 FORMAT(' Please enter the NASA thermo file name : ') READ(*,'(A20)')FILE_THM
OPEN(UNIT=17,FILE=FILE_THM,STATUS='OLD',ERR=130)
  GO TO 150
130 CONTINUE
  WRITE(*,140)FILE_THM
140 FORMAT(' Can not find file : ',A20)
  GO TO 105
150 CONTINUE
  OPEN(UNIT=6,FILE='CKINTERP.OUT',STATUS='UNKNOWN')
  OPEN(UNIT=25,FILE='CHEMKIN.LNK',STATUS='UNKNOWN',
  FORM='BINARY')
  RETURN
END
D.5 CKLIB4

SUBROUTINE CKRAT (RCKWRK, ICKWRK, II, KK, MAXSP, MAXTB, RU, PATM, 
C
T, C, NSPEC, NU, NUNK, NPAR, PAR, NREV, IREV, 
RPAR, NPAR, IFAL, IFOP, KFAL, NFAR, FPAR, NLAN, 
NLAR, ILAN, PLT, NRLT, IRLT, RPLT, NTHB, ITHB, 
NTBS, AIK, NKTB, SMH, RKF, RKR, EQK, CTB)

START PROLOGUE

SUBROUTINE CKRAT (RCKWRK, ICKWRK, II, KK, MAXSP, MAXTB, RU, PATM, 
T, C, NSPEC, NU, NUNK, NPAR, PAR, NREV, IREV, 
RPAR, NPAR, IFAL, IFOP, KFAL, NFAR, FPAR, NLAN, 
NLAR, ILAN, PLT, NRLT, IRLT, RPLT, NTHB, ITHB, 
NTBS, AIK, NKTB, SMH, RKF, RKR, EQK, CTB)

END PROLOGUE

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
DIMENSION RCKWRK(*), ICKWRK(*), C(*), NSPEC(*), NU(MAXSP,*), 
NUNK(MAXSP,*), PAR(NPAR,*), IREV(*), RPAR(NPAR,*), 
ILAN(*), IRLT(*), PLT(NLAR,*), RPLT(NLAR,*), 
IFAL(*), IFOP(*), KFAL(*), FPAR(NFAR,*), ITHB(*), 
NTBS(*), AIK(MAXTB,*), NKTB(MAXTB,*), SMH(*), 
RKF(*), RKR(*), EQK(*), CTB(*)

COMMON /MACH/ SMALL,BIG,EXPARG

C
WRITE(*,*)'...CALL...CKRAT.....'
ALOGT = LOG(T)
C
DO 20 I = 1, II
CTB(I) = 1.0
RKF(I) = PAR(1,I) * EXP(PAR(2,I)*ALOGT - PAR(3,I)/T)
20 CONTINUE
C
Landau-Teller reactions
C
DO 25 N = 1, NLAN
I = ILAN(N)
TFAC = PLT(1,N)/T**(1.0/3.0) + PLT(2,N) / T**2(0.0/3.0)
RKF(I) = RKF(I) * EXP(TFAC)
25 CONTINUE
C
CALL CKSMH (T, ICKWRK, RCKWRK, SMH)
DO 50 I = 1, II
SUMSMH = 0.0
DO 40 N = 1, MAXSP
IF (NUNK(N,I).NE.0) SUMSMH=SUMSMH+NU(N,I)*SMH(NUNK(N,I))
40 CONTINUE
C
EQK(I) = EXP(MIN(SUMSMH,EXPARG))
50 CONTINUE

355
PFAC = PATH / (RU*T)
DO 60 I = 1, II
   NUSUMK = NU(1,1)+NU(2,1)+NU(3,1)+NU(4,1)+NU(5,1)+NU(6,1)
   EQK(I) = EQK(I) * PFAC**NUSUMK
END

RKR=0.0 for irreversible reactions, else RKR=RKF/\text{MAX}(EQK, SMALL)

IF (NNSPEC(I).GT.0) RKR(I) = RKF(I) / \text{MAX}(EQK(I), SMALL)
60 CONTINUE

if reverse parameters have been given:

DO 70 N = 1, NREV
   I = IREV(N)
   RKR(I) = RPAR(1,N) * EXP(RPAR(2,N)*ALOGT - RPAR(3,N)/T)
   EQK(I) = RKF(I)/RKR(I)
70 CONTINUE

if reverse Landau-Teller parameters have been given:

DO 75 N = 1, NRLT
   I = IRLT(N)
   TFAC = RPLT(1,N)/T**(1.0/3.0) + RPLT(2,N)/T**(2.0/3.0)
   RKR(I) = RKR(I) * EXP(TFAC)
   EQK(I) = RKF(I)/RKR(I)
75 CONTINUE

C third-body reactions

CTOT = 0.0
DO 10 K = 1, KK
   CTOT = CTOT + C(K)
10 CONTINUE

DO 80 N = 1, NTHB
   C_TB(ITHB(N)) = CTOT
   DO 80 L = 1, NTBS(N)
      C_TB(ITHB(N)) = C_TB(ITHB(N)) + (AIK(L, N)-1.0)*C(NKTB(L,N))
80 CONTINUE

C If fall-off (pressure dependence):

WRITE(*,*)'....NFAL = ',NFAL
DO 90 N = 1, NFAL
   WRITE(*,*)'.... IFOP('',N,'') = ',IFOP(N)
90 CONTINUE

CONCENTRATION OF THIRD BODY

IF (KFAL(N) .EQ. 0) THEN
   C_TBH = C_TB(IFAL(N))
   C_TB(IFAL(N)) = 1.0
ELSE
   C_TBH = C(KFAL(N))
ENDIF

RKL0W = FPAR(1,N) * EXP(FPAR(2,N)*ALOGT - FPAR(3,N)/T)
PR = RKL0W*C_TBH / RKF(IFAL(N))
PRLOG = LOG10(MAX(PR, SMALL))

IF (IFOP(N) .EQ. 1) THEN
   LINDEMANN FORM
   FC = 1.0
ELSE
  IF (IFOP(N) .EQ. 2) THEN
    SRI FORM
    XP = 1.0/(1.0 + PRLOG**2)
    FC = ((FPAR(4,N)**EXP(-FPAR(5,N)/T) + EXP(-T/FPAR(6,N)))
    **XP) * FPAR(7,N) * T**FPAR(8,N)
  ENDIF
ELSE
  6-PARAMETER TROE FORM
  FCENT = (1.0 - FPAR(4,N)) * EXP(-T/FPAR(5,N))
  7-PARAMETER TROE FORM
  IF (IFOP(N) .EQ. 4) FCENT = FCENT + EXP(-T/FPAR(6,N))
  8-PARAMETER EXT-TROE FORM
  IF (IFOP(N) .EQ. 5) FCENT = FCENT + EXP(-T/FPAR(7,N))
IF (IFOP(N) .EQ. 5) PR = PR * (CTHB)**(FPAR(8,N)-1.0)
ELSE
  W.Ing 08/01/95
  Now we defined
  Tmin = 300. (K)
  Tmax = 2500. (K)
  Pmin = 1.0E-03 (ATM)
  Pmax = 1.0E+02 (ATM)
  **************************************************************************
  * which can be redefined but make sure Cheby's polynomials *
  * refitting with corresponding T & P range. IMPORTANT !!! *
  **************************************************************************
  Tmin = 300.
  Tmax = 2500.
  Pmin = 1.0E-03
  Pmax = 1.0E+02
  T_cheb = (2.*(l./T)-(1/Tmin)-(1/Tmax))/
    ((1/Tmax)-(1/Tmin))
  P_cheb = (2.*dlog10(PATM)-dlog10(Pmin)-
    2*dlog10(Pmax)) / (dlog10(Pmax)-dlog10(Pmin))
  RK_tempLog = 0.0
  DO 82 I = 1, FPAR(4,N)
    DO 81 J = 1, FPAR(4,5)
      write(*,*) RK_tempLog, ((I-1)*M_cheb+J), X ((I-1)*M_cheb+J),
      Phich(I,T_cheb),Phich(I,P_cheb)
    81    CONTINUE
  82    CONTINUE
  FC = 10.0**RK_tempLog
ELSE
  FCLOG = LOG10(MAX(FCENT,SMALL))
  XN = 0.75 - 1.27*FCLOG
  CPRLOG = PRLOG - (0.4 + 0.67*FCLOG)
  FLOG = FCLOG/(1.0 + (CPRLOG/(XN-0.14*CPRLOG))**2)
  FC = 10.0**FLOG
ENDIF
ENDIF
ENDIF
IF (IFOP(N) .EQ. 6) THEN
RKF(IFAL(N)) = FC
RKR(IFAL(N)) = RKF(IFAL(N)) * FC
ELSE
PCOR = FC * PR/(1.0+PR)
RKF(IFAL(N)) = RKF(IFAL(N)) * PCOR
RKR(IFAL(N)) = RKR(IFAL(N)) * PCOR
ENDIF
WRITE(*,*)'..IFOP = ',IFOP(N),' RKF(',N,' ) = ',RKF(N)
WRITE(*,*)'..IFOP = ',IFOP(N),' RKR(',N,' ) = ',RKR(N)
90 CONTINUE

C Multiply by the product of reactants and product of products
C PAR(4,I) is a perturbation factor
C
DO 150 I = 1, II
RKF(I) = RKF(I)*CTB(I)*C(NUNK(1,I))**IABS(NU(1,I))*PAR(4,I)
RKR(I) = RKR(I)*CTB(I)*C(NUNK(4,I))**NU(4,I)*PAR(4,I)
IF (NUNK(2,I) .NE. 0) THEN
RKF(I) = RKF(I) * C(NUNK(2,I))**IABS(NU(2,I))
IF (NUNK(3,I) .NE. 0)
1 RKF(I) = RKF(I) * C(NUNK(3,I))**IABS(NU(3,I))
ENDIF
IF (NUNK(5,I) .NE. 0) THEN
RKR(I) = RKR(I) * C(NUNK(5,I))**NU(5,I)
IF (NUNK(6,I) .NE. 0) RKR(I) = RKR(I) * C(NUNK(6,I))**NU(6,I)
ENDIF
150 CONTINUE
C RETURN
END

C---------------------------------------------------------------C
C SUBROUTINE CKRATT (RCKWRK, ICKWRK, II, MAXSP, RU, PATM, T, NSPEC,
C 1 NU, NUNK, NPAR, PAR, NREV, IREV, RPAR, NLAN,
C 2 NLAR, ILAN, PLT, NRLT, IRLT, RPLT, SMH, RKFT,
C 3 RKRT, EQK)
C
C START PROLOGUE
C
C SUBROUTINE CKRATT (RCKWRK, ICKWRK, II, MAXSP, RU, PATM, T, NSPEC,
C 1 NU, NUNK, NPAR, PAR, NREV, IREV, RPAR, NLAN,
C 2 NLAR, ILAN, PLT, NRLT, IRLT, RPLT, SMH, RKFT,
C 3 RKRT, EQK)
C
C END PROLOGUE
C
C*****precision > double
C 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
DIMENSION RCKWRK(*), ICKWRK(*), NSPEC(*), NU(MAXSP,*),
1 NPAR(MAXSP,*), PAR(NPAR,*), IREV(*), RPAR(NPAR,*),
2 ILAN(*), IRLT(*), PLT(NLAR,*), RPLT(NLAR,*), SMH(*),
3 RKFT(*), RKRT(*), EQK(*)
C
COMMON /MACH/ SMALL,BIG,EXPARG
ALOGT = LOG(T)

DO 20 I = 1, II
   RKFT(I) = PAR(1,I) * EXP(PAR(2,I)*ALOGT - PAR(3,I)/T)
20 CONTINUE

Landau-Teller reactions

DO 25 N = 1, NLAN
   I = ILAN(N)
   TFAC = PLT(1,N)/T**(1.0/3.0) + PLT(2,N)/T**(2.0/3.0)
   RKFT(I) = RKFT(I) * EXP(TFAC)
25 CONTINUE

CALL CKSMH (T, ICKWRK, RCKWRK, SMH)
DO 50 I = 1, II
   SUMSMH = 0.0
   DO 40 N = 1, MAXSP
      IF (NU(N,I).NE.0) SUMSMH=SUMSMH+NU(N,I)*SMH(NU(N,I))
40 CONTINUE
   EQK(I) = EXP(MIN(SUMSMH,EXPARG))
50 CONTINUE

PFAC = PATM / (RU*T)
DO 60 I = 1, II
   NUSUMK = NU(1,I)+NU(2,I)+NU(3,I)+NU(4,I)+NU(5,I)+NU(6,I)
   EQK(I) = EQK(I) * PFAC**NUSUMK
60 CONTINUE

RKRT=0.0 for irreversible reactions, else RKRT=RKFT/MAX(EQK,SMALL)

IF (NSPEC(I).GT.0) RKRT(I) = RKFT(I) / MAX(EQK(I),SMALL)

if reverse parameters have been given:

DO 70 N = 1, NREV
   I = IREV(N)
   RKRT(I) = RPAR(1,N) * EXP(RPAR(2,N)*ALOGT - RPAR(3,N)/T)
   EQK(I) = EQK(I) / RKRT(I)
70 CONTINUE

if reverse Landau-Teller parameters have been given:

DO 75 N = 1, NRLT
   I = IRLT(N)
   TFAC = RPLT(1,N)/T**(1.0/3.0) + RPLT(2,N)/T**(2.0/3.0)
   RKRT(I) = RKRT(I) * EXP(TFAC)
   EQK(I) = RKFT(I) / RKRT(I)
75 CONTINUE

DO 80 I = 1, II
   RKFT(I) = RKFT(I) * PAR(4,I)
   RKRT(I) = RKRT(I) * PAR(4,I)
80 CONTINUE

RETURN
END
START PROLOGUE

SUBROUTINE CKRATX (II, KK, MAXSP, MAXTB, T, C, NSPEC, NU, NUNK, NFAL, IFAL, IFOP, KFAL, NFAR, FPAR, NTHB, ITHB, NTBS, AIK, NKTB, RKFT, RKR, CTB)

END PROLOGUE

*****precision > double
IMPLICIT DOUBLE PRECISION (A-H, O-Z), INTEGER (I-N)
*****END precision > double

*****precision > single
IMPLICIT REAL (A-H, O-Z), INTEGER (I-N)
*****END precision > single

DIMENSION C(*), NSPEC(*), NU(MAXSP,*), NUNK(MAXSP,*),
1 IFAL(*), IFOP(*), KFAL(*), NFAR(*), NTHB(*),
2 NTBS(*), AIK(MAXTB,*), NKTB(MAXTB,*), RKFT(*),
3 RKR(*), RKFT(*), RKR(*), CTB(*)

COMMON /MACH/ SMALL, BIG, EXPARG

DO 20 I = 1, II
  CTB(I) = 1.0
  RKF(I) = RKFT(I)
  RKR(I) = RKR(I)
20 CONTINUE

third-body reactions

CTOT = 0.0
DO 10 K = 1, KK
  CTOT = CTOT + C(K)
10 CONTINUE

DO 80 N = 1, NTHB
  CTB(ITHB(N)) = CTOT
  DO 80 L = 1, NTBS(N)
    CTB(ITHB(N)) = CTB(ITHB(N)) + (AIK(L,N)-1.0)*C(NKTB(L,N))
80 CONTINUE

If fall-off (pressure correction):

ALOGT = LOG(T)

DO 90 N = 1, NFAL

  WRITE(*,*)' IFOP(',N,') = ',IFOP(N)
  RKLOW = FPAR(1,N) * EXP(FPAR(2,N)*ALOGT - FPAR(3,N)/T)
  WRITE(*,*)' RKLOW = ', RKLOW

  CONCENTRATION OF THIRD BODY

  IF (KFAL(N) .EQ. 0) THEN
    PR = RKLOW * CTB(IFAL(N)) / RKF(IFAL(N))
    CTB(IFAL(N)) = 1.0
  ELSE
    PR = RKLOW * C(KFAL(N)) / RKF(IFAL(N))
  ENDIF

  PCOR = PR / (1.0 + PR)

  IF (IFOP(N) .GT. 1) THEN
    PRLOG = LOG10(MAX(PR, SMALL))
  ENDIF
IF (IFOP(N) .EQ. 2) THEN

6-PARAMETER SRI FORM

\[
XP = 1.0/(1.0 + PLOG**2)
 FC = ((FPAR(4,N)*EXP(-FPAR(5,N)/T) + EXP(-T/FPAR(6,N)))
   **XP) * FPAR(7,N) * T**FPAR(8,N)
\]

ELSE

6-PARAMETER TROE FORM

WRITE(*,*)' BEFORE EXP'

IF (IFOP(N) .NE. 6) FCENT = (1.0 - FPAR(4,N)) *
 \[ \exp(-T/FPAR(5,N)) + FPAR(4,N) \times \exp(-T/FPAR(6,N)) \]
 WRITE(*,*)' FCENT = ', FCENT

7-PARAMETER TROE FORM

IF (IFOP(N) .EQ. 4) FCENT = FCENT + \exp(-FPAR(7,N)/T)

8-PARAMETER EXT-TROE FORM

IF (IFOP(N) .EQ. 5) FCENT = FCENT + \exp(-FPAR(7,N)/T)
 IF (IFOP(N) .EQ. 5) PR = PR * (CTHB)**(FPAR(8,N)-1.0)

W. ING 07/11/95

8-PARAMETER EXT-TROE FORM

IF (IFOP(N) .EQ. 5) FCENT = FCENT + \exp(-FPAR(7,N)/T)
 IF (IFOP(N) .EQ. 5) PR = PR * (CTHB)**(FPAR(8,N)-1.0)

W. ING 08/01/95

Now we defined

\[
T_{\text{min}} = 300. \text{ (K)}
 T_{\text{max}} = 2500. \text{ (K)}
 P_{\text{min}} = 1.0E-03 \text{ (ATM)}
 P_{\text{max}} = 1.0E+02 \text{ (ATM)}
\]

which can be redefined but make sure Cheby's polynomials
* refitting with corresponding T & P range. IMPORTANT !!! *

\[
T_{\text{cheb}} = \frac{2\times(1/T)-(1/T_{\text{min}})-(1/T_{\text{max}})}{2((1/T_{\text{max}})-(1/T_{\text{min}}))}
 P_{\text{cheb}} = \frac{2d\log_{10}(P)-(d\log_{10}(P_{\text{min}})-d\log_{10}(P_{\text{max}}))}{d\log_{10}(P_{\text{max}})-d\log_{10}(P_{\text{min}})}
\]

WRITE(*,*)' T = ', T, ' T_{\text{cheb}} = ', T_{\text{cheb}}
 WRITE(*,*)' P = ', P, ' P_{\text{cheb}} = ', P_{\text{cheb}}

RK_tempLog = 0.0

DO 82 I = 1, FPAR(4,N)
 DO 81 J = 1, FPAR(5,N)

WRITE(*,*)' RK_tempLog = ', RK_tempLog

81 CONTINUE

WRITE(*,*)' FC = ', FC

ELSE

FCLOG = LOG10(MAX(FCENT, SMALL))
 Xavier N. = 0.75 - 1.27*FCLOG
CPRLOG = PRLOG - (0.4 + 0.67*FCLOG)
FLOG = FCLOG/(1.0 + (CPRLOG/(XN-0.14*CPRLOG))**2)
FC = 10.0**FLOG
ENDIF
ENDIF
ENDIF
IF (IFOP(N) .EQ. 6) THEN
RKF(IFAL(N)) = FC
WRITE(*,*)' RKF( ', IFAL(N), ') = ', RKF(IFAL(N))
RKR(IFAL(N)) = RKR(IFAL(N)) * FC / RKFT(IFAL(N))
ELSE
IF (IFOP(N) .GT. 1) PCOR = FC * PCOR
RKF(IFAL(N)) = RKF(IFAL(N)) * PCOR
RKR(IFAL(N)) = RKR(IFAL(N)) * PCOR
ENDIF
WRITE(*,*)'. .IFOP = ', IFOP(N), ', RKF(', IFAL(N), ') = ', RKF(IFAL(N))
WRITE(*,*)'. .RKR(', IFAL(N), ') = ', RKR(IFAL(N))
20 CONTINUE
FCLOG = LOG10(MAX(FCENT, SMALL))
XN = 0.75 - 1.27*FCLOG
CPRLOG = PRLOG - (0.4 + 0.67*FCLOG)
FLOG = FCLOG/(1.0 + (CPRLOG/(XN-0.14*CPRLOG))**2)
FC = 10.0**FLOG
ENDIF
PCOR = FC * PCOR
ENDIF
RKF(IFAL(N)) = RKF(IFAL(N)) * PCOR
RKR(IFAL(N)) = RKR(IFAL(N)) * PCOR
20 CONTINUE
C
C Multiply by the product of reactants and product of products
DO 150 I = 1, II
RKF(I) = RKF(I)*CTB(I)*C(NUNK(1,I))**IABS(NU(1,I))
RKR(I) = RKR(I)*CTB(I)*C(NUNK(4,I))**NU(4,I)
IF (NUNK(2,I) .NE. 0) THEN
RKF(I) = RKF(I) * C(NUNK(2,I))**IABS(NU(2,I))
IF (NUNK(3,I) .NE. 0) THEN
RKF(I) = RKF(I) * C(NUNK(3,I))**IABS(NU(3,I))
ENDIF
150 CONTINUE
C
RETURN
END
PROGRAM MECHCVT
C
C--------------------------------------------------------------------------C
C***************changed W.Ing 06-15-94, 08-01-95*****************************C
C
C  MDIM = maximum number of elements in a problem;                   (20)
C  KDIM = maximum number of species in a problem;                     (500)
C  MAXTP = maximum number of temperatures used to fit                 (3)
C  NPC = number of polynomial coefficients to fits                    (5)
C  NPCP2= number of fit coefficients for a temperature range          (7)
C  IDIM = maximum number of reactions in a mechanism;                 (1500)
C  NPAR = number of Arrhenius parameters in a reaction;               (3)
C  NLAR = number of Landau-Teller parameters in a reaction;           (2)
C  NFAR = number of fall-off parameters in a reaction;                (66)
C  MAXSP= maximum number of species in a reaction                    (6)
C  MAXTB= maximum number of third bodies for a reaction               (10)
C  LSYM = character string length of element and species names       (16)
C
C  User input is read from LIN (Unit15), a thermodynamic database
C  is read from LTHRM (Unit17), printed output is assigned to LOUT
C  (Unit16), and binary linking data is written to LINC (Unit25).
C
C  REQUIRED ELEMENT INPUT: (Subroutine CKCHAR) (DIMENSION)
C
C  The word 'ELEMENTS' followed by a list of element
C  names, terminated by the word 'END';
C
C  The resulting element data stored in LINK is:
C  MM - integer number of elements found
C  ENAME(*) - CHARACTER(*) array of element names (MDIM)
C  AWT(*) - real array of atomic weights; (MDIM)
C  default atomic weights are those on
C  atomic weight charts; if an element
C  is not on the periodic chart, or if
C  it is desirable to alter its atomic
C  weight, this value must be included
C  after the element name, enclosed by
C  slashed, i.e., D/2.014/
C
C  REQUIRED SPECIES INPUT: (Subroutine CKCHAR)
C
C  The word 'SPECIES' followed by a list of species
C  names, terminated by the word 'END';
C
C  The resulting species data stored in LINK is:
C  KK - integer number of species found
C  KNAME(*) - CHARACTER(*) array of species names (KDIM)
C
C  OPTIONAL THERMODYNAMIC DATA: (Subroutine CKTHRM)
C  (If this feature is not used, thermodynamic properties are
C  obtained from a CHEMKIN database.) The format for this option
C  is the word 'THERMO' followed by any number of 4-line data sets:
C
C  Line 1: species name, optional comments, elemental composition,
C  phase, T(low), T(high), T(mid), additional elemental
C  composition, card number (col. 60);
C  format(A10,A14,4(A2,13),A1,E10.0,E10.0,E8.0,(A2,13),I1)
C  Line 2: coefficients a(1--5) for upper temperature range,
C  card number (col. 80);
C
C  363
format(5(e15.0),11)

Line 3: coefficients a(6-7) for upper temperature range,
coefficients a(1-3) for lower temperature range,
card number (col. 80);
format(5(e15.0),11)

Line 4: coefficients a(4-7) for lower temperature range,
card number (col. 80);
format(4(e15.0),11)

End of THERMO data is indicated by 'END' line or new keyword.

The resulting thermodynamic data stored in LINK are:

WTM(*) - real array of molecular weights (KDIM)

KNCF(*,*) - integer composition of species (MDIM,KDIM)

MPHSE(*) - integer phase of a species; (KDIM)

KCHRG(*) - ion charge of a species; (KDIM)

= 0 except in presence/absence of electrons
= +n in absence of n electrons
= -n in presence of n electrons

NCHRG - integer number of species with KCHRG<>0

NT(*) - array of number of temperatures used (KDIM)
in fits

T(*,*) - array of temperatures used in fits (MAXTP,KDIM)

A(N,L,K) - Thermodynamic properties for (NPC+2,NTR,KDIM)
species K consists of polynomial coefficients for fits to

\[ \frac{C_P}{R} = \text{SUM} \left( A(N,L,K) \times \text{Temperature}^{(N-1)} \right), \quad N=1,NPC+2 \]

where \( T(L,K) \leq \text{Temperature} < T(L+1,K) \),

and,

\[ N=\text{NPC+1} \text{ is formation enthalpy } \frac{H_0}{R} = A(\text{NPC+1},L,K), \]

\[ N=\text{NPC+2} \text{ is formation entropy } \frac{S_0}{R} = A(\text{NPC+2},L,K) \]

OPTIONAL REACTION INPUT:

Reaction data is input after all ELEMENT, SPECIES and THERMO
data in the following format:

1) (Subroutine CKREAC)

The first line contains the keyword 'REACTIONS' and an
optional description of units:

'MOLES' - (default), pre-exponential units are moles-sec-K;

'MOLECULES' - pre-exponential units are molecules and
will be converted to moles.

'KELVINS' - activation energies are Kelvins, else the
activation energies are converted to Kelvins;

'CAL/MOLE' - (default), activation energies are cal/mole;

'KCAL/MOLE' - activation energies are Kcal/mole;

'JOULES/MOLE' - activation energies are joules/mole;

'KJOULES/MOLE' - activation energies are Kjoules/mole.

A description of each reaction is expected to follow.

Required format for a reaction is a list of '+'-delimited reactants, followed by a list of '+'-delimited reactants,
each preceded by its stoichiometric coefficient if greater
than 1; separating the reactants from the products is a '='
if reversible reaction, else a '=>'. Following the reaction
string on the same line are the space-delimited Arrhenius
coefficients.

If the reaction contains a third body, this is indicated by
by the presence of an 'M' as a reactant or product or both,
and enhancement factors for third-bodies may be defined on
additional lines as described in (2).

If the reaction contains a radiation wavelength, this is
indicated by the presence of an 'HV' either as a reactant or as a product. Unless otherwise defined on additional lines as described in (2), the value of the wavelength is -1.0 if a reactant or +1.0 if a product.

If the reaction is a fall-off reaction, this is indicated either by a '(+M)' or a '+KNAME(K)', and there must be additional lines as described in (2) to define fall-off parameters.

2) (Subroutine CKAUXL)

Additional information for a reaction is given on lines immediately following the reaction description; this data will consist of a 'keyword' to denote the type of data, followed by a '/', then the required parameters for the keyword, followed by another '/'. There may be more than one keyword per line, and there may be any number of lines. The keywords and required parameters are as follows:

KNAME(K)/efficiency value/ - species (K) is an enhanced third body in the reaction
HV/wavelength/ - radiation wavelength parameter
LT/vall val2 val3/ - Landau-Teller coefficients
LOW/vall val2 val3/ - low fall-off parameters
TROE/vall val2 val3 val4/ - Troe fall-off parameters;
    if val4 is omitted, a default parameter will be used
EXTROE/vall val2 val3 val4 val5/ -
    Extended Troe fall-off parameters;
    vall=val4 = Troe parameters
    val5 = well # (for cal. ext reduced pressure)
SRI/vall val2 val3 val4/ - SRI fall-off parameters;
    if val4 is omitted, a default parameter will be used
    (it is an error to have both LT and Fall-off defined)
REV/par1 par2 par3/ - reverse parameters given
RLT/vall val2/ - Landau-Teller coefficients for reverse
    (it is an error if REV given and not RLT)

The end of all reaction data is indicated by an 'END' card or <eof>.

Resulting reaction data stored in LINC are:

II - integer number of reactions found
PAR(*,*) - array of real Arrhenius coefficients (NPAR, IDIM)
NSPEC(*) - total number of species in a reaction (IDIM)
    if NSPEC < 0, reaction is irreversible
NREAC(*) - number of reactants only (IDIM)
NUNK(*,*) - array of species numbers for reaction (MAXSP, IDIM)
NU(*,*) - array of stoichiometric coefficients (MAXSP, IDIM)
    of species in a reaction, negative=reactant, positive=product
NWL - number of reactions with radiation wavelength
IWL(*) - integer reaction numbers (IDIM)
WL(*) - real radiation wavelengths (IDIM)
NTHB - number of reactions with third bodies
**EN D** precision > double
**IMPLICIT** DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
**EN D** precision > single
**IMPLICIT** REAL (A-H,O-Z), INTEGER (I-N)

PARAMETER (MDIM=20, KDIM=500, MKDIM=MDIM*KDIM, IDIM=1500, LSYM=16,
1 NPAR=3, NPIDIM=IDIM*NPAR, NPC=5, NPCP2=NPC+2, MAXTP=3,
2 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
3 NLAR=2, NSIDIM=MAXSP*IDIM, NSIDIM=MAXSP*IDIM,
4 NLIDIM=NLAR*IDIM, NFAR=70, NFIDIM=NFAR*IDIM,
5 NTIDIM=KDIM*MAXSP, NTIDIM=KDIM*MAXSP,
6 NPAR=3, NPIDIM=IDIM*NPAR, NPC=5, NPCP2=NPC+2, MAXTP=3,
1 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
2 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
3 NLAR=2, NSIDIM=MAXSP*IDIM, NSIDIM=MAXSP*IDIM,
4 NLIDIM=NLAR*IDIM, NFAR=70, NFIDIM=NFAR*IDIM,
5 NTIDIM=KDIM*MAXSP, NTIDIM=KDIM*MAXSP,
6 NPAR=3, NPIDIM=IDIM*NPAR, NPC=5, NPCP2=NPC+2, MAXTP=3,
1 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
2 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
3 NLAR=2, NSIDIM=MAXSP*IDIM, NSIDIM=MAXSP*IDIM,
4 NLIDIM=NLAR*IDIM, NFAR=70, NFIDIM=NFAR*IDIM,
5 NTIDIM=KDIM*MAXSP, NTIDIM=KDIM*MAXSP,
6 NPAR=3, NPIDIM=IDIM*NPAR, NPC=5, NPCP2=NPC+2, MAXTP=3,
1 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
2 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
3 NLAR=2, NSIDIM=MAXSP*IDIM, NSIDIM=MAXSP*IDIM,
4 NLIDIM=NLAR*IDIM, NFAR=70, NFIDIM=NFAR*IDIM,
5 NTIDIM=KDIM*MAXSP, NTIDIM=KDIM*MAXSP,
6 NPAR=3, NPIDIM=IDIM*NPAR, NPC=5, NPCP2=NPC+2, MAXTP=3,
1 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
2 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
3 NLAR=2, NSIDIM=MAXSP*IDIM, NSIDIM=MAXSP*IDIM,
4 NLIDIM=NLAR*IDIM, NFAR=70, NFIDIM=NFAR*IDIM,
5 NTIDIM=KDIM*MAXSP, NTIDIM=KDIM*MAXSP,
6 NPAR=3, NPIDIM=IDIM*NPAR, NPC=5, NPCP2=NPC+2, MAXTP=3,
1 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
2 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
3 NLAR=2, NSIDIM=MAXSP*IDIM, NSIDIM=MAXSP*IDIM,
4 NLIDIM=NLAR*IDIM, NFAR=70, NFIDIM=NFAR*IDIM,
5 NTIDIM=KDIM*MAXSP, NTIDIM=KDIM*MAXSP,
6 NPAR=3, NPIDIM=IDIM*NPAR, NPC=5, NPCP2=NPC+2, MAXTP=3,
1 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
2 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
3 NLAR=2, NSIDIM=MAXSP*IDIM, NSIDIM=MAXSP*IDIM,
4 NLIDIM=NLAR*IDIM, NFAR=70, NFIDIM=NFAR*IDIM,
5 NTIDIM=KDIM*MAXSP, NTIDIM=KDIM*MAXSP,
6 NPAR=3, NPIDIM=IDIM*NPAR, NPC=5, NPCP2=NPC+2, MAXTP=3,
1 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
2 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
3 NLAR=2, NSIDIM=MAXSP*IDIM, NSIDIM=MAXSP*IDIM,
4 NLIDIM=NLAR*IDIM, NFAR=70, NFIDIM=NFAR*IDIM,
5 NTIDIM=KDIM*MAXSP, NTIDIM=KDIM*MAXSP,
6 NPAR=3, NPIDIM=IDIM*NPAR, NPC=5, NPCP2=NPC+2, MAXTP=3,
1 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
2 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
3 NLAR=2, NSIDIM=MAXSP*IDIM, NSIDIM=MAXSP*IDIM,
4 NLIDIM=NLAR*IDIM, NFAR=70, NFIDIM=NFAR*IDIM,
5 NTIDIM=KDIM*MAXSP, NTIDIM=KDIM*MAXSP,
6 NPAR=3, NPIDIM=IDIM*NPAR, NPC=5, NPCP2=NPC+2, MAXTP=3,
1 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
2 NTR=MAXTP-1, NKTDIM=NTR*NPCP2*KDIM, MAXSP=6, MAXTB=10,
3 NLAR=2, NSIDIM=MAXSP*IDIM, NSIDIM=MAXSP*IDIM,
4 NLIDIM=NLAR*IDIM, NFAR=70, NFIDIM=NFAR*IDIM,
5 NTIDIM=KDIM*MAXSP, NTIDIM=KDIM*MAXSP,
DIMENSION CHEB(70),T(51),RK(51),lnK(51),PAR(3)

CHARACTER*80 ISTR, IREAC, IPROD, ISPEC, INAME, ITEM

LOGICAL KERR, M_PPP, COMM

COMMON/FILE_IO/FILE_IN, FILE_OUT

, THERMO, ITHRM(KDIM)

Initialize variables

DATA KEY/'ELEM', 'SPEC', 'THER', 'REAC', 'END'/

, KERR/ .FALSE./

1 ITASK, NCHRG, MM, KK, II, NLAN, NFAL, NTHB, NREV, NRTL, NWL/11*0/,

2 ENAME, AMT/MDIM*' ', MDIM*0.0/, THERMO/ .TRUE./,

3 T/NDIM*-1.0/, KNAME, WTM, NT, KPHSE, KCHAR, ITHRM

4 /KDIM*' ', KDIM*0.0, KDIM*3, KDIM*0, KDIM*.FALSE./,

5 WL, IFOP, NTBS, IUP /IDIM*0.0/, THERMO/.TRUE./,

6 NSPEC, NREV, ILAN, IRTL, IWAL, IPAL, KFAL, ITHB/NDIM*0/

DATA NUNK, NU/NSIDIM*0, NSIDIM*0/, NKTB, AUK/NTIDIM*0, NTIDIM*-1.0/

DATA PAR,RPAR/NPIDIM*0.0, NPIDIM*0.0/

DATA PLAN,RLAN/NLIDIM*0.0, NLIDIM*0.0/

DATA PFAL/NFIDIM*0.0/, KNCF/MDIM*0.0/, A/NKTDIM*0.0/

ADD FILEIO SUBROUTINE 02/94 ING

CALL FILEIO_CVT

FILE_LOG = 'MECHCVT.LOG'

OPEN(UNIT=15, FILE=FILE_LOG, STATUS='UNKNOWN', FORM='FORMATTED')

5 WRITE(*,6)

6 FORMAT(' Enter PRESSURE (atm) for output mechanism : ')

READ(*,P)

IF (P.LE.1E2 .OR. P.GE.1E-3) GO TO 8

IF (P.GT.1E2 .OR. P.LT.1E-3) THEN

WRITE(*,7)P

7 FORMAT(' Pressure " ',1pe8.2,' atm " is out of 1.0e-3 and 1.0e2+2 atm range' )

write(*,*)' please re-enter a valid pressure '

write(*,*)' '

ENDIF

GO TO 5

8 CONTINUE

WRITE(*,*)' Enter Temperature range (K) for output mechanism '

51 WRITE(*,*)' Temperature (low) = '

READ(*,T_Low)

IF (T_LOW.LE.2500. .OR. T_LOW.GE.300.) GO TO 53

IF (T_LOW.LE.1E2 .OR. T_LOW.LE.1E-3) THEN

WRITE(*,52)T_Low

52 FORMAT(' Temperature " ',f7.1,' K " is out of 300 and 2500 K range' )

write(*,*)' please re-enter a valid temperature '

write(*,*)' '

ENDIF

GO TO 51

53 CONTINUE

WRITE(*,*)' Temperature (high) = '

READ(*,T_HIGH)

IF (T_LOW.LE.2500. .OR. T_LOW.GE.300.) GO TO 56

IF (T_HIGH.LE.2500. .OR. T_HIGH.LE.1E-3) THEN

WRITE(*,55)T_HIGH

55 FORMAT(' Temperature " ',f7.1,' K " is out of 300 and 2500 K range' )

write(*,*)' please re-enter a valid temperature '

write(*,*)' '

ENDIF
GO TO 54

56 CONTINUE
IF (T_LOW .GT. T_HIGH) THEN
  T_TEMP = T_LOW
  T_LOW = T_HIGH
  T_HIGH = T_TEMP
ENDIF
IF (T_LOW .EQ. T_HIGH) THEN
  T_HIGH = T_HIGH + 10.
  T_LOW = T_LOW - 10.
ENDIF

C
C
C  START OF MECHANISM CONVERSION
C
R=1.987
M = 0
LINE3 = ' '
LINE2 = ' '
LINE1 = ' '
ILEN3 = 0
ILEN2 = 0
ILEN1 = 0
COMM = .FALSE.
100 CONTINUE
IF (.NOT.COMM) THEN
  M = M + 1
  LINE3 = LINE2
  LINE2 = LINE1
  LINE1 = ' '
  ILEN3 = ILEN2
  ILEN2 = ILEN1
  ILEN1 = 0
ENDIF
COMM = .FALSE.
DO 102 I = 1, 120
  SUB(I) = ' '
102 CONTINUE
READ (LIN, '(A)', END=5000) LINE1
write(*,*),LINE1
105 CONTINUE
ILEN1 = IPPLEN(LINE1)
IF (LINE1(1:1) .EQ. '!'') THEN
  M = M -1
  COMM = .TRUE.
  GO TO 100
ENDIF
IF (ILEN1 .EQ. 0) GO TO 100
CALL CKISUB (LINE1(:ILEN1), SUB, NSUB)
N = 0
IF (UPCASE(SUB(1),4) .EQ. 'CHEB') THEN
  LINE = LINE3

  I1 = INDEX(SUB(NSUB),'/')
  I2 = INDEX(SUB(NSUB)(I1+1:),'/')
  WRITE(*,*),I1 = ',',I2 = ',',I2
IF (II.L EQ .0 .OR. I2.LE.0) THEN
  KERR = .TRUE.
  WRITE (LOUT, 2090) SUB(NSUB):ILEN)
ENDIF
C KEY = SUB(N):I1-1)
RSTR = ' 'RSTR = SUB(NSUB)(I1+1:I1+I2-1)
C WRITE(*,*)RSTR
CALL IPPARR (RSTR,1,-8,CHEB(1),NVAL,IER,LOUT)
N_CHEB = CHEB(1)
M_CHEB = CHEB(2)
NPARA_CHEB = N_CHEB*M_CHEB + 2
INDEX_CHEB = 0
C WRITE(*,*)' ..NPARA ..NPARA_CHEB ..N_CHEB ..M_CHEB
CALL IPPARR (RSTR,1,-5,PFA L(4,NFAL),NVAL,IER,LOUT)
INDEX_CHEB = INDEX_CHEB + NVAL
C WRITE (LOUT, 2028) SUB_CH(N):ILEN)
C WRITE(*,*)' ..NPARA ..INDEX ..NPARA_CHEB ..NVAL
C WRITE(*,*)' ..N ..M ..N_CHEB ..M_CHEB
C WRITE (LOUT, 3098) (PFA L(L,NFAL),L=4,NVAL+3)
CONTINUE
LINE_CH = ' 'READ ('LIN,'(A)') LINE_CH
C WRITE(*,*)LINE_CH
C WRITE(*,*)LINE_CH
C ILEN = IPPLEN(LINE_CH)
C WRITE(*,*)ILEN
CALL CKISUB (LINE_CH(:ILEN), SUB_CH, NSUB_CH)
C KEY = ' 'DO 151 N_CH = 1, NSUB CH
  IILEN = ILASCH(SUB_CH(NSUB_CH))
  WRITE(*,*)' NSUB_CH = ',NSUB_CH
  WRITE(*,*)SUB_CH(NSUB_CH)
  II1 = INDEX(SUB_CH(NSUB_CH)(),'/')
  II2 = INDEX(SUB_CH(NSUB_CH)(II1+1 : ),'/')
  WRITE(*,*)' ..II1 ..II2 ..II1,II2
  IF (II1.LE.0 .OR. II2.LE.0) THEN
    KERR = .TRUE.
    WRITE (LOUT, 2090) SUB_CH(N_CH)(:IILEN)
GO TO 199
ENDIF
C KEY = SUB(N):I1-1)
RSTR_CH = ' 'RSTR_CH = SUB_CH(NSUB_CH)(I1+1:I1+I2-1)
C WRITE(*,*)RSTR_CH
CALL IPPARR (RSTR_CH,1,-8,CHEB(INDEX_CHEB+1)
INDEX_CHEB = INDEX_CHEB + NVAL
C WRITE(*,*)' ..INDEX ..NVAL ..INDEX_CHEB ..NVAL
C WRITE(*,*)' ..INDEX ..NVAL ..INDEX_CHEB ..NVAL
C WRITE(*,*)' ..INDEX ..NVAL
INDEX_CHEB = INDEX_CHEB + NVAL
C WRITE(*,*)' ..INDEX ..INDEX_CHEB
C WRITE (LOUT, 2028) SUB_CH(N):ILEN)
CONTINUE
IF (INDEX_CHEB.LT.NP PARA_CHEB) GOTO 101
ELSE
  WRITE (LOUT, 2028) SUB_CH(N):ILEN)
  KERR = .TRUE.
ENDIF

C  ******************************************************
C  ************** W.ING 6/16/95  ********************

C201  N  =  N  +  1
C WRITE(LOUT,*)' ....CALL SUB CHEB....',N

C WRITE(LOUT,810)LINE3
C LINE3 = LINE2
C LINE2 = LINE1
C ILEN3 = ILEN2
C ILEN2 = ILEN1
C READ (LIN,'(A)',END=5000) LINE1
C ILEN1 = IPPLEN(LINE1)
C CALL CKISUB (LINE1; ILEN1, SUB, NSUB)
C IF(UPCASE(SUB(1),4) .EQ. 'CHEB') GOTO 101
C WRITE(LOUT,810)LINE3
C LINE3 = ' '
C LINE2 = ' '
C LINE1 = ' '
C ILEN3 = 0
C ILEN2 = 0
C ILEN1 = 0
C DO 299 N = 1, NPAR_A_CHEB
C WRITE(LOG,* ) 1 CHEB( ' , N , 1) = ' , CHEB(N)
C299  CONTINUE
Tmin * *  300.
Tmax * *  2500.
Pmin = 1.0E-03
Pmax = 1.0E+02

C--------------Find NPAR real parameters---------------------
C NPAR = 3
CALL IPNPAR (LINE, NPAR, ISTR, ISTART)
CALL IPPARR (ISTR, 1, NPAR, PAR(I), NVAR, IER, LOUT)
IF (IER .NE. 0) KERR = .TRUE.
C-------Remove blanks from reaction string
C
C INAME = ' '
C ILEN = 0
C DO 10 I = 1, ISTART-1
C IF (LINE(I:I) .NE. ' ') THEN
C ILEN = ILEN+1
C INAME(ILEN:ILEN) = LINE(I:I)
C ENDIF
C END
C 10  CONTINUE
C INAME = ' '
C ILEN = 0
M_PPP = .TRUE.
DO 10 I = 1, ISTART-1
IF (LINE(I:I) .NE. ' (' .AND. M_PPP ) THEN
ILEN = ILEN+1
INAME(ILEN:ILEN) = LINE(I:I)
ELSEIF (LINE(I:I).EQ. '(' .OR..NOT.M_PPP) THEN
ILEN = ILEN+1
INAME(ILEN:ILEN) = ' '
ENDIF
IF (LINE(I:I) .EQ. '(' ) M_PPP = .FALSE.
IF (LINE(1:1) .EQ. ' ') M_PPP = .TRUE.
10 CONTINUE

C
C-----Find reaction string, product string
C
I1 = 0
I2 = 0
DO 25 I = 1, ILEN
IF (II .LE. 0) THEN
IF (INAME(I:1+2) .EQ. '<=>') THEN
I1 = 1
I2 = I+2
IR = 1
ELSEIF (INAME(I:1+1) .EQ. '=>') THEN
I1 = I
I2 = I+1
IR = -1
ELSEIF (I.GT.1 .AND. INAME(I: I ) .EQ.'=' .AND. INAME(I-1:I-1).NE.'=') THEN
I1 = I
I2 = I
IR = 1
ENDIF
ENDIF
25 CONTINUE
C
IF (ILASCH(INAME).GE.45 .AND. I1.GT.0) THEN
WRITE (LOUT, 1900) INAME(:11-1),Af it,rnfit,Eafit*1000.
WRITE (LOUT, 1920) INAME(I1:)
ELSE
WRITE (LOUT, 1900) INAME(:45),Af it,rnfit,Eafit*1000.,
FILE_IN,AVGERR
ENDIF
C
IF (T_HIGH .EQ. T_LOW) THEN
C
N_TFIT = 1
nK = 1
C
ELSE
N_TFIT = 21
nK = 1
C
ENDIF

DO 301 NFIT = 1, N_TFIT
T(NFIT) = T_LOW + (T_HIGH-T_LOW)/(N_TFIT-1)*(NFIT-1)
c WRITE(*,*) ' T(NFIT) = ', T(NFIT)
2 T_cheb = (2.*(1./T(NFIT))-(1/Tmin)-(1/Tmax)) /
( (1/Tmax)-(1/Tmin))
c WRITE(*,*)' T_cheb = ', T_cheb
P_cheb = (2.*dlog10(P)-dlog10(Pm in)-
dlog10(Pmax)) / (dlog10(Pmax)-dlog10(Pm in))
c WRITE(*,*)' P_cheb = ', P_cheb
RK_tempLog = 0.0
c to avoid fortran compiler significant digits error
IF (t_cheb .ge. 0.9999999999999) t_cheb = 1.0
do 82 I = 1, CHEB(1)
do 81 J = 1, CHEB(2)
c write(*,*)RK_tempLog,((I-1)*M_cheb+J),X((I-1)*M_cheb+J),
c 2 PhiCh(I,T_cheb),PhiCh(J,P_cheb)
c WRITE(*,*)T_LOW,T_HIGH,NFIT,T(NFIT)
c WRITE(*,77)1,J,NFIT,T(NFIT),T_CHEB,P_CHEB
c77 FORMAT('I=',I2, 'J = ', I 2, * T (', 12, 1) = ', F7.2, ', 2F9.2)
c TT = PHICH(J,P_CHEB)
c WRITE(*,*)'P_CHEB = ',TT
c TT = PHICH(I,T_CHEB)
c WRITE(*,*)'T_CHEB = ',TT
RK_tempLog = RK_tempLog + CHEB((I-1)*
2 CHEB(2)+J)*PhiCh(I,T_cheb)*
3 PhiCh(J,P_cheb)
CONTINUE

RK(NFIT) = 10.0**RK_tempLog

CONTINUE

nK = 21

do 45 I=1,nK

lnK(I) = dlog(RK(I))

45 continue

nlow = 1

nhigh = nK

WRITE(LOG,*), '

WRITE(LOG,*), '

IF (IIASCH(INAME).GE.45 .AND. I1.GT.0) THEN

WRITE (LOG, 1800) INAME(:I1-1)

WRITE (LOG, 1820) INAME(I1:

ELSE

WRITE (LOG, 1800) INAME(:45)

ENDIF

ENDIF

WRITE(LOG,*), '

call arrnlin(LOG, 0.0d0, nK, 3, T, lnK, nlow, nhigh,

Afit, rnfih, Eafih, errpar)

avgerr=0.0

do 300 I = 1,nK

K_fit = Afit*T(I)**rnfih*exp(-Eafih*1000./R/T(I))

error = 100.0d0*(rk(I) - K_fit) /

2
dmin1(rk(I), K_fit)

avgerr = avgerr + dabs(error)

WRITE(LOG,277)T(I),rk(I),K_fit,error

1


continue

avgerr = avgerr/nK

write(LOG,*), '

write(LOG,320) Afit, rnfih, Eafih, avgerr

format(8X,'Fitted A = ',1pe10.3, ' n = ', 0pf6.2, ' Ea = ',

1ipe10.3, ' avg error ', 0pf7.2, '%')

ELSE

IF (LINE3.NE.' ' .AND. M.GT.2) WRITE(LOUT,810)LINE3

ENDIF

GOTO 100

IF (LINE3 .NE. ' ') WRITE(LOUT,810) LINE3

IF (LINE2 .NE. ' ') WRITE(LOUT,810) LINE2

WRITE('(*,*)','...END'

810 FORMAT(A80)

1800 FORMAT (A)

1820 FORMAT (6X,A)

1900 FORMAT ( A, T47, 1PE8.2, 2X, 0PF7.3, 1X, F8.0,' ! ',A12, 1

1 ' avg err ',f6.2,'%')

1920 FORMAT (6X,A)

2028 FORMAT (6X,'Error in CHEB fall-off parameters...',A)

2090 FORMAT (6X,'Error in auxiliary data...',A)

3098 FORMAT (6X,'CHEB Polynomials: ','8E13.5)

CLOSE (LIN)

CLOSE (LOG)

CLOSE (LOUT)

STOP

END
**Function PhiCh**

```c
FUNCTION PhiCh(i,x)
implicit none
local variables
integer i
real*8 x,PhiCh

PhiCh = DCOS((i-1)*DACOS(x))
return
end
```

**Subroutine arrnlin**

```c
subroutine arrnlin(lerr,dummy1,nData,npara,xin,yin,nlow,nhigh,
+ AfIt,rnfit,EafIt,chiSq)
implicit none
local variables
integer mxWells, mxProds, npara
parameter (mxWells = 12, mxProds = 10)
integer mxTpts, mxPpts
parameter (mxTpts = 20, mxPpts = 61)
integer mxColls
parameter (mxColls = 10)
integer mxFreqs
parameter (mxFreqs = 6)
integer mxFitF
parameter (mxFitF = 20)
integer mxKfit
parameter (mxKFit = 6)
integer nlow,nhigh
real*8 xin(mxTpts), yin(mxTpts),AfIt,rnfit,EafIt,dummy1,chiSq
integer listA(3)
real*8 sig(mxTpts), A(3), covar(3,3)
external fArrn,fArrn3
end declarations

c nlow and nhigh are part of old autoranging stuff - here, we fix and
c leave alone
nlow = 1
nhigh = nData
c perform some initializing
do 20 i = 1,nData
   sig(i) = 1.0
20 continue
c listA is a vector telling what coefficients to adjust;
c we want all of them
do 50 i = 1,npara
   listA(i) = i
50 continue
c note we added add'l argument lerr to this sub
if (npara.eq.2) then
   call lfit (xin,yin,sig,nData,A,npara,listA,npara,covar,
   + npara,chiSq,fArrn,lerr)
endif
if (npara.eq.3) then
   call lfit (xin,yin,sig,nData,A,npara,listA,npara,covar,
   + npara,chiSq,fArrn3,lerr)
endif
c set parameters for output
if (npara.eq.2) then
   AfIt = dexp(A(1))
   EafIt = -A(2)*1.987/1000
endif
if (npara.eq.3) then
```
\[
\text{Afit} = \text{dexp}(\text{A}(1)) \\
\text{rnfir} = \text{A}(2) \\
\text{Eafit} = -\text{A}(3) \times 1.987/1000.
\]

end if
return
end

c***FARRN****************************
c extern basis functions for lsq fit of arrhenius coefficients
subroutine fArrn(x,VecCoefs,nCoefs)
imPLICIT NONE
INTEGER nCoefs
REAL*8 x,VecCoefs(nCoefs)
VecCoefs(1) = 1.d0
VecCoefs(2) = 1/x
return
end

c***FARRN3***************************
c extern basis functions for lsq fit of arrhenius coefficients
subroutine fArrn3(x,VecCoefs,nCoefs)
imPLICIT NONE
INTEGER nCoefs
REAL*8 x,VecCoefs(nCoefs)
VecCoefs(1) = 1.d0
VecCoefs(2) = dlog(x)
VecCoefs(3) = 1/x
return
end

c this is all matrix solving stuff from press et al., numerical recipies
*****ludcmp*******************************
subroutine ludcmp(A,n,np,indx,d,vv,ierr,lout)
c this is adapted from numerical recipes by press et al. <ayc 3/93>
c this is an lu decomposer and the output is passed to subsequent
routines
Note: change of inputs - esp array vv so it can be fixed in size
from above
c A is input array of n equations and is physically dimensioned np x np
c note input matrix A is changed by this routine to give LU decomposed A
c indx is output 1-D array of pivot info - this along with variable D is
c passed to companion subroutines
Note: code assumes no-trip do's (fortran 77 standard)
imPLICIT NONE
REAL*8 TINY
PARAMETER(TINY = 1.d-175)
INTEGER N, NP, INDEX(NP), IMAX, IERR, LOUT
INTEGER I, J, K
REAL*8 A(NP,NP), VV(NP), D, AAMAX, SUM, DUM
IERR = 0
TINY = 1.d-175
D = 1.d0
do 120 I = 1,N
   AAMAX = 1.d0
   DO 110 J = 1,N
      IF (DABS(A(I,J)) .GT. AAMAX) AAMAX = DABS(A(I,J))
   110 CONTINUE
   IF (AAMAX .LE. TINY) THEN
      WRITE(OUT,*) 'ERROR: singular matrix.'
      IERR = 1
      RETURN
   ENDIF
   VV(I) = 1.d0/AAMAX
120 CONTINUE
DO 190 J = 1,N
   DO 140 I = 1,J-1
      SUM = A(I,J)
      DO 130 K = 1,I-1

```
sum = sum - A(i,k)*A(k,j)
continue
A(i,j) = sum
continue

aamax = 0.d0
do 160 i = j,n
   sum = A(i,j)
   do 150 k = 1,j-1
      sum = sum - A(i,k)*A(k,j)
   continue
A(i,j) = sum
   dum = vv(i)*dabs(sum)
   if (dum .ge. aamax) then
      imax = i
      aamax = dum
   endif
continue
if (j .ne. imax) then
   do 170 k = 1,n
      dum = A(imax,k)
      A(imax,k) = A(j,k)
      A(j,k) = dum
   continue
   d = -d
   vv(imax) = vv(j)
endif

   indx(j) = imax
   if (dabs(A(j,j)) .le. tiny) A(j,j) = dsign(tiny,A(j,j))
   if (j .ne. n) then
      dum = 1.d0/A(j,j)
      do 180 i = j+1,n
         A(i,j) = A(i,j)*dum
      continue
      return
end
```

**subroutine lubksb(A,n,np,indx,B)**

- This is taken from Numerical Recipes by Press et al. (ayc 3/93).
- A is LU decomposed array of n equations, physically dimensioned np x np.
- And was obtained by ludcmp.
- Index is 1-D array of pivot info from lubksb.
- B is input column vector (A X = B) but is changed on output to give X.

Implicit none

Integer n,np,indx(np),ii,ll
Integer i,j
Real*8 A(np,np),B(np),sum
ii = 0
do 120 i = 1,n
   ll = indx(i)
   sum = B(ll)
   B(ll) = B(i)
   if (ii .ne. 0) then
      do 110 j = ii,1,-1
         sum = sum - A(i,j)*B(j)
      continue
      elseif (sum .ne. 0.d0) then
         ii = i
      endif
   B(i) = sum
110   continue
   do 140 i = n,1,-1
      sum = B(i)
      do 130 j = i+1,n
         sum = sum - A(i,j)*B(j)
   continue
   return
end
```
130 continue  
   \[ B(i) = \frac{\text{sum}}{A(i,i)} \]  
140 continue  
return  
end

c*** lfit **********************************
subroutine lfit(x, y, sig, ndata, a, ma, lista, mfit, covar, ncvm,  
2 chisq, funcs, ierr)

c this is taken from numerical recipes by press et al. <ayc 4/93>
c its pretty complicated so you'd better see description in book

c comments in code are mine <ayc>
c this program allows you to only adjust mfit of ma coefficients, 
c while keeping the rest fixed - lista is an input vector specifying 
c what parameters you want adjusted
implicit none

c mmmax is max \# of coefficients
integer mmmax
parameter (mmmax = 10)
integer ndata, ma, lista(ma), mfit, ncvm
integer kk, ihit, i, j, k
real*8 x(ndata), y(ndata), sig(ndata), a(ma), covar(ncvm, ncvm),  
2 beta(mmmax), afunc(mmmax), chisq  
real*8 ym, sig2i, wt, sum  
external funcs

c the following are for ldu routines which we now use  
c instead of gaussj
integer indx(mmmax), ierr, lerr
real*8 d, vv(mmmax)

c done declarations
kk = mfit + 1

12 continue
   j = 1, ma
   ihit = 0
   do 11 k = 1, mfit
      if (lista(k).eq.j) ihit = ihit + 1
      continue
      if (ihit.eq.0) then
         lista(kk) = j
         kk = kk + 1
      else if (ihit.gt.1) then
         stop 'improper set in lista'
      endif
      continue
   end do 11

12 continue
   if (kk.ne.(ma+1)) stop 'improper set in lista'
   do 13 j = 1, mfit
      do 12 k = 1, mfit
         covar(j,k) = 0.d0
      end do 12
      beta(j) = 0.d0
   end do 13

14 continue
   do 18 i = 1, ndata
      call funcs(x(i), afunc, ma)
      ym = y(i)
      if (mfit.lt.ma) then
         do 15 j = mfit+1, ma
            ym = ym - a(lista(j)) * afunc(lista(j))
         end do 15
      endif
      sig2i = 1.d0/sig(i)**2
      do 16 j = 1, mfit
         wt = afunc(lista(j))*sig2i
         do 15 k = 1, j
            covar(j,k) = covar(j,k) + wt*afunc(lista(k))
         end do 15
      end do 16
      beta(j) = beta(j) + ym*wt
      continue
   end do 18

if (mfit.gt.1) then
  do 21 j = 2, mfit
    do 19 k = 1, j-1
      covar(k,j) = covar(j,k)
    19 continue
  21 continue
endif

 we have replaced this with call to ldu solver
 c call gaussj(covar,mfit,ncvm,beta,1,1)
 c we use lud routines instead of gaussj
 c we aren't going to do anything with inverse
 c so we just solve to get solution vector
 c NOTE: we have changed some of the inputs of these
 c routines as compared to book!!
call ludcmp(covar,mfit,ncvm,index,d,vv,ierr,lerr)
call lubksb(covar,mfit,ncvm,index,beta)

 this has totally messed up covar but we don't
 c need it anymore - see book to get the 3 lines
 c of code to compute inverse of covar from here
 do 22 j = 1, mfit
 a(lista(j)) = beta(j)
22 continue

 chisq = 0.0d0
 do 24 i = 1, ndata
   call funcs(x(i), afunc, ma)
   sum = 0.0d0
   do 23 j = 1, ma
     sum = sum + a(j) * afunc(j)
   23 continue
   chisq = chisq + ((y(i) - sum) / sig(i))**2
24 continue

 c this sorts covariant matrix to original ordering
 c of fitting coefficients; here we neither changed
 c ordering nor do we care about this matrix so
 c we don't worry about it - note with gaussj,
c c covar would have been the inverse of the original;
here its become befuddled - but if we cared we
c c could fix it
return
end

C

C---------------------------------------------------------------------------------------C
SUBROUTINE CKCHAR (SUB, NSUB, NDIM, STRAY, RAY, NN, KERR, LOUT)
C
C Extracts names and real values from an array of CHAR*(*)
C substrings; stores names in STRAY array, real values in RAY;
C i.e. can be used to store element and atomic weight data,
C species names, etc.
C
C Input: SUB(N), N=1, NSUB - array of CHAR*(*) substrings
C NSUB     - number of substrings
C NDIM     - size of STRAY, RAY arrays
C NN       - actual number of STRAY found
C STRAY(N), N=1, NN - CHAR*(*) array
C RAY(N), N=1, NN - Real array
C LOUT     - output unit for error messages
C
C Output: NN  - incremented if more STRAY found
C STRAY(N), N=1, NN - incremented array of STRAY
C RAY(N), N=1, NN - incremented array of reals
C KERR     - logical, .TRUE. = error in data
C
C F. Rupley, Div. 8245, 2/5/88
C---------------------------------------------------------------------------------------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
DIMENSION RAY(*), PAR(1)
CHARACTER SUB(*), STRAY(*), ISTR*80, UPCASE*4
LOGICAL KERR
C
ILEN = LEN(STRA(Y(1))
C
DO 200 N = 1, NSUB
   IF (UPCASE(SUB(N), 3) .EQ. 'END') RETURN
   I1 = INDEX(SUB(N), '/')
   IF (I1 .EQ. 1) THEN
      KERR = .TRUE.
      WRITE (LOUT, 130) SUB(N)(:ILASCH(SUB(N)))
   ELSE
      IF (I1 .LE. 0) THEN
         ISTR = SUB(N)
      ELSE
         ISTR = SUB(N)(:I1-1)
      ENDIF
      CALL CKCOMP (ISTR, STRAY, NN, INUM)
   ENDIF
   CALL IPPARR (ISTR, 1, 1, PAR, NVAL, IER, LOUT)
   IF (IER .EQ. 0) THEN
      RAY(NN) = PAR(1)
   ELSE
      KERR = .TRUE.
   ENDIF
200 CONTINUE
C
100 FORMAT (6X,'Warning... duplicate array element ignored...',A)
110 FORMAT (6X,'Error... character array size too small for ....',A)
120 FORMAT (6X,'Error... character array element name too long...',A)
130 FORMAT (6X,'Error... misplaced value...',A)
END
C ---------------------------------------------------------------
C
SUBROUTINE CKAWTM (ENAME, AWT)
C
Returns atomic weight of element ENAME.
Input: ENAME - CHARM(*) element name
Output: AWT - real atomic weight

F. Rupley, Div. 8245, 11/11/86

C *****precision > double
IMPLICIT DOUBLE PRECISION (A-H,0-Z), INTEGER (I-N)
C *****END precision > double
C *****precision > single
IMPLICIT REAL (A-H,0-Z), INTEGER (I-N)
C *****END precision > single

PARAMETER (NATOM = 102)
DIMENSION ATOM(NATOM)
CHARACTER ENAME(*), IATOM(NATOM)*2, UPCASE*2

DATA (IATOM(I),ATOM(I),I=1,40) /
  'F ', 18.99840, 'NE', 20.18300, 'NA', 22.98980, 'MG', 24.31200,
  'CL', 35.45400, 'AR', 39.94800, 'K ', 39.10200, 'CA', 40.08000,
  'SC', 44.95600, 'TI', 47.90000, 'CR', 50.94200, 'CE', 51.99600,
  'MN', 54.95600, 'FE', 55.84700, 'CO', 58.93320, 'NI', 58.71000,
  'AS', 74.92160, 'SE', 78.96000, 'BR', 81.91000, 'RE', 82.59000,
  'Y ', 89.06400, 'ZR', 91.22000/

DATA (IATOM(I),ATOM(I),I=41,80) /
  'NB', 92.90600, 'MO', 95.94000, 'TC', 99.00000, 'RU', 101.07000,
  'RH', 102.90500, 'PD', 106.40000, 'AG', 107.87000, 'CD', 112.40000,
  'IN', 114.82000, 'SN', 118.69000, 'SB', 121.75000, 'TE', 127.60000,
  'I ', 126.90440, 'XO', 131.30000, 'CS', 132.90500, 'BA', 137.34000,
  'LA', 138.91000, 'CE', 140.12000, 'PR', 140.97000, 'ND', 144.24000,
  'PM', 145.00000, 'SM', 150.35000, 'EU', 151.96000, 'GD', 157.25000,
  'TB', 158.94200, 'DY', 162.50000, 'HO', 164.93000, 'ER', 167.26000,
  'TM', 168.93400, 'YW', 173.04000, 'PB', 178.49000, 'OS', 180.00000,
  'IR', 182.94800, 'W ', 183.85000, 'RE', 186.20000, 'OS', 189.00000,

DATA (IATOM(I),ATOM(I),I=81,NATOM) /
  'TL', 204.37000, 'PB', 207.19000, 'BI', 208.98000, 'PO', 210.00000,
  'AT', 210.00000, 'RN', 222.00000, 'FR', 223.00000, 'RA', 226.00000,
  'AC', 227.00000, 'TH', 232.03800, 'PA', 231.00000, 'U ', 238.03000,
  'NP', 237.00000, 'PU', 242.00000, 'AM', 243.00000, 'CM', 247.00000,
  'BK', 249.00000, 'CF', 251.00000, 'ES', 254.00000, 'FM', 253.00000,
  'D ', 260.01410, 'E ', 5.45E-4/

CALL CKCOMP (UPCASE(ENAME, 2), IATOM, NATOM, L)
IF (L .GT. 0) AWT = ATOM(L)
RETURN
END

C subroutine CKTHRM (LUNIT, MDIM, ENAME, MM, AWT, KNAMEx, KK, KNCF, KPHSE, KCHRG, WTM, MAXTP, NT, NTR, TLO, TMID, T, NPCP2, A, ITHRM, KERR, LOUT, ISTR)

C Finds thermodynamic data and elemental composition for species
C Input: LUNIT - unit number for input of thermo properties
C         MDIM - maximum number of elements allowed
C         ENAME(M),M=1,MM - array of CHAR(*) element names
C         MM - total number of elements declared
C         AWT(M),M=1,MM - array of atomic weights for elements
C         KNAMEx(K),K=1,KK - array of CHAR(*) species names
C         KK - total number of species declared

C--- subroutine CKTHRM (LUNIT, MDIM, ENAME, MM, AWT, KNAMEx, KK, KNCF, KPHSE, KCHRG, WTM, MAXTP, NT, NTR, TLO, TMID, T, NPCP2, A, ITHRM, KERR, LOUT, ISTR)
LOUT - output unit for messages
NT(K), K=1, KK - number of temperature values
NTR - number of temperature ranges
Output: KNCF(M,K) - elemental composition of species
KPHSE(K), K=1, KK - integer array, species phase
KCHRG(K), K=1, KK - integer array of species charge
   =0, if no electrons,
   =(-1)*number of electrons present
WTM(K), K=1, KK - array of molecular weights of species
A(M,L,K) - array of thermodynamic coefficients
T(N), N=1, NT - array of temperatures
KERR - logical error flag

*****precision > double
IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
*****END precision > double
*****precision > single
IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
*****END precision > single
C
DIMENSION WTM(*), NT(*), T(MAXTP,*), KPHSE(*), KCHRG(MDIM,*),
1 KCHRG(*), A(NPCP2,NTR,*), AWT(*); VALUE(5)
C     CHARACTER ENAME(*)*(*) , KNAM(*)*(*) , LINE(4)*80, ELEM*16
C     CHARACTER UPCASE*4, ISTR*80, SUB(80)*80
LOGICAL KERR, ITHRM(*)
C
IF (MM.LE.0 .OR. KK.LE.O) WRITE (LOUT, 80)
C
GO TO 20
10 CONTINUE
ISTR = ' '
READ (LUNIT,' (A)' ,END=40) ISTR
20 CONTINUE
ILEN = IPPLEN(ISTR)
IF (ILEN .LE. 0) GO TO 10
C
CALL CKISUB (ISTR(:ILEN), SUB, NSUB)
CALL CKCOMP (SUB(1), KNAM, KK, K)
IF (K .EQ. 0) THEN
   IF (UPCASE(SUB(1), 3) .EQ. 'END' .OR.
1   UPCASE(SUB(1), 4) .EQ. 'REAC') RETURN
   GO TO 10
ENDIF
C
IF (ITHRM(K)) GO TO 10
ITHRM(K) = .TRUE.
LINE(1) = ' '
LINE(1) = ISTR
DO 25 L = 2, 4
   LINE(L) = ' '
   READ (LUNIT,' (A)' ,END=40) LINE(L)
25 CONTINUE
C
ICOL = 20
DO 60 I = 1, 5
   ICOL = ICOL + 5
   IF (I .EQ. 5) ICOL = 74
   ELEM = LINE(1)(ICOL:ICOL+1)
   IELEM = 0
C
   IF (LINE(1)(ICOL+2:ICOL+4) .NE. ' ') THEN
      CALL IPPARR
1      (LINE(1)(ICOL+2:ICOL+4), 0, 1, VALUE, NVAL, IER, LOUT)
   IELEM = VALUE(1)
   ENDIF
C

IF (ELEM.NE.' ' .AND. IELEM.NE.0) THEN
  IF (UPCASE(ELEM, 1) .EQ. 'E')
  1 KCHRG(K)=KCHRG(K)+IELEM*(-1)
  CALL CKCOMP (ELEM, ENAME, MM, M)
  IF (M .GT. 0) THEN
    KNCF(M,K) = IELEM
    WTM(K) = WTM(K) + AWT(M)*FLOAT(IELEM)
  ELSE
    WRITE (LOUT, 100) ELEM, KNAME(K)(:10)
    KERR = .TRUE.
  ENDIF
ENDIF
60 CONTINUE
C
IF (UPCASE(LINE(1)(45:),1) .EQ. 'L') KPHSE(K)=1
IF (UPCASE(LINE(1)(45:),1) .EQ. 'S') KPHSE(K)=-1
C
Currently allows for three temperatures, two ranges;
C in future, NT(K) may vary, NTR = NT(K)-1
C
T(1,K) = TLO
IF (LINE(1)(46:55) .NE. ' ') CALL IPPARR
  1 (LINE(1)(46:55), 0, 1, T(1,K), NVAL, IER, LOUT)
C
T(2,K) = TMD
IF (LINE(1)(66:73) .NE. ' ') CALL IPPARR
  1 (LINE(1)(66:73), 0, 1, T(2,K), NVAL, IER, LOUT)
C
T(NT(K),K) = THI
IF (LINE(1)(56:65) .NE. ' ') CALL IPPARR
  1 (LINE(1)(56:65), 0, 1, T(NT(K),K), NVAL, IER, LOUT)
C
READ (LINE(2)(:75),'(5E15.8)') (A(I,NTR,K),I=1,5)
READ (LINE(3)(:75),'(5E15.8)')
  1 (A(I,NTR,K),I=6,7), (A(I,1,K),I=1,3)
READ (LINE(4)(:60),'(4E15.8)') (A(I,1,K),I=4,7)
GO TO 10
C
40 RETURN
80 FORMAT (6X,'Warning... THERMO cards misplaced will be ignored...')
100 FORMAT (6X,'Error... element...','A','not declared for...','A)
END
C-----------------------------------------------C
SUBROUTINE CKREAC (LINE, II, KK, KNAME, LOUT, MAXSP, NSPEC, NREAC,
  1 NUNK, NU, NFAL, PAR, NTHB, ITHE,
  2 NFAI, IFAL, KFAI, NWL, IWL, WL, KERR)
C
CKREAC parses the main CHAR*(*) line representing a gas-phase
reaction; first, the real Arrhenius parameters are located and
stored in PAR[N,I], N=1,NFAR, where I is the reaction number;
then a search is made over the reaction string:
'=': reaction I is reversible;
'->': reaction I is irreversible;
'(+[n]KNAME(K))': reaction I is a fall-off reaction;
NFAL is incremented, the total number of
fall-off reactions;
IFAL(NFAL)=I, KFAL(NFAL)=K;
this species is eliminated from consideration
as a reactant or product in this reaction.
'(+M)': reaction I is a fall-off reaction;
NFAL is incremented, IFAL(NFAL)=I, KFAL(NFAL)=0;
'(+[n]KNAME(K)'): NSPEC(I) is incremented, the total number of
species for this reaction;
  n is an optional stoichiometric coefficient
  of KNAM\(\text{E}(K)\), if omitted, \(n=1\);
  if this string occurs before the \(=\),
  NREAC(II) is incremented, the total number of
reactants for this reaction, \(NUNK(N,1)=K\), and
  \(NU(N,1) = -n\), where \(N=1-3\) is reserved for
reactants;
  if this string occurs after the \(=\),
  \(NUNK(N,1) = K\), and \(NU(N,1) = n\), where \(N=4-6\)
is reserved for products;

  'M' : I is a third-body reaction; NTHB is incremented, the
total number of third-body reactions, and ITHB(NTHB)=1.

Input: LINE - a CHAR* line (from data file)
  II - the index of this reaction, and the total number
  of reactions found so far.
  KK - actual integer number of species
  KNAM\(\text{E}(K)\), \(K=1,KK\) - array of CHAR* species names
  LOUT - output unit for error messages
  MAXSP - maximum number of species allowed in reaction
  NPAR - number of parameters expected
  A '!!' will comment out a line, or remainder of the line.

Output: NSPEC - total number of reactants+products in reaction
  NREAC - number of reactants
  NUNK - species numbers for the NSPEC species
  NU - stoichiometric coefficients for the NSPEC spec.
  NFAL - total number of fall-off reactions
  IFAL - reaction numbers for the NFAL reactions
  KFAL - 3rd body species numbers for the NFAL reactions
  NTHB - total number of 3rd-body reactions
  ITHB - reaction numbers for the NTHB reactions
  NWL - number of radiation-enhanced reactions
  IWL - reaction numbers for the NWL reactions
  WL - radiation wavelengths for the NWL reactions
  KERR - logical, .TRUE. = error in data file

F. Rupley, Div. 8245, 5/13/86

 ifstream > double
 IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
 IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)

 DIMENSION NSPEC(*), NREAC(*), NUNK(MAXSP,*), NU(MAXSP,*),
 1 PAR(NPAR,*), IFAL(*), KFAL(*), ITHB(*), IWL(*), WL(*),
 2 IPLUS(20)
 CHARACTER KNAM\(\text{E}(K)\)(*), LINE(*), INUM(10)*1, UPCASE*4
 CHARACTER*80 ISTR, IREAC, IPROD, ISPEC, INAME, ITEMP
 LOGICAL KERR, LTHB, LWL
 DATA INUM/'0','1','2','3','4','5','6','7','8','9'/

 LTHB = .FALSE.
 LWL = .FALSE.
 NSPEC(II) = 0
 NREAC(II) = 0

 Find NPAR real parameters-------------------------

 CALL IPNPAR (LINE, NPAR, ISTR, ISTART)
 CALL IPPARR (ISTR, 1, NPAR, PAR(1,II), NVAL, IER, LOUT)
 IF (IER .NE. 0) KERR = .TRUE.
C Remove blanks from reaction string
C
   INAME = ' '  
   ILEN = 0
   DO 10 I = 1, ISTART-1
      IF (LINE(I:I) .NE. ' ') THEN  
         ILEN = ILEN+1
         INAME(ILEN:ILEN) = LINE(I:I)
      ENDIF
   10 CONTINUE
C
C Find reaction string, product string
C
   I1 = 0
   I2 = 0
   DO 25 I = 1, ILEN
      IF (I1 .LE. 0) THEN
         IF (INAME(I:I+2) .EQ. '<=>') THEN  
            I1 = I
            I2 = I+2
            IR = 1
         ELSEIF (INAME(I:I+1) .EQ. '=>') THEN  
            I1 = I
            I2 = I+1
            IR = -1
         ELSEIF (I.GT.1 .AND. INAME(I : I) .EQ. '='  
            .AND. INAME(I-1:I-1) .NE. '=' ) THEN
            I1 = I
            I2 = I+1
            IR = 1
         ENDIF
      ENDIF
   25 CONTINUE
C
   IF (ILASCH(INAME).GE.45 .AND. I1.GT.0) THEN
      WRITE (LOUT, 1900) I1,INAME (: I1-1), (PAR(N,II),N=1,NPAR)
      WRITE (LOUT, 1920) INAME(I1:)
   ELSE
      WRITE (LOUT, 1900) I1,INAME (: 45), (PAR(N,II),N=1,NPAR)
   ENDIF
C
   IREA C = ' '
   IPROD = ' '
   IF (I1 .GT. 0) THEN
      IREA C = INAME (: I1-1)
      IPROD = INAME (I2+1:)
   ELSE
   ENDIF
C
   IF (INDEX(IREAC,'=>').GT.0 .OR. INDEX(IPROD,'=>').GT.0) THEN
   C----- more than one '=>'
      WRITE (LOUT, 800)
      KERR = .TRUE.
      RETURN
   ENDIF
C
   C----- Is this a fall-off reaction?
   }
C
IF (INDEX(IREAC,'('+') .GT.0 .OR. INDEX(IPROD,'('+') .GT.0) THEN
  KRTB = 0
  KPTB = 0
DO 300 J = 1, 2
  ISTR = ' '
  KTB = 0
  IF (J .EQ. 1) THEN
    ISTR = IREAC
  ELSE
    ISTR = IPROD
  ENDIF
DO 35 N = 1, ILASCH(ISTR)-1
  IF (ISTR(N:N+1) .EQ. ' (+') THEN
    II = N+2
    I2 = II + INDEX(ISTR(II:],')')-1
    IF (ISTR(I2:I2+1) .EQ. 'M' .OR.
        ISTR(I2:I2+1) .EQ. 'm') THEN
      IF (KTB .NE. 0) THEN
        WRITE (LOUT, 630)
        KERR = .TRUE.
        RETURN
      ELSE
        KTB = -1
      ENDIF
    ELSE
      CALL CKCOMP (ISTR(I2+1), KNAME, KK, KNUM)
      IF (KNUM .GT. 0) THEN
        IF (KTB .NE. 0) THEN
          WRITE (LOUT, 630)
          KERR = .TRUE.
          RETURN
        ELSE
          KTB = KNUM
        ENDIF
      ENDIF
      IF (KTB .NE. 0) THEN
        ITEMP = ' '
        IF (I1 .EQ. 1) THEN
          ITEMP = ISTR(I2+1:)
        ELSE
          ITEMP = ISTR(I2-I:)/ISTR(I2+1:)
        ENDIF
        IF (J .EQ. 1) THEN
          IREAC = ' '
          IREAC = ITEMP
          KRTB = KTB
        ELSE
          IPROD = ' '
          IPROD = ITEMP
          KPTB = KTB
        ENDIF
      ENDIF
    ENDIF
  ENDIF
35 CONTINUE
300 CONTINUE
C
NFAL = NFAL + 1
IFAL(NFAL) = II
KFAL(NFAL) = 0
C
LTHB = .TRUE.
NTHB = NTHB + 1
ITHB(NTHB) = II
C
ELSEIF (KRTB .EQ. KFTB) THEN
NFAL = NFAL + 1
IFAL(NFAL) = II
KFAL(NFAL) = KRTB
C
ELSE
WRITE (LOUT, 640)
KERR = .TRUE.
RETURN
ENDIF
ENDIF
ENDIF
C
C-----------------Find reactants, products-----------------------
C
DO 600 J = 1, 2
ISTR = ' ';
LTHB = .FALSE.
IF (J .EQ. 1) THEN
ISTR = IREA
NS = 0
ELSE
ISTR = IPROD
NS = 3
ENDIF
C
C-------------------store pointers to '+'-signs
C
NPLUS = 1
IPLUS(NPLUS) = 0
DO 500 L = 2, ILASCH(ISTR)-1
IF (ISTR(L:L).EQ. '+') THEN
NPLUS = NPLUS + 1
IPLUS(NPLUS) = L
ENDIF
500 CONTINUE
NPLUS = NPLUS + 1
IPLUS(NPLUS) = ILASCH(ISTR)+1
C
NSTART = 1
505 CONTINUE
N1 = NSTART
DO 510 N = NPLUS, N1, -1
ISPEC = ' '
ISPEC = ISTR(IPLUS(N1)+1 : IPLUS(N)-1)
C
IF (UPCASE(ISPEC, 1) .EQ. 'M') THEN
IF (LTHB) THEN
WRITE (LOUT, 900)
KERR = .TRUE.
RETURN
ELSEIF (NFAL.GT.0 .AND. IFAL(NFAL).EQ. II) THEN
WRITE (LOUT, 640)
KERR = .TRUE.
RETURN
ELSE
LTHB = .TRUE.
1 IF (NTHB.EQ.0 .OR. (NTHB.GT.0 .AND. ITHB(NTHB).NE.II)) THEN
   NTHB = NTHB + 1
   ITHB(NTHB) = II
ENDIF
IF (N .EQ. NPLUS) GO TO 600
NSTART = N
GO TO 505
ENDIF
C
ELSEIF (UPCASE(ISPEC, 2) .EQ. 'HV') THEN
   IF (LWL) THEN
      WRITE (LOUT, 670)
      KERR = .TRUE.
      RETURN
   ELSE
      LWL = .TRUE.
      NWL = NWL + 1
      IWL(NWL) = II
      WL(NWL) = 1.0
      IF (J .EQ. 1) WL(NWL) = -1.0
      IF (N .EQ. NPLUS) GO TO 600
      NSTART = N
      GO TO 505
   ENDIF
ENDIF
C
C- - - - - - - - - - - - - - - - - - - - - does this string start with a one- or two-digit number?
C
IVAL = 0
CALL CKCOMP (ISPEC(1:1), INUM, 10, II)
CALL CKCOMP (ISPEC(2:2), INUM, 10, I2)
IF (II .GT. 0) THEN
  ITEMP = ''
   IF (I2 .GT. 0) THEN
      IVAL = 10*(II-1) + (I2-1)
      ITEMP = ISPEC(3:)
   ELSE
      IVAL = II-1
      ITEMP = ISPEC(2:)
   ENDIF
   ISPEC = ITEMP
ENDIF
C
CALL CKCOMP (ISPEC, NAME, KK, KNUM)
IF (KNUM .EQ. 0) THEN
   IF ((N-N1) .GT. 1) GO TO 510
   WRITE (LOUT, 680) ISPEC(:,ILASCH(ISPEC))
   KERR = .TRUE.
ELSE
C
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - a species has been found
C
IVAL = MAX(IVAL,1)
IF (J .EQ. 1) IVAL = -IVAL
NNUM = 0
DO 111 K = 1, NS
   IF (KNUM.EQ.NUNK(K,II) .AND. 
      NU(K,II)/IVAL.GT.0) THEN
      NNUM=K
      NU(NNUM,II) = NU(NNUM,II) + IVAL
111   CONTINUE
C
C- - - - - - - - - - - - - - - - - - - - - - - - - - - increment species coefficient count
C
C
ENDIF
IF (NNUM .LE. 0) THEN

C----------------------are there too many species?
C
IF (J.EQ.1 .AND. NS.EQ.3) THEN
WRITE (LOUT, 690)
KERR = .TRUE.
RETURN
ELSEIF (J.EQ.2 .AND. NS.EQ.MAXSP) THEN
WRITE (LOUT, 700)
KERR = .TRUE.
RETURN
ELSE

C----------------------increment species count
C
NS = NS + 1
NSPEC(II) = NSPEC(II)+1
IF (J .EQ. 1) NREAC(II) = N
NUNK(NS,II) = KNUM
NU(NS,II) = IVAL
ENDIF
ENDIF
ENDIF
IF (N .EQ. NPLUS) GO TO 600
NSTART = N
GO TO 505

C 510 CONTINUE
600 CONTINUE
C
NSPEC(II) = IR*NSPEC(II)
C
630 FORMAT (6X,'Error...more than one fall-off declaration...')
640 FORMAT (6X,'Error in fall-off declaration...')
650 FORMAT (6X,'Error...reaction string not found...')
660 FORMAT (6X,'Error in reaction...')
670 FORMAT (6X,'Error in HV declaration...')
680 FORMAT (6X,'Error...undeclared species...','A')
690 FORMAT (6X,'Error...more than 3 reactants...')
700 FORMAT (6X,'Error...more than 3 products...')
800 FORMAT (6X,'Error in reaction delimiter...')
900 FORMAT (6X,'Error in third-body declaration...')
C 1900 FORMAT (14,'.',A,T51,E10.3,F7.3,F11.3)
1900 FORMAT (14,'.',A,T53,1PE8.2,2X,0PF5.1,2X,F9.1)
1920 FORMAT (6X,A)
RETURN
END

------------------------------------------------------------------
SUBROUTINE CKAUXL (SUB, NSUB, II, KK, KNAME, LOUT, MAXSP, NPAR, 1  NSPEC, NTHB, ITHB, NTBS, MAXTB, NKTB, AIN, 2  NFAI, IPAL, IIDUP, NPAR, PFAL, IFOP, KFAL, NLAN, 3  ILAN, NLAN, PLAN, NREV, IREV, RPAR, 4  NRLT, IRLT, RLAN, NWL, IWL, WL, KERR)
C
CKAUXL parses the auxiliary CHAR*(*) lines representing
additional options for a gas-phase reaction; data is stored
based on finding a 'keyword' followed by its required
parameters:

KNAME(K)/val1/: this is an enhanced third-body;
if ITHB(NTHB) <> I, this is an error, reaction I is not a
third-body reaction;
else NTBS(NTHB) is incremented,
   AIK(NTBS(NTHB),NTHB) = K,
   NKTB(NTBS(NTHB)),NTHB) = val1;

START (LOW, TROE, and SRI define fall-off data):

   LOW/val1 val2 val3/;  PFAL(N,NFAL) = val(N), N=1,3;
     if IFAL(NFAL)<1, this is an error, reaction I is not a
        fall-off reaction;
     if ILAN(NLAN)=1, this is an error, cannot have T-L numbers.
     if IRLT(NRLT)=1, this is an error, "
     if IREV(NREV)=1, this is an error, cannot declare reverse
        parameters;
     if IFOP(NFAL)>0, this is an error, LOW already declared;
     else
       IFOP(NFAL) = ABS(IFOP(NFAL))
   TROE/val1 val2 val3 [val4]/;  PFAL(N,NFAL) = val(N), N=4,7;
     if IFAL(NFAL)<1, this is an error, reaction I is not a
        fall-off reaction;
     if ILAN(NLAN)=1, this is an error, cannot have T-L numbers.
     if IRLT(NRLT)=1, this is an error, "
     if IREV(NREV)=1, this is an error, cannot declare reverse
        parameters;
     if ABS(IFOP(NFAL)).GT.1, this is an error,
     else
       if 3 TROE values, IFOP(NFAL) = 3*IFOP(NFAL);
       if 4 TROE values, IFOP(NFAL) = 4*IFOP(NFAL);
   SRI/val1 val2 val3/;  PFAL(N,NFAL) = val(N), N=4,6;
     if IFAL(NFAL)<1, this is an error, reaction I is not a
        fall-off reaction;
     if ILAN(NLAN)=1, this is an error, cannot have T-L numbers.
     if IRLT(NRLT)=1, this is an error, "
     if IREV(NREV)=1, this is an error, cannot declare reverse
        parameters;
     if ABS(IFOP(NFAL)).GT.1, this is an error;
     else
       if IFOP(NFAL)= 2*IFOP(NFAL);
   LT/val1 val2/:
     if IFAL(NFAL)=1, this is an error, cannot have fall-off and
        T-L numbers;
     else increment NLAN, the number of T-L reactions,
       ILAN(NLAN)=1, PLAN(N,NLAN)=val(N), N=1,2
     if IREV(NREV)=1, need IRLT(NRLT)=1.
   REV[ERSE]/val1 val2 val3/ :
     if IFAL(NFAL)=1, this is an error;
     if IREV(NREV)=1, this is an error, REV already declared;
     if NSPEC(I)<0, this an error, as I is irreversible;
     else increment NREV, the number of reactions with reverse
        parameters given,
       IREV(NREV)=1, RPAR(N,NREV)=val(N), N=1,3;
     if ILAN(NLAN)=1, need IRLT(NRLT)=1;
     if IRLT(NRLT)=1, need ILAN(NRLT)=1.
   RLT/val1 val2/:
     if IFAL(NFAL)=1, this is an error, cannot have fall-off and
        T-L numbers;
     if IRLT(NRLT)=1, this is an error, RLT already declared;
     else increment NRLT, the number of reactions with BOTH
        reverse parameters given, and T-L numbers;
There is a document that needs to be converted into a plain text representation. The document appears to be a computer program or a part of a computer program, written in a scientific or technical context, possibly related to chemistry or physics. The text includes various comments, variables, and functions, with a focus on input and output parameters, as well as some logical conditions and loops.

### Converted Text

```
C IRLT(NRLT)=I, RLAN(N,NRLT)=val(N),N=1,2;
C if IREV(NREV)<>I, need IREV(NREV)=I;
C if ILAN(NREV)<>I, need ILAN(NLAN)=1;

C DUPLICATE:
C This reaction is allowed to be duplicated.

C Input: LINE - CHAR(^*) auxiliary information string
C KK - total number of species declared
C KNAME - CHAR(*^*) species names
C LOUT - output unit for error messages
C MAXSP - maximum third bodies allowed in a reaction

C Output: NTHB - total number of reactions with third bodies
C ITHB - array of third-body reaction numbers
C AIK - non-zero third body enhancement factors
C NKTB - array of species numbers for the third body
C enhancement factors
C NFAL - total number of fall-off reactions
C IFAL - array of fall-off reaction numbers
C IFOP - array of fall-off type
C PFAL - fall-off parameters
C NLAR - total number of Landau-Teller reactions
C ILAN - array of T-L reaction numbers
C NRLT - number of Landau-Teller numbers allowed
C PLAN - array of Landau-Teller numbers
C PLAN - array of Landau-Teller numbers
C NLA - total number of 'reverse' T-L reactions
C IRLT - array of 'reverse' T-L reaction numbers
C IREV - array of 'reverse' Landau-Teller numbers
C RLAN - array of 'reverse' Landau-Teller numbers
C IWL - array of radiation-enhanced reaction numbers
C WL - array of wavelengths
C KERR - logical, = .TRUE. if error found
C F. Rupley, Div. 8245, 5/27/87

C*****precision > double
C IMPLICIT DOUBLE PRECISION (A-H,0-Z), INTEGER (I-N)
C*****END precision > double
C*****precision > single
C IMPLICIT REAL (A-H,0-Z), INTEGER (I-N)
C*****END precision > single

DIMENSION NSPEC(*), ITHB(*), NTBS(*), NKTB(MAXTB,*), IDUP(*),
1 AIK(MAXTB,*), IFAL(*), IFOP(*), PFAL(NFAR,*),
2 ILAN(*), PLAN(NLAR,*), IREV(*), RPAR(NPAR,*), IRLT(*),
3 RLAN(NLAR,*), IWL(*), WL(*), VAL(l)
CHARACTER SUB(*^*)*, KNAME(*^*)*, KEY*80, RSTR*80, UPCASE*4
CHARACTER SUB_CH (80)*80,RSTR_CH*80,LINE_CH*80
LOGICAL KERR, ILAN, LRLT, ITHB, IFAL, IREV, LSRI, LREV

LTHB = (NTHB.GT.0 .AND. ITHB(NTHB).EQ.II)
LFAL = (NFAL.GT.0 .AND. IFAL(NFAL).EQ.II)
LWL = (NWL.GT.0 .AND. IWL(NWL).EQ.II)
LREV = (NREV.GT.0 .AND. IREV(NREV).EQ.II)
ILLAN = (NLAR.GT.0 .AND. ILAN(NLAR).EQ.II)
LRLT = (NRLT.GT.0 .AND. IRLT(NRLT).EQ.II)
ILTR = (NFA.GT.0 .AND. IFAL(NFA).EQ.II .AND. IFOP(NFA).GT.2)
LSRI = (NFOP.GT.0 .AND. IFOP(NFOP).EQ.II .AND. IFOP(NFOP).GT.2)

DO 500 N = 1, NSUB
  ILEN = ILASCH(SUB(N))
  KEY = ' ' 
  IF ( UPCASE(SUB(N), 3) .EQ. 'DUP') THEN
    IDUP(I) = -1
    WRITE (LOUT, 4000)
  GO TO 500
ELSE
```

This text contains a mix of comments, variable declarations, and logical conditions that are typical of a computer program. The comments explain the purpose of certain parts of the code, and the variables and functions are designed to process information related to chemical reactions or physical systems.
I1 = INDEX(SUB(N),'/')
I2 = INDEX(SUB(N)(I1+1:),'/')
IF (I1.LE.0 .OR. I2.LE.0) THEN
  KERR = .TRUE.
  WRITE (LOUT, 2090) SUB(N)(:ILEN)
  GO TO 500
ENDIF
KEY = SUB(N)(:I1-1)
RSTR = '
RSTR = SUB(N)(I1+1:I1+I2-1)
ENDIF
C **************WING 6/16/95**************************
IF (UPCASE(KEY, 3).EQ.'LOW' .OR.
  UPCODE(KEY, 4).EQ.'TROE'.OR.
  UPCODE(KEY, 4).EQ.'EXTR'.OR.
  UPCODE(KEY, 4).EQ.'CHEB'.OR.
  UPCODE(KEY, 3).EQ.'SRI') THEN
  C
  C FALL-OFF DATA
  C
  IF ( (.NOT. LFAL) .OR. LLAN .OR. LRLT .OR. LREV ) THEN
    KERR = .TRUE.
    IF ( .NOT. LFAL) WRITE (LOUT, 1050) SUB(N)(:ILEN)
    IF ( LLAN ) WRITE (LOUT, 1060) SUB(N)(:ILEN)
    IF ( LRLT) WRITE (LOUT, 1070) SUB(N)(:ILEN)
    IF ( LREV ) WRITE (LOUT, 1090) SUB(N)(:ILEN)
  ELSE
    IF ( UPCODE(KEY, 3) .EQ. 'LOW') THEN
      IF ( IFOP(NFAL) .GT. 0) THEN
        WRITE (LOUT, 2000) SUB(N)(:ILEN)
        KERR = .TRUE.
      ELSE
        IFOP(NFAL) = ABS(IFOP(NFAL))
        CALL IPPARR (RSTR, 1,3,PFAL(1,NFAL),NVAL,IER,LOUT)
        IF ( IER .NE. 0) KERR = .TRUE.
        WRITE (LOUT, 3050) (PFAL(L,NFAL), L=1,3)
      ENDIF
    ELSEIF ( UPCODE(KEY, 4) .EQ. 'TROE') THEN
      IF ( LTR0 .OR. LSRI) THEN
        KERR = .TRUE.
        IF ( LTR0) WRITE (LOUT, 2010) SUB(N)(:ILEN)
        IF ( LSRI) WRITE (LOUT, 2030) SUB(N)(:ILEN)
      ELSE
        LTR0 = .TRUE.
        CALL IPPARR (RSTR, 1,-4,PFAL(4,NFAL),NVAL,IER,LOUT)
        IF ( NVAL .EQ. 3) THEN
          IFOP(NFAL) = 3*IFOP(NFAL)
          WRITE (LOUT, 3080) (PFAL(L,NFAL), L=4,6)
        ELSEIF ( NVAL .EQ. 4) THEN
          IFOP(NFAL) = 4*IFOP(NFAL)
          WRITE (LOUT, 3090) (PFAL(L,NFAL), L=4,7)
        ELSE
          WRITE (LOUT, 2020) SUB(N)(:ILEN)
          KERR = .TRUE.
        ENDIF
      ENDIF
    ENDIF
  ENDIF
C **************WING 8/10/95**************************
ELSEIF ( UPCODE(KEY, 4) .EQ. 'CHEB') THEN
  C
  LTRO = .TRUE.
  CALL IPPARR (RSTR, 1,-8,PFAL(4,NFAL),NVAL,IER,LOUT)
  N_CHEB = PFAL(4,NFAL)
  M_CHEB = PFAL(5,NFAL)
  NFARA_CHEB = N_CHEB*M_CHEB + 2
INDEX_CHEB = 0
IPOP(NFAL) = 6*IPOP(NFAL)
c write(*,*)'..NFAL...IPOP...',NFAL,IPOP(NFAL)
c WRITE(*,’..NPARA_CHEX.’)
c CALL IPPARR(RSTR,1,-5,PFAL(4,NFAL),NVAL,IER,LOUT)
INDEX_CHEB = INDEX_CHEB + NVAL
c WRITE (LOUT, 2026) SUB_CH(N):ILEN
C WRITE(*,*)'..NPARA..INDEX..NVAL..NPARA_CHEB,NVAL
C WRITE(*,*)'..N..M..',N_CHEB,M_CHEB
WRITE (LOUT, 3098) (PFAL(L,NFAL),L=4,NVAL+3)
101 CONTINUE
LINE_CH = '
READ(5,‘(A)’) LINE_CH
c WRITE(*,*)LINE_CH
C WRITE(*,*)LINE_CH
C
ILEN = IPPLEN(LINE_CH)
CALL CKISUB(LINE_CH(:ILEN),SUB_CH,NSUB_CH)
C KEY =
DO 151 NSUB_CH = 1, NSUB_CH
IILEN = ILASCH(SUB_CH(NSUB_CH))
c WRITE(*,*)SUB_CH = ',NSUB_CH
C WRITE(*,*)SUB_CH(NSUB_CH)
II1 = INDEX(SUB_CH(NSUB_CH),'/')
II2 = INDEX(SUB_CH(NSUB_CH)(II1+1:),'/')
c WRITE(*,*)'...II1...II2...',II1,II2
IF (II1.LE.0 .OR. II2.LE.0) THEN
  KERR = .TRUE.
  WRITE (LOUT, 2090) SUB_CH(N_CH)(:IILEN)
GO TO 199
ENDIF
C KEY = SUB(N)(:II-1)
RSTR_CH = '
RSTR_CH = SUB_CH(NSUB_CH)(II1+1:II1+II2-1)
c WRITE(*,*)RSTR_CH
C CALL IPPARR(RSTR_CH,1,-8,PFAL(INDEX_CHEB+4,NFAL),
           ,NVAL,IER,LOUT)
c WRITE(*,*)'...INDEX..NVAL..',INDEX_CHEB,NVAL
C WRITE(*,*)'...IER..=',IER
C INDEX_CHEB = INDEX_CHEB + NVAL
WRITE (*,3098) (PFAL(L,NFAL),L=INDEX_CHEB+4
,NVAL+INDEX_CHEB+3)
2 WRITE (LOUT, 3098) (PFAL(L,NFAL),L=INDEX_CHEB+4
,NVAL+INDEX_CHEB+3)
C INDEX_CHEB = INDEX_CHEB + NVAL
C WRITE(*,*)'...INDEX..',INDEX_CHEB
151 CONTINUE
IF (INDEX_CHEB.LT.NPARA_CHEB) GOTO 101
199 CONTINUE
IF (INDEX_CHEB .EQ. NPARA_CHEB) THEN
ELSE
WRITE (LOUT, 2028) SUB_CH(N):ILEN
KERR = .TRUE.
ENDIF
C **********************************************************
C **********************************************************
ELSEIF (UPCASE(KEY, 4) .EQ. 'EXTR') THEN
IF (LTRO .OR. LSRI) THEN
  KERR = .TRUE.
  IF (LTRO) WRITE (LOUT, 2015) SUB(N):ILEN
  IF (LSRI) WRITE (LOUT, 2035) SUB(N):ILEN
ELSE
  LTRO = .TRUE.
  CALL IPPARR(RSTR,1,-5,PFAL(4,NFAL),NVAL,IER,LOUT)
  IF (NVAL .EQ. 5) THEN

IFOP(NFAL) = 5*IFOP(NFAL)
WRITE(*,*)IFOP(NFAL)
WRITE (LOUT, 3095) (PFAL(L,NFAL),L=4,8)
ELSE
WRITE (LOUT, 2025) SUB(N)(:ILEN)
KERR = .TRUE.
ENDIF
ENDIF
ELSEIF (UPCASE(KEY, 3) .EQ. 'SRI') THEN
IF (LTRO .OR. LSRI) THEN
KERR = .TRUE.
IF (LTRO) WRITE (LOUT, 2030) SUB(N)(:ILEN)
IF (LSRI) WRITE (LOUT, 2040) SUB(N)(:ILEN)
ELSE
LSRI = .TRUE.
IFOP(NFAL) = 2*IFOP(NFAL)
CALL IPPARR (RSTR,1,-5,PFAL(4,NFAL),NVAL,IER,LOUT)
IF (NVAL .EQ. 3) THEN
PFAL(7,NFAL) = 1.0
PFAL(8,NFAL) = 0.0
WRITE (LOUT, 3060) (PFAL(L,NFAL),L=4,6)
ELSEIF (NVAL .EQ. 5) THEN
WRITE (LOUT, 3070) (PFAL(L,NFAL),L=4,8)
ELSE
WRITE (LOUT, 2020) SUB(N)(:ILEN)
KERR = .TRUE.
ENDIF
ENDIF
ENDIF
ENDIF
ELSEIF (UPCASE(KEY, 3) .EQ. 'REV') THEN
REVERSE ARRHENIUS PARAMETERS
IF (LFAL .OR. LREV .OR. NSPEC(II).LT.0) THEN
KERR = .TRUE.
IF (LFAL) WRITE (LOUT, 1090) SUB(N)(:ILEN)
IF (LREV) WRITE (LOUT, 2050) SUB(N)(:ILEN)
IF (NSPEC(II) .LT. 0) WRITE (LOUT, 2060) SUB(N)(:ILEN)
ELSE
LREV = .TRUE.
NREV = NREV+1
IREV(NREV) = II
CALL IPPARR (RSTR,1,NPAR,RPAR(1,NREV),NVAL,IER,LOUT)
IF (IER .NE. 0) KERR = .TRUE.
WRITE (LOUT, 1900) ' Reverse Arrhenius coefficients:',
(RPAR(L,NREV),L=1,3)
ENDIF
ELSEIF (UPCASE(KEY, 3) .EQ. 'RLT') THEN
REVERSE LANDAU-TELLER PARAMETERS
IF (LFAL .OR. LRLT .OR. NSPEC(II).LT.0) THEN
KERR = .TRUE.
IF (LFAL) WRITE (LOUT, 1070) SUB(N)(:ILEN)
IF (LRLT) WRITE (LOUT, 2080) SUB(N)(:ILEN)
IF (NSPEC(II) .LT. 0) WRITE (LOUT, 1080) SUB(N)(:ILEN)
ELSE
LRLT = .TRUE.
NRLT = NRLT + 1
IRLT(NRLT) = II
CALL IPPARR (RSTR,1,NLAR,RLAN(1,NRLT),NVAL,IER,LOUT)
IF (IER .NE. 0) KERR = .TRUE.
ENDIF
WRITE (LOUT, 3040) (RLAN(L,NRLT),L=1,2) ENDIF

C ELSEIF (UPCASE(KEY, 2) .EQ. 'HV') THEN

RADIATION WAVELENGTH ENHANCEMENT FACTOR

IF (.NOT.LWL) THEN
WRITE (LOUT, 1000) SUB(N)(:ILEN)
KERR = .TRUE.
ELSE
CALL IPPARR (RSTR,1,1,VAL,NVAL,IER,LOUT)
IF (IER .EQ. 0) THEN
WL(NWL) = WL(NWL)*VAL(1)
WRITE (LOUT, 3020) ABS(WL(NWL))
ELSE
WRITE (LOUT, 1000) SUB(N)(:ILEN)
KERR = .TRUE.
ENDIF
ENDIF

C ELSEIF (UPCASE(KEY, 2) .EQ. 'LT') THEN

LANDAU-TELLER PARAMETERS

IF (LFAL .OR. LLAN) THEN
KERR = .TRUE.
IF (LFAL) WRITE (LOUT, 1060) SUB(N)(:ILEN)
IF (LLAN) WRITE (LOUT, 2070) SUB(N)(:ILEN)
ELSE
LLAN = .TRUE.
NLAN = NLAN + 1
ILAN(NLAN) = II
CALL IPPARR (RSTR,1,NLAR,PLAN(1,NLAN),NVAL,IER,LOUT)
IF (IER .NE. 0) THEN
WRITE (LOUT, 1010) SUB(N)(:ILEN)
KERR = .TRUE.
ENDIF
WRITE (LOUT, 3000) (PLAN(L,NLAN),L=1,2)
ENDIF

C ELSE

ENHANCED THIRD BODIES

CALL CKCOMP (KEY, KNAME, KK, K)
IF (K .EQ. 0) THEN
WRITE (LOUT, 1040) KEY(:ILASCH(KEY))
KERR = .TRUE.
ELSE
IF (.NOT.LTHB) THEN
KERR = .TRUE.
WRITE (LOUT, 1020) SUB(N)(:ILEN)
ELSE
IF (NTBS(NTHB) .EQ. MAXTB) THEN
KERR = .TRUE.
WRITE (LOUT, 1030) SUB(N)(:ILEN)
ELSE
CALL IPPARR (RSTR,1,1,VAL,NVAL,IER,LOUT)
IF (IER .EQ. 0) THEN
WRITE (LOUT, 3010) KNAME(K),VAL(1)
NTBS(NTHB) = NTBS(NTHB) + 1
NKTB(NTHB),NTHB) = K
AIK(NTHB,NTHB) = VAL(1)
ELSE
WRITE (LOUT, 1020) SUB(N)(:ILEN)
ENDIF
ENDIF
ENDIF
ENDIF
KERR = .TRUE.
ENDIF
ENDIF
ENDIF
500 CONTINUE

C FORMATS

C 1000 FORMAT (6X,'Error in HV declaration...',A)
1010 FORMAT (6X,'Error in LT declaration...',A)
1020 FORMAT (6X,'Error in third body declaration...',A)
1030 FORMAT (6X,'Error...more than MAXTB third bodies...',A)
1040 FORMAT (6X,'Error...undeclared species...',A)
1050 FORMAT (6X,'Error...this is not a fall-off reaction...',A)
1060 FORMAT (6X,'Error...LT declared in fall-off reaction...',A)
1070 FORMAT (6X,'Error...RLT declared in fall-off reaction...',A)
1080 FORMAT (6X,'Error...RLT declared in irreversible reaction...',A)
1090 FORMAT (6X,'Error...REV declared in fall-off reaction...',A)
2000 FORMAT (6X,'Error...LOW declared more than once...',A)
2010 FORMAT (6X,'Error...TROE declared more than once...',A)
2015 FORMAT (6X,'Error...EXTROE declared more than once...',A)
2020 FORMAT (6X,'Error in fall-off parameters...',A)
2025 FORMAT (6X,'Error in EXTROE fall-off parameters...',A)
2028 FORMAT (6X,'Error in CHEB fall-off parameters...',A)
2030 FORMAT (6X,'Error...cannot use both TROE and SRI...',A)
2035 FORMAT (6X,'Error...cannot use TROE, EXTROE or SRI...',A)
2040 FORMAT (6X,'Error...SRI declared more than once...',A)
2050 FORMAT (6X,'Error...REV declared more than once...',A)
2060 FORMAT (6X,'Error...REV declared for irreversible reaction...',A)
2070 FORMAT (6X,'Error...LT declared more than once...',A)
2080 FORMAT (6X,'Error...RLT declared more than once...',A)
2090 FORMAT (6X,'Error in auxiliary data...',A)
3000 FORMAT (5X,'Landau-Teller parameters: B=',E12.5,', C=',E12.5)
3010 FORMAT (5X,'Landau-Teller parameters: B=',E12.5)
3020 FORMAT (5X,'Radiation wavelength (A): ',F10.2)
3040 FORMAT (5X,'Reverse Landau-Teller parameters: B=',E12.5,  
1 ',' C=',E12.5)
3050 FORMAT (6X,'Low pressure limit: ',3E13.5)
3060 FORMAT (6X,'SRI centering: ',3E13.5)
3070 FORMAT (6X,'SRI centering: ',5E13.5)
3080 FORMAT (6X,'TROE centering: ',3E13.5)
3090 FORMAT (6X,'TROE centering: ',4E13.5)
3095 FORMAT (6X,'EXTROE centering: ',5E13.5)
3098 FORMAT (6X,'CHEB Polynomials: ',8E13.5)
4000 FORMAT (6X,'Declared duplicate reaction...')

END

SUBROUTINE CKPRNT (MDIM, MAXTP, MM, ENAME, KK, KNAME, WTM,  
1 KPHSE, KCHRG, NT, T, TLO, TMID, THI, KNCF,  
2 ITHRM, LOUT, KERR)

C Prints species interpreter output and checks for completeness.

C**********************************************************************
C SUBROUTINE CKPRNT (MDIM, MAXTP, MM, ENAME, KK, KNAME, WTM,  
1 KPHSE, KCHRG, NT, T, TLO, TMID, THI, KNCF,  
2 ITHRM, LOUT, KERR)
C**********************************************************************
C**** End precision > double
C**********************************************************************
C**** End precision > single
C**********************************************************************
C DIMENSION WTM(*), KPHSE(*), KCHRG(*), T(MAXTP,*),  
1 NT(*), KNCF(MDIM,*), IPLUS(10)
LOGICAL KERR, ITHRM(*)
CHARACTER ENAME(*)*(*), KNAME(*)*(*), IPHSE(3)*1, INUM(10)*1
DATA IPHSE/'S','G','L'/
DATA INUM/'0','1','2','3','4','5','6','7','8','9'/

WRITE (LOUT, 400) (ENAME(M), M = 1, MM)
WRITE (LOUT, 300)

DO 100 K = 1, KK
  IF (T(1,K) .LT. 0.0) T(1,K) = TLO
  IF (T(2,K) .LT. 0.0) T(2,K) = TMID
  IF (T(3,K) .LT. 0.0) T(NT(K),K) = THI
  WRITE (LOUT, 500) K, KNAME(K), IPHSE(KPHSE(K)+2), KCHRG(K),
    WTM(K), T(1,K), T(NT(K),K), (KNCF(M, K),M=1,MM)
  IF (T(1,K) .GE. T(NT(K), K)) THEN
    KERR = .TRUE.
    WRITE (LOUT, 240)
  ENDIF
  IF (T(1,K) .GT. T(2,K)) THEN
    KERR = .TRUE.
  ENDIF
  IF (T(NT(K),K) .LT. T(2,K)) THEN
    KERR = .TRUE.
  ENDIF

each species must have thermodynamic data

IF (.NOT. ITHRM(K)) THEN
  KERR = .TRUE.
  WRITE (LOUT, 200)
ENDIF

a species cannot start with a number

CALL CKCOMP (KNAME(K)(: 1), INUM, 10, I)
IF (I .GT. 0) THEN
  KERR = .TRUE.
  WRITE (LOUT, 210)
ENDIF

if '+' sign is used in a species name,
examples of legal species symbols with + are:
  OH(+2), OH(+2), OH+, OH++, OH++, OH(+), OH(++)
examples of illegal species symbols with + are:
  +OH  (symbol starts with a +, this will cause
        confusion in a reaction)
  OH(+OH) (symbol in parentheses is another species-
           this arrangement is reserved for a fall-off
           reaction)
  OH+OH (plus delimits other species names, this
         will cause confusion in a reaction)

NPLUS = 0
DO 50 N = 1, ILASCH(KNAME(K))
  IF (KNAME(K)(N:N) .EQ. '+' ) THEN
    NPLUS = NPLUS + 1
    IPLUS(NPLUS) = N
  ENDIF
50 CONTINUE
DO 60 N = 1, NPLUS
11 = IPLUS(N)
IF (II .EQ. 1) THEN
  WRITE (LOUT, 220)
  KERR = .TRUE.
ELSE
  is there another species name in parentheses
  IF (KNAME(K)(II-1:II-1) .EQ. '(') THEN
    II = II + 1
    I2 = II + INDEX(KNAME(K)(II:<)), '))'-1
    IF (I2 .GT. II) THEN
      CALL CKCOMP (KNAME(K)(II:II-1), KNAME, KK, KNUM)
      IF (KNUM .GT. 0) THEN
        WRITE (LOUT, 230)
        KERR = .TRUE.
      ENDIF
    ENDIF
  ENDIF
ENDIF

is there another species name after a +
II = II + 1
IF (N .LT. NPLUS) THEN
  DO 55 L = N+1, NPLUS
    I2 = IPLUS(L)
    IF (I2 .GT. II) THEN
      CALL CKCOMP (KNAME(K)(II:II-1), KNAME, KK, KNUM)
      IF (KNUM .GT. 0) THEN
        WRITE (LOUT, 230)
        KERR = .TRUE.
      ENDIF
    ENDIF
  CONTINUE
ENDIF

55 CONTINUE

ENDIF

I2 = ILASCH(KNAME(K))
IF (I2 .GE. II) THEN
  CALL CKCOMP (KNAME(K)(II:II-1), KNAME, KK, KNUM)
  IF (KNUM .GT. 0) THEN
    WRITE (LOUT, 230)
    KERR = .TRUE.
  ENDIF
ENDIF

60 CONTINUE

100 CONTINUE
  WRITE (LOUT, 300)
  RETURN
C
200 FORMAT (6X, 'Error...no thermodynamic properties for species')
210 FORMAT (6X, 'Error...species starts with a number')
220 FORMAT (6X, 'Error...species starts with a plus')
230 FORMAT (6X, 'Error...illegal + in species name')
240 FORMAT (6X, 'Error...High temperature must be < Low temperature')
250 FORMAT (6X, 'Error...Low temperature must be <= Mid temperature')
260 FORMAT (6X, 'Error...High temperature must be => Mid temperature')
300 FORMAT (1X, 79('-'))
C 2 'MOLAR', 2X, 'TEMPERATURE', 4X, 'ELEMENT COUNT', 1X,
C 3 'CONSIDERED', 7X, 'E', 2X, 'E', 2X, 'WEIGHT', 6X, 'LOW', 5X,
C 4 'HIGH', 3X, 15(A3), /1X, 79 ('-'))
C 500 FORMAT (14, ',', 'A10', 2X, 'A1', 13, 'F11.5', 2(F8.1), 15(I3))
1 /IX,'SPECIES',T24,'S G',T28,'MOLECULAR',T38,'TEMPERATURE',
2 T52,'ELEMENT COUNT',
3 /IX,'CONSIDERED',T24,'E E',T28,'WEIGHT',T38,'LOW',
4 T45,'HIGH',T52,15(A3))
500 FORMAT (IX,13,' ',A16,T24,A1,T26,11,T28,F9.5,T38,F6.1,T45,F6.1,
1 T51,15(I3))
END

SUBROUTINE CPREAC (II, MAXSP, NSPEC, NPAR, PAR, RPAR, AUNITS,
1 EUNITS, NREAC, NUNK, NU, KCHRG, MDIM, MM, KNCF,
2 IDUP, NFAL, IFAL, KFAL, NFAR, PFAL, IFOP, NREV,
3 IREV, NTHB, ITHB, NLAN, ILAN, NRLT, IRLT, KERR,
4 LOUT)
C
C Prints reaction interpreter output and checks for reaction
C balance, duplication, and missing data in 'REV' reactions;
C correct units of Arrhenius parameters
C
C Input: II - the index number of the reaction
C MAXSP - maximum number of species allowed in a reaction
C NSPEC - array of the number of species in the reactions
C NPAR - the number of Arrhenius parameters required
C PAR - matrix of Arrhenius parameters for the reactions
C RPAR - matrix of reverse Arrhenius parameters for the
C reactions which declared them
C AUNITS - character string which describes the input units
C EUNITS - character string which describes the input units
C of A, the pre-exponential factor PAR(1,1)
C NREAC - array of the number of reactants in the reactions
C NUNK - matrix of the species numbers of the reactants
C and products in the reactions
C NU - matrix of the stoichiometric coefficients of the
C reactants and products in the reactions
C KCHRG - array of the electronic charges of the species
C MDIM - the maximum number of elements allowed
C MM - the actual number of elements declared
C KNCF - matrix of elemental composition of the species
C IDUP - array of integer flags to indicate duplicate
C reactions
C NFAL - total number of reactions with fall-off
C IFAL - array of the NFAL reaction numbers
C NPAR - maximum number of fall-off parameters allowed
C PFAL - matrix of fall-off parameters for the NFAL
C reactions
C IFOP - array of integer fall-off types for the NFAL
C reactions
C NREV - total number of reactions with reverse parameters
C IREV - array of the NREV reaction numbers
C NTB - total number of reactions with third-bodies
C ITHB - array of the NTB reaction numbers
C NLAN - total number of reactions with Landauer-Teller
C parameters
C ILAN - array of the NLAN reaction numbers
C NRLT - total number of reactions with reverse
C Landauer-Teller parameters
C IRLT - array of the NRLT reaction numbers
C KERR - logical error flag
C LOUT - unit number for output messages
C
C******************************************************************************
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
DIMENSION NSPEC(*), PAR(NPAR,*), RPAR(NPAR,*), NREAC(*),
1 NUNK(MAXSP,*), NU(MAXSP,*), KCHRG(*), KNCF(MDIM,*),
2 IDUP(*), IFAL(*), KFAL(*), PFAL(NPAR,*), IFOP(*),
3 IREV(*), ITHB(*), ILAN(*), IRLT(*)
CHARACTER(*) AUNITS, EUNITS
LOGICAL IERR,KERR,LREV,LLAN,LRLT

C CALL CKBAL (MAXSP, NUNK(1,II), NU(1,II), MDIM, MM, KCHRG, KNCF,
1 IERR)
C
C IF (IERR) THEN
  KERR = .TRUE.
  WRITE (LOUT, 1060)
ENDIF
C
C CALL CKDUP (II, MAXSP, NSPEC, NREAC, NU, NUNK, NFAL, IFAL, KFAL,
1 ISAME)
C
C IF (ISAME .GT. 0) THEN
  IF (IDUP(ISAME).NE.0 .AND. IDUP(II).NE.0) THEN
    IDUP(ISAME) = ABS(IDUP(ISAME))
    IDUP(II) = ABS(IDUP(II))
  ELSE
    N1 = 0
    N2 = 0
    IF (NTHB .GT. 1) THEN
      DO 150 N = 1, NTHB
        IF (ITHB(N) .EQ. ISAME) N1 = 1
        IF (ITHB(N) .EQ. II) N2 = 1
      150 CONTINUE
    ENDIF
    IF (N1 .EQ. N2) THEN
      KERR = .TRUE.
      WRITE (LOUT, 1050) ISAME
    ENDIF
  ENDIF
ENDIF
C
IF (NFAL.GT.0 .AND. IFAL(NFAL).EQ.II .AND. IFOP(NFAL).LT.0) THEN
  KERR = .TRUE.
  WRITE (LOUT, 1020)
ENDIF
C
LREV = (NREV.GT.0 .AND. IREV(NREV).EQ.II)
LLAN = (NLAN.GT.0 .AND. ILAN(NLAN).EQ.II)
LRLT = (NRLT.GT.0 .AND. IRLT(NRLT).EQ.II)
IF (LREV .AND. LLAN .AND. (.NOT.LRLT)) THEN
  KERR = .TRUE.
  WRITE (LOUT, 1030)
ENDIF
IF (LRLT .AND. (.NOT.LLAN)) THEN
  KERR = .TRUE.
  WRITE (LOUT, 1040)
ENDIF
IF (LRLT .AND. (.NOT.LREV)) THEN
  KERR = .TRUE.
  WRITE (LOUT, 1045)
ENDIF
C
IF (EUNITS.EQ. 'KELV') THEN
  EFAC = 1.0
ELSEIF (EUNITS.EQ. 'CAL/') THEN
  C convert E from cal/mole to Kelvin
EFAC = 1.0 / 1.987
ELSEIF (EUNITS .EQ. 'KCAL') THEN
  convert E from kcal/mole to Kelvin
  EFAC = 1000.0 / 1.987
ELSEIF (EUNITS .EQ. 'JOU') THEN
  convert E from Joules/mole to Kelvin
  EFAC = 1.0 / 8.314
ELSEIF (EUNITS .EQ. 'KJOU') THEN
  convert E from Kjoules/mole to Kelvin
  EFAC = 1000.0 / 8.314
ENDIF
PAR(3,II) = PAR(3,II) * EFAC

IF (NREV.GT.0 .AND. IREV(NREV).EQ.II)
  RPAR(3,NREV) = RPAR(3,NREV) * EFAC
IF (NFAL.GT.0 .AND. IFAL(NFAL).EQ.II)
  PFAL(3,NFAL) = PFAL(3,NFAL) * EFAC

IF (AUNITS .EQ. 'MOLC') THEN
  NSTOR = 0
  NSTOP = 0
  DO 50 N = 1, MAXSP
    IF (NU(N,II) .LT. 0) THEN
      sum of stoichiometric coefficients of reactants
      NSTOR = NSTOR + ABS(NU(N,II))
    ELSEIF (NU(N,II) .GT. 0) THEN
      sum of stoichiometric coefficients of products
      NSTOP = NSTOP + NU(N,II)
    ENDIF
  50 CONTINUE
  AVAG = 6.023E23
  IF (NFAL.GT.0 .AND. IFAL(NFAL).EQ.II) THEN
    fall-off reaction, "(+M)" or "(+species name)" does not
    count except in "LOW" A-factor;
    reverse-rate declarations are not allowed
    IF (NSTOR.GT.0) PAR(1,II) = PAR(1,II) * AVAG** (NSTOR-1)
    NSTOR = NSTOR + 1
    IF (NSTOR.GT.0) PFAL(1,NFAL) = PFAL(1,NFAL)*AVAG** (NSTOR-1)
  ELSEIF (NTHB.GT.0 .AND. ITHB(NTHB).EQ.II) THEN
    third body reaction, "+M" counts as species in
    forward and reverse A-factor conversion
    NSTOR = NSTOR + 1
    NSTOP = NSTOP + 1
    IF (NSTOR.GT.0) PAR(1,II) = PAR(1,II) * AVAG** (NSTOR-1)
    IF (NREV.GT.0 .AND. IREV(NREV).EQ.II .AND. NSTOP.GT.0)
      1
    RPAR(1,NREV) = RPAR(1,NREV) * AVAG** (NSTOR-1)
  ELSE
    not third-body or fall-off reaction, but may have
    reverse rates.
    IF (NSTOR .GT. 0) PAR(1,II) = PAR(1,II) * AVAG** (NSTOR-1)
    IF (NREV.GT.0 .AND. IREV(NREV).EQ.II .AND. NSTOP.GT.0)
      1
    RPAR(1,NREV) = RPAR(1,NREV) * AVAG** (NSTOR-1)
  ENDIF
SUBROUTINE CKBAL (NSPEC, KSPEC, KCOEF, MDIM, MM, KCHRG, KNCF, IERR)

Checks elemental balance of reactants vs. products.
Checks charge balance of reaction.

Input: NSPEC - total number of species in this reaction
KSPEC(N), N=1,NSPEC- array of species numbers in reaction
KCOEF(N) - stoichiometric coefficients of the species
MDIM - maximum number of elements allowed
MM - actual integer number of elements
KCHRG(K) - ionic charge Kth species
KNCF(M,K) - integer elemental composition of Kth species

Output: KERR - logical, = .TRUE. if reaction does not balance

F. Rupley, Div. 8245, 5/13/86

IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)

DIMENSION KSPEC(*) , KCOEF(*) , KNCF(MDIM,*), KCHRG(*)
LOGICAL IERR

IERR = .FALSE.

charge balance

KBAL = 0
DO 50 N = 1, ABS(NSPEC)
   KBAL = KBAL + KCOEF(N)*KCHRG(KSPEC(N))
50 CONTINUE
IF (KBAL .NE. 0) IERR = .TRUE.

element balance

DO 100 M = 1, MM
   MBAL = 0
   DO 80 N = 1, ABS(NSPEC)
      MBAL = MBAL + KCOEF(N) * KNCF(M, KSPEC(N))
80 CONTINUE
IF (MBAL .NE. 0) IERR = .TRUE.
100 CONTINUE
RETURN

END

SUBROUTINE CKDUP (I, MAXSP, NS, NR, NU, NUNK, NFAL, IFAL, KFAL, ISAME)

Checks reaction I against the (I-1) reactions for duplication

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
DIMENSION NS(*), NR(*), NU(MAXSP,*), NUNK(MAXSP,*), IFAL(*),
1 KFAL(*)
C
ISAME = 0
NRI = NR(I)
NPI = ABS(NS(I)) - NR(I)
C
DO 500 J = 1, I-1
C
NRJ = NR(J)
NPJ = ABS(NS(J)) - NR(J)
C
IF (NRJ.EQ.NRI .AND. NPJ.EQ.NPI) THEN
C
NSAME = 0
DO 20 N = 1, MAXSP
1 KI = NUNK(N,I)
NI = NU(N,I)
C
DO 15 L = 1, MAXSP
2 KJ = NUNK(L,J)
NJ = NU(L,J)
C
IF (NJ.NE.0 .AND. KJ.EQ.KI .AND. NJ.EQ.NI)
1 NSAME = NSAME + 1
20 CONTINUE
C
IF (NSAME .EQ. ABS(NS(J))) THEN
C
same products, reactants, coefficients, check fall-off
C third body
C
IF (NFAL.GT.0 .AND. IFAL(NFAL).EQ.I) THEN
DO 22 N = 1, NFAL-1
2 IF (J.EQ.IFAL(N) .AND. KFAL(N).EQ.KFAL(NFAL)) THEN
ISAME = J
RETURN
ENDIF
22 CONTINUE
C
RETURN
ENDIF
C
ISAME = J
RETURN
ENDIF
ENDIF
C
IF (NPI.EQ.NRJ .AND. NPJ.EQ.NRI) THEN
C
NSAME = 0
DO 30 N = 1, MAXSP
1 KI = NUNK(N,I)
NI = NU(N,I)
C
DO 25 L = 1, MAXSP
2 KJ = NUNK(L,J)
NJ = NU(L,J)
C
IF (NJ.NE.0 .AND. KJ.EQ.KI .AND. -NJ.EQ.NI)
1 NSAME = NSAME + 1
25 CONTINUE
30 CONTINUE
IF (NSAME.EQ.ABS(NS(J)) .AND. 
   (NS(J).GT.0 .OR. NS(I).GT.0)) THEN
  same products as J reactants, and vice-versa
  IF (NFAL.GT.0 .AND. IFAL(NFAL).EQ.I) THEN
    DO 32 N = 1, NFAL-1
      IF (J.EQ.IFAL(N) .AND. KFAL(N).EQ.KFAL(NFAL)) THEN
        ISAME = J
        RETURN
      ENDIF
    32 CONTINUE
    RETURN
  ENDIF
  ISAME = J
  RETURN
ENDIF
500 CONTINUE
RETURN
END

SUBROUTINE CKISUB (LINE, SUB, NSUB)
Generates an array of CHAR(*) substrings from a CHAR(*) string,
using blanks or tabs as delimiters
Input: LINE - a CHAR(*) line
Output: SUB - a CHAR(*) array of substrings
NSUB - number of substrings found
A '!' will comment out a line, or remainder of the line.
F. Rupley, Div. 8245, 5/15/86
----------------------------------------------------------C
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
  CHARACTER(*) SUB(*), LINE
  NSUB = 0
  DO 5 N = 1, LEN(LINE)
    IF (ICHAR(LINE(N:N)) .EQ. 9) LINE(N:N) = '  '
  5 CONTINUE
  IF (IPPLEN(LINE) .LE. 0) RETURN
  ILEN = ILASCH(LINE)
  NSTART = IFIRCH(LINE)
  10 CONTINUE
    ISTART = NSTART
    NSUB = NSUB + 1
    SUB(NSUB) = '  '
    DO 100 I = ISTART, ILEN
      ILAST = INDEX(LINE(ISTART:), ' ') - 1
      IF (ILAST .GT. 0) THEN
        ILAST = ISTART + ILAST - 1
      ELSE
        ILAST = ILEN
      ENDIF
SUB(NSUB) = LINE(ISTART:ILAST)
IF (ILAST .EQ. ILEN) RETURN

NSTART = ILAST + IFIRCH(LINE(ILAST+1:))

Does SUB have any slashes?

I1 = INDEX(SUB(NSUB),'/')
IF (I1 .LE. 0) THEN
   IF (LINE(NSTART:NSTART) .NE. '/') GO TO 10
   NEND = NSTART + INDEX(LINE(NSTART+1:),'/')
   IND = INDEX(SUB(NSUB), ' ')
   SUB(NSUB)(IND:) = LINE(NSTART:NEND)
   IF (NEND .EQ. ILEN) RETURN
   NSTART = NEND + IFIRCH(LINE(NEND+1:))
   GO TO 10
ENDIF

Does SUB have 2 slashes?

I2 = INDEX(SUB(NSUB)(I1+1:),'/')
IF (I2 .GT. 0) GO TO 10

NEND = NSTART + INDEX(LINE(NSTART+1:),'/')
IND = INDEX(SUB(NSUB), ' ') + 1
SUB(NSUB)(IND:) = LINE(NSTART:NEND)
IF (NEND .EQ. ILEN) RETURN
NSTART = NEND + IFIRCH(LINE(NEND+1:))
GO TO 10

100 CONTINUE
RETURN
END

C***************************************************************
SUBROUTINE IPNPAR (LINE, NPAR, IPAR, ISTART)
C
C Returns CHAR(*) IPAR substring of CHAR(*) string LINE which
C contains NPAR real parameters
C
C Input: LINE - a CHAR(*) line
C NPAR - number of parameters expected
C Output: IPAR - the substring of parameters only
C ISTART - the starting location of IPAR substring
C A '!' will comment out a line, or remainder of the line.
C
F. Rupley, Div. 8245, 5/14/86
C***************************************************************

CHARACTER*(*) LINE,IPAR

C -----------Find Comment String ( ! signifies comment)
ILEN = IPPLEN(LINE)
ISTART = 0
N = 0
IF (ILEN.GT.0) THEN
   DO 40 I = ILEN, 1, -1
      ISTART = I
      IPAR = ' ' 
      IPAR = LINE(ISTART:IEN)
      IF (LINE(1:1).NE.' ') THEN
IF (I .EQ. 1) RETURN
IF (LINE(I-1:I-1) .EQ. ' ') THEN
    N = N + 1
    IF (N .EQ. NPAR) RETURN
ENDIF
40 CONTINUE
ENDIF
RETURN
END

C----------------------------------------------------------------------------------C
SUBROUTINE IPPARI(STRING, ICARD, NEXPEC, IVAL, NFOUND, IERR, LOUT)
C BEGIN PROLOGUE IPPARI
C REFER TO IPGETI
C DATE WRITTEN 850625 (YYMMDD)
C REVISION DATE 851725 (YYMMDD)
C CATEGORY NO. J3.,J4.,M2.
C KEYWORDS PARSE
C AUTHOR CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Parses integer variables from a character variable. Called
C by IPGETI, the IOPAK routine used for interactive input.
C DESCRIPTION
C
CIPPARI may be used for parsing an input record that contains integer
values, but was read into a character variable instead of directly
into integer variables.

The following benefits are gained by this approach:
- specification of only certain elements of the array is allowed,
  thus letting the others retain default values
- variable numbers of values may be input in a record, up to a
  specified maximum
- control remains with the calling program in case of an input
  error
- diagnostics may be printed by IPPARI to indicate the nature
  of input errors

The contents of STRING on input indicate which elements of IVAL
are to be changed from their entry values, and values to which
they should be changed on exit. Commas and blanks serve as
delimiters, but multiple blanks are treated as a single delimiter.
Thus, an input record such as:
' 1, 2,,40000 , 60'
is interpreted as the following set of instructions by IPGETI:
(1) set IVAL(1) = 1
(2) set IVAL(2) = 2
(3) leave IVAL(3) unchanged
(4) set IVAL(4) = 40000
(5) leave IVAL(5) unchanged
(6) set IVAL(6) = 60

IPPARI will print diagnostics on the default output device, if
desired.

IPPARI is part of IOPAK, and is written in ANSI FORTRAN 77

Examples:

Assume IVAL = (0, 0, 0) and NEXPEC = 3 on entry:

<table>
<thead>
<tr>
<th>input string</th>
<th>IVAL on exit</th>
<th>IERR</th>
<th>NFOUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>' 2, 3, 45 '</td>
<td>(2, 3, 45)</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>' 2, 0, 3 '</td>
<td>(2, 0, 3)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>' 0, 0, 0 '</td>
<td>(0, 0, 0)</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Assume IVAL = (0, 0, 0, 0) and NEXPEC = -4 on entry:

<table>
<thead>
<tr>
<th>input string</th>
<th>IVAL on exit</th>
<th>IERR</th>
<th>NFOUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>'1, 2'</td>
<td>(1, 2)</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>',37 400'</td>
<td>(0, 0, 37, 400)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>'1,,-3,5'</td>
<td>(1, 0, -3, 0)</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

arguments: (I=input, O=output)

STRING (I) - the character string to be parsed.

ICARD (I) - data statement number, and error processing flag
< 0: no error messages printed
> 0: print error messages, but not ICARD
> 0: print error messages, and ICARD

NEXPEC (I) - number of real variables expected to be input. If
< 0, the number is unknown, and any number of values
between 0 and abs(nexpec) may be input. (see NFOUND)

PROMPT (I) - prompting string, character type. A question
mark will be added to form the prompt at the screen.

IVAL (I,O) - the integer value or values to be modified. On entry,
the values are printed as defaults. The formal parameter
corresponding to IVAL must be dimensioned at least NEXPEC
in the calling program if NEXPEC > 1.

NFOUND (0) - the number of real values represented in STRING,
only in the case that there were as many or less than
NEXPEC.

IERR (0) - error flag:
= 0 if no errors found
= 1 syntax errors or illegal values found
= 2 for too few values found (NFOUND < NEXPEC)
= 3 for too many values found (NFOUND > NEXPEC)

REFERENCES (NONE)

ROUTINES CALLED IFIRCH, ILASCH

END PROLOGUE IPPARI

CHARACTER STRING(*), ITEMP*80
DIMENSION IVAL(*)
CHARACTER *8 FMT(14)
LOGICAL OKINCR

FIRST EXECUTABLE STATEMENT IPPARI
IERR = 0
NFOUND = 0
NEXP = IABS(NEXPEC)
IE = ILASCH(STRING)
IF (IE .EQ. 0) GO TO 500
NC = 1
OKINCRA is a flag that indicates it is OK to increment. NFOUND, the index of the array into which the value should be read. It is set false when a space follows an integer value substring, to keep incrementing from occurring if a comma should be encountered before the next value.

OKINCRA = .TRUE.

begin overall loop on characters in string

CONTINUE

IF (STRING(NC:NC) .EQ. ',') THEN
  IF (OKINCRA .OR. NC .EQ. IE) THEN
    NFOUND = NFOUND + 1
  ELSE
    OKINCRA = .TRUE.
  ENDIF
  GO TO 450
ENDIF
IF (STRING(NC:NC) .EQ. ' ') GO TO 450

first good character (non-delimeter) found - now find last good character

IBS = NC
CONTINUE
NC = NC + 1
IF (NC .GT. IE) GO TO 180
IF (STRING(NC:NC) .EQ. ' ') THEN
  OKINCRA = .FALSE.
ELSEIF (STRING(NC:NC) .EQ. ',') THEN
  OKINCRA = .TRUE.
ELSE
  GO TO 160
ENDIF
ENDIF

end of substring found - read value into integer array

CONTINUE
NFOUND = NFOUND + 1
IF (NFOUND .GT. NEXP) THEN
  IERR = 3
  GO TO 500
ENDIF

IES = NC - 1
NCH = IES - IBS + 1
DATA FMT/' (II)', ' (12)', ' (13)', ' (14)', ' (15)',
  ' (16)', ' (17)', ' (18)', ' (19)', ' (110)',
  ' (111)', ' (112)', ' (113)', ' (114)'/
ITEMP = ' 'ITEMP = STRING(IBS:IES)
READ (ITEMP(NCH), FMT(NCH), ERR = 400) IVAL(NFOUND)
GO TO 450
CONTINUE
IERR = 1
GO TO 510
CONTINUE
NC = NC + 1
IF (NC .LE. IE) GO TO 100
CONTINUE
IF (NEXPEC .GT. 0 .AND. NFOUND .LT. NEXP) IERR = 2
CONTINUE
C
IF (IERR .EQ. 0 .OR. ICARD .LT. 0) RETURN
IF (ICARD .NE. 0) WRITE (LOUT, '(A,I3)')
1 ' ! ERROR IN DATA STATEMENT NUMBER', ICARD
IF (IERR .EQ. 1) WRITE (LOUT, '(A)') 'SYNTAX ERROR, OR ILLEGAL VALUE'
IF (IERR .EQ. 2) WRITE (LOUT, '(A,12,A,12)')
1 ' TOO FEW DATA ITEMS. NUMBER FOUND = ', NFOUND,
2 ' NUMBER EXPECTED = ', NEXPEC
IF (IERR .EQ. 3) WRITE (LOUT, '(A,12)')
1 ' TOO MANY DATA ITEMS. NUMBER EXPECTED = ', NEXPEC
END
C
SUBROUTINE IPPARR(STRING, ICARD, NEXPEC, RVAL, NFOUND, IERR, LOUT)
C BEGIN PROLOGUE IPPARR
C REFER TO IPGETR
C DATE WRITTEN 850625 (YYMMDD)
C REVISION DATE 851625 (YYMMDD)
C CATEGORY NO. J3., J4., M2.
C KEYWORDS PARSE
C AUTHOR CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT’L LAB
C PURPOSE Parses real variables from a character variable. Called by IPGETR, the IOPAK routine used for interactive input.
C DESCRIPTION
C-----------------------------------------------------------------------------------------------------------------
C IPPARR may be used for parsing an input record that contains real values, but was read into a character variable instead of directly into real variables.
C The following benefits are gained by this approach:
C - specification of only certain elements of the array is allowed, thus letting the others retain default values
C - variable numbers of values may be input in a record, up to a specified maximum
C - control remains with the calling program in case of an input error
C - diagnostics may be printed by IPPARR to indicate the nature of input errors
C The contents of STRING on input indicate which elements of RVAL are to be changed from their entry values, and values to which they should be changed on exit. Commas and blanks serve as delimiters, but multiple blanks are treated as a single delimiter. Thus, an input record such as:
C ' 1., 2,,4.e-5 , 3.e-6'
C is interpreted as the following set of instructions by IPGETR:
C(1) set RVAL(1) = 1.0
(2) set RVAL(2) = 2.0
(3) leave RVAL(3) unchanged
(4) set RVAL(4) = 4.0E-05
(5) leave RVAL(5) unchanged
(6) set RVAL(6) = 6.0E-06
C IPPARR will print diagnostics on the default output device, if desired.
C IPPARR is part of IOPAK, and is written in ANSI FORTRAN 77
C Examples:
C Assume RVAL = (0., 0., 0.) and NEXPEC = 3 on entry:
C input string    RVAL on exit    IERR   NFOUND
Assume RVAL = (0.,0.,0.,0.) and NEXPEC = -4 on entry:

<table>
<thead>
<tr>
<th>input string</th>
<th>RVAL on exit</th>
<th>IERR</th>
<th>NFOUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>'1.,2.'</td>
<td>(1.0, 2.0)</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>'1.,3 4.0'</td>
<td>(0.0, 0.0, 3.0, 4.0)</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>'1.,3,5.0'</td>
<td>(0.0, 0.0, 3.0, 0.0)</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

arguments: (I=input, O=output)

STRING (I) - the character string to be parsed.

ICARD (I) - data statement number, and error processing flag
< 0 : no error messages printed
= 0 : print error messages, but not ICARD
> 0 : print error messages, and ICARD

NEXPEC (I) - number of real variables expected to be input. If
< 0, the number is unknown, and any number of values
between 0 and abs(nexpec) may be input. (see NFOUND)

PROMPT (I) - prompting string, character type. A question
mark will be added to form the prompt at the screen.

RVAL (I,0) - the real value or values to be modified. On entry,
the values are printed as defaults. The formal parameter
 corresponding to RVAL must be dimensioned at least NEXPEC
in the calling program if NEXPEC > 1.

NFOUND (O) - the number of real values represented in STRING,
only in the case that there were as many or less than
NEXPEC.

IERR (O) - error flag:
= 0 if no errors found
= 1 syntax errors or illegal values found
= 2 for too few values found (NFOUND < NEXPEC)
= 3 for too many values found (NFOUND > NEXPEC)
NEXP = IABS(NEXPEC)
IE = ILASCH(STRING)
IF (IE .EQ. 0) GO TO 500
NC = 1

C--- OKINCR is a flag that indicates it's OK to increment
C--- NFOUND, the index of the array into which the value
C--- should be read. It is set negative when a space follows
C--- a real value substring, to keep incrementing from
C--- occurring if a comma should be encountered before the
C--- next value.
C
OKINCR = .TRUE.
C
C--- begin overall loop on characters in string
C
100 CONTINUE
C
IF (STRING(NC:NC) .EQ. ',') THEN
IF (OKINCR) THEN
NFOUND = NFOUND + 1
ELSE
OKINCR = .TRUE.
ENDIF
GO TO 450
ENDIF
IF (STRING(NC:NC) .EQ. ' ') GO TO 450
C
C--- first good character (non-delimeter) found - now find
C--- last good character
C
IBS = NC
160 CONTINUE
NC = NC + 1
IF (NC .GT. IE) GO TO 180
IF (STRING(NC:NC) .EQ. ' ') THEN
OKINCR = .FALSE.
ELSEIF (STRING(NC:NC) .EQ. ',') THEN
OKINCR = .TRUE.
ELSE
GO TO 160
ENDIF
C
C--- end of substring found - read value into real array
C
180 CONTINUE
NFOUND = NFOUND + 1
IF (NFOUND .GT. NEXP) THEN
IERR = 3
GO TO 500
ENDIF
C
DATA FMT/ (E1.0)’, (E2.0)’, (E3.0)’, (E4.0)’,
1 (E5.0)’, (E6.0)’, (E7.0)’, (E8.0)’, (E9.0)’,
2 (E10.0)’, (E11.0)’, (E12.0)’, (E13.0)’, (E14.0)’,
3 (E15.0)’, (E16.0)’, (E17.0)’, (E18.0)’, (E19.0)’,
4 (E20.0)’, (E21.0)’, (E22.0)’/
IES = NC - 1
NCH = IES - IBS + 1
ITEMP = ‘
ITEMP = STRING(IBS:IES)
READ (ITEMP(1:NCH), FMT(NCH), ERR = 400) RVAL(NFOUND)
GO TO 450
400 CONTINUE
WRITE (LOUT, 555) STRING(IBS:IES)
555 FORMAT (A)
   IERR = 1
   GO TO 510
450 CONTINUE
   NC = NC + 1
   IF (NC .LE. IE) GO TO 100
C
500 CONTINUE
   IF (NEXPEC .GT. 0 .AND. NFOUND .LT. NEXP) IERR = 2
510 CONTINUE
C
   IF (IERR .EQ. 0 .OR. ICARD .LT. 0) RETURN
   IF (ICARD .NE. 0) WRITE (LOUT, '(A,13)')
      1 '!! ERROR IN DATA STATEMENT NUMBER', ICARD
   IF (IERR .EQ. 1)
      1 WRITE (LOUT, '  (A) ' )  'SYNTAX ERROR, OR ILLEGAL VALUE'
   IF (IERR .EQ. 2) WRITE (LOUT, '(A,12, A, 12)')
      1 ' TOO FEW DATA ITEMS. NUMBER FOUND = ', NFOUND,
      2 ' NUMBER EXPECTED = ', NEXPEC
   IF (IERR .EQ. 3) WRITE (LOUT, '(A,12)')
      1 ' TOO MANY DATA ITEMS. NUMBER EXPECTED = ', NEXPEC
END
C
FUNCTION IFIRCH(STRING)
C BEGIN PROLOGUE IFIRCH
C DATE WRITTEN 85062 6
C REVISION DATE 850626
C CATEGORY NO. M 4.
C KEYWORDS CHARACTER STRINGS,SIGNIFICANT CHARACTERS
C AUTHOR CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Determines first significant (non-blank) character
C • in character variable
C DESCRIPTION
C---------------------------------------------------------------------------------
C IFIRCH locates the first non-blank character in a string of
C arbitrary length. If no characters are found, IFIRCH is set = 0.
C When used with the companion routine ILASCH, the length of a string
C can be determined, and/or a concatenated substring containing the
C significant characters produced.
C---------------------------------------------------------------------------------
C IFIRCH Locate first significant character in STRING
C FILENAME IFIRCH
C DATE 850626
C PURPOSE Determine first non-blank character
C ARGUMENT STRING
C RETURN VALUE IFLRCH
C-------------------------------------------------------------------------------
C IFIRCH Locates first non-blank character in STRING
C-------------------------------------------------------------------------------
C IFIRCH Locate first significant character in STRING
C FILENAME IFIRCH
C DATE 850626
C PURPOSE Determine first non-blank character
C ARGUMENT STRING
C RETURN VALUE IFLRCH
C-------------------------------------------------------------------------------
C FUNCTION IFIRCH(STRING)
C BEGIN PROLOGUE IFIRCH
C DATE WRITTEN 85062 6
C REVISION DATE 850626
C CATEGORY NO. M 4.
C KEYWORDS CHARACTER STRINGS,SIGNIFICANT CHARACTERS
C AUTHOR CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Determines first significant (non-blank) character
C • in character variable
C DESCRIPTION
C---------------------------------------------------------------------------------
C IFIRCH locates the first non-blank character in a string of
C arbitrary length. If no characters are found, IFIRCH is set = 0.
C When used with the companion routine ILASCH, the length of a string
C can be determined, and/or a concatenated substring containing the
C significant characters produced.
C---------------------------------------------------------------------------------
C IFIRCH Locate first significant character in STRING
C FILENAME IFIRCH
C DATE 850626
C PURPOSE Determine first non-blank character
C ARGUMENT STRING
C RETURN VALUE IFLRCH
C-------------------------------------------------------------------------------
C IFIRCH Locate first significant character in STRING
C FILENAME IFIRCH
C DATE 850626
C PURPOSE Determine first non-blank character
C ARGUMENT STRING
C RETURN VALUE IFLRCH
C-------------------------------------------------------------------------------
C FUNCTION IFIRCH(STRING)
C BEGIN PROLOGUE IFIRCH
C DATE WRITTEN 85062 6
C REVISION DATE 850626
C CATEGORY NO. M 4.
C KEYWORDS CHARACTER STRINGS,SIGNIFICANT CHARACTERS
C AUTHOR CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Determines first significant (non-blank) character
C • in character variable
C DESCRIPTION
C---------------------------------------------------------------------------------
C IFIRCH locates the first non-blank character in a string of
C arbitrary length. If no characters are found, IFIRCH is set = 0.
C When used with the companion routine ILASCH, the length of a string
C can be determined, and/or a concatenated substring containing the
C significant characters produced.
C---------------------------------------------------------------------------------
C IFIRCH Locate first significant character in STRING
C FILENAME IFIRCH
C DATE 850626
C PURPOSE Determine first non-blank character
C ARGUMENT STRING
C RETURN VALUE IFLRCH
C-------------------------------------------------------------------------------
C IFIRCH Locate first significant character in STRING
C FILENAME IFIRCH
C DATE 850626
C PURPOSE Determine first non-blank character
C ARGUMENT STRING
C RETURN VALUE IFLRCH
C-------------------------------------------------------------------------------
C FUNCTION IFIRCH(STRING)
C BEGIN PROLOGUE IFIRCH
C DATE WRITTEN 85062 6
C REVISION DATE 850626
C CATEGORY NO. M 4.
C KEYWORDS CHARACTER STRINGS,SIGNIFICANT CHARACTERS
C AUTHOR CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Determines first significant (non-blank) character
C • in character variable
C DESCRIPTION
C---------------------------------------------------------------------------------
C IFIRCH locates the first non-blank character in a string of
C arbitrary length. If no characters are found, IFIRCH is set = 0.
C When used with the companion routine ILASCH, the length of a string
C can be determined, and/or a concatenated substring containing the
C significant characters produced.
C---------------------------------------------------------------------------------
C IFIRCH Locate first significant character in STRING
C FILENAME IFIRCH
C DATE 850626
C PURPOSE Determine first non-blank character
C ARGUMENT STRING
C RETURN VALUE IFLRCH
C-------------------------------------------------------------------------------
C IFIRCH Locate first significant character in STRING
C FILENAME IFIRCH
C DATE 850626
C PURPOSE Determine first non-blank character
C ARGUMENT STRING
C RETURN VALUE IFLRCH
C-------------------------------------------------------------------------------
C FUNCTION IFIRCH(STRING)
C BEGIN PROLOGUE IFIRCH
C DATE WRITTEN 85062 6
C REVISION DATE 850626
C CATEGORY NO. M 4.
C KEYWORDS CHARACTER STRINGS,SIGNIFICANT CHARACTERS
C AUTHOR CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Determines first significant (non-blank) character
C • in character variable
C DESCRIPTION
C---------------------------------------------------------------------------------
C IFIRCH locates the first non-blank character in a string of
C arbitrary length. If no characters are found, IFIRCH is set = 0.
C When used with the companion routine ILASCH, the length of a string
C can be determined, and/or a concatenated substring containing the
C significant characters produced.
C---------------------------------------------------------------------------------
C IFIRCH Locate first significant character in STRING
C FILENAME IFIRCH
C DATE 850626
C PURPOSE Determine first non-blank character
C ARGUMENT STRING
C RETURN VALUE IFLRCH
C-------------------------------------------------------------------------------
C IFIRCH Locate first significant character in STRING
C FILENAME IFIRCH
C DATE 850626
C PURPOSE Determine first non-blank character
C ARGUMENT STRING
C RETURN VALUE IFLRCH
C-------------------------------------------------------------------------------
C FUNCTION IFIRCH(STRING)
C BEGIN PROLOGUE IFIRCH
C DATE WRITTEN 85062 6
C REVISION DATE 850626
C CATEGORY NO. M 4.
C KEYWORDS CHARACTER STRINGS,SIGNIFICANT CHARACTERS
C AUTHOR CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Determines first significant (non-blank) character
C • in character variable
C DESCRIPTION
C---------------------------------------------------------------------------------
C IFIRCH locates the first non-blank character in a string of
C arbitrary length. If no characters are found, IFIRCH is set = 0.
C When used with the companion routine ILASCH, the length of a string
C can be determined, and/or a concatenated substring containing the
C significant characters produced.
C---------------------------------------------------------------------------------
C IFIRCH Locate first significant character in STRING
C FILENAME IFIRCH
C DATE 850626
C PURPOSE Determine first non-blank character
C ARGUMENT STRING
C RETURN VALUE IFLRCH
C-------------------------------------------------------------------------------
C IFIRCH Locate first significant character in STRING
C FILENAME IFIRCH
C DATE 850626
C PURPOSE Determine first non-blank character
C ARGUMENT STRING
C RETURN VALUE IFLRCH
C-------------------------------------------------------------------------------
C FUNCTION IFIRCH(STRING)
C BEGIN PROLOGUE IFIRCH
C DATE WRITTEN 85062 6
C REVISION DATE 850626
C CATEGORY NO. M 4.
C KEYWORDS CHARACTER STRINGS,SIGNIFICANT CHARACTERS
C AUTHOR CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Determines first significant (non-blank) character
C • in character variable
C DESCRIPTION
C---------------------------------------------------------------------------------
C IFIRCH locates the first non-blank character in a string of
C arbitrary length. If no characters are found, IFIRCH is set = 0.
C When used with the companion routine ILASCH, the length of a string
C can be determined, and/or a concatenated substring containing the
C significant characters produced.
C---------------------------------------------------------------------------------
C IFIRCH Locate first significant character in STRING
C FILENAME IFIRCH
C DATE 850626
C PURPOSE Determine first non-blank character
C ARGUMENT STRING
C RETURN VALUE IFLRCH
C-------------------------------------------------------------------------------
C IFIRCH Locate first significant character in STRING
C FILENAME IFIRCH
C DATE 850626
C PURPOSE Determine first non-blank character
C ARGUMENT STRING
C RETURN VALUE IFLRCH
C-------------------------------------------------------------------------------
C FUNCTION IFIRCH(STRING)
C BEGIN PROLOGUE IFIRCH
C DATE WRITTEN 85062 6
C REVISION DATE 850626
C CATEGORY NO. M 4.
C KEYWORDS CHARACTER STRINGS,SIGNIFICANT CHARACTERS
C AUTHOR CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Determines first significant (non-blank) character
C • in character variable
C DESCRIPTION
C---------------------------------------------------------------------------------
C IFIRCH locates the first non-blank character in a string of
C arbitrary length. If no characters are found, IFIRCH is set = 0.
C When used with the companion routine ILASCH, the length of a string
C can be determined, and/or a concatenated substring containing the
C significant characters produced.
FUNCTION ILASCH(STRING)
C BEGIN PROLOGUE ILASCH
C DATE WRITTEN  850626
C REVISION DATE  850626
C CATEGORY NO.  M4.
C KEYWORDS  CHARACTER STRINGS,SIGNIFICANT CHARACTERS
C AUTHOR  CLARK,G.L.,GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE  Determines last significant (non-blank) character
           in character variable
C DESCRIPTION
C----------------------------------------------------------------------------------------------------------------
C IFIRCH locates the last non-blank character in a string of
arbitrary length.  If no characters are found, ILASCH is set = 0.
C When used with the companion routine IFIRCH, the length of a string
C can be determined, and/or a concatenated substring containing the
C significant characters produced.
C Note that the FORTRAN intrinsic function LEN returns the length
C of a character string as declared, rather than as filled.  The
C declared length includes leading and trailing blanks, and thus is
C not useful in generating 'significant' substrings.
C----------------------------------------------------------------------------------------------------------------
C REFERENCES  (NONE)
C ROUTINES CALLED  (NONE)
C END PROLOGUE IFIRCH
C*****precision > double
   IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
C*****END precision > double
C
C*****precision > single
   IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
C*****END precision > single
C
CHARACTER*(*) STRING
C
C***FIRST EXECUTABLE STATEMENT ILASCH
   NLOOP = LEN(STRING)
   IF (NLOOP.EQ.0) THEN
      ILASCH = 0
      RETURN
   ENDIF
C
   DO 100 I = NLOOP, 1, -1
      IF (STRING(I:1) .NE. ' ')  GO TO 120
   100 CONTINUE
C
   120 CONTINUE
   ILASCH = I
   END
C
SUBROUTINE CKCOMP (IST, IRAY, II, I)
C
C START PROLOGUE
C
C SUBROUTINE CKCOMP (IST, IRAY, II, I)*
C Returns the index of an element of a reference character
C string array which corresponds to a character string;
C leading and trailing blanks are ignored.
INPUT

IST  - A character string.
      Data type - CHARACTER(*)

IRAY  - An array of character strings;
        dimension IRAY(*) at least II
      Data type - CHARACTER(*)

II  - The length of IRAY.
      Data type - integer scalar.

OUTPUT

I  - The first integer location in IRAY in which IST
      corresponds to IRAY(I); if IST is not also an
      entry in IRAY, I=0.

END PROLOGUE

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
CHARACTER(*) IST, IRAY(*)
C
I = 0
DO 10 N = II, 1, -1
  IS1 = IFIRCH(IST)
  IS2 = ILASCH(IST)
  IR1 = IFIRCH(IRAY(N))
  IR2 = ILASCH(IRAY(N))
  IF ( IS2.GE.IS1 .AND. IS2.GT.0 .AND. 
       IR2.GE.IR1 .AND. IR2.GT.0 .AND. 
       1 IST(IS1:IS2).EQ.IRAY(N)(IR1:IR2) ) I=N 
10 CONTINUE
RETURN
END

C---------------------------------------------------------------
SUBROUTINE CKUNIT (LINE, AUNITS, EUNITS, IUNITS)
C
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
CHARACTER(*) LINE, AUNITS, EUNITS, IUNITS
CHARACTER*4 UPCASE

AUNITS = '  '
EUNITS = '  '
IUNITS = '  '
DO 85 N = 1, ILASCH(LINE)-3
  IND = ILASCH(IUNITS)
  IF (EUNITS .EQ. '  ') THEN
    IF (UPCASE(LINE(N:), 4) .EQ. 'C A L /') THEN
      EUNITS = 'CAL/'
    ELSEIF (UPCASE(LINE(N:), 4) .EQ. '1 K') THEN
      EUNITS = 'E units cal/mole'
    ELSE
      IUNITS(IND:) = ',', E units cal/mole'
    ELSE
  ENDIF
ENDIF
ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KCAL') THEN
EUNITS = 'KCAL'
IF (IUNITS .EQ. ' ') THEN
   IUNITS = 'E units Kcal/mole'
ELSE
   IUNITS(IND:) = ', E units Kcal/mole'
ENDIF
ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'JOUL') THEN
   EUNITS = 'JOUL'
   IF (IUNITS .EQ. ' ') THEN
      IUNITS = 'E units Joules/mole'
   ELSE
      IUNITS(IND:) = ', E units Joules/mole'
   ENDIF
ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KJOU') THEN
   EUNITS = 'KJOU'
   IF (IUNITS .EQ. ' ') THEN
      IUNITS = 'E units Kjoule/mole'
   ELSE
      IUNITS(IND:) = ', E units Kjoule/mole'
   ENDIF
ELSEIF (UPCASE(LINE(N:), 4) .EQ. 'KELV') THEN
   EUNITS = 'KELV'
   IF (IUNITS .EQ. ' ') THEN
      IUNITS = 'E units Kelvins'
   ELSE
      IUNITS(IND:) = ', E units Kelvins'
   ENDIF
ENDIF
ENDIF
ENDIF
IF (AUNITS .EQ. ' ') THEN
   IF (UPCASE(LINE(N:), 4) .EQ. 'MOLE') THEN
      AUNITS = 'MOLE'
      IF (UPCASE(LINE(N+4:), 1) .EQ. 'S') THEN
         AUNITS = 'MOLE'
         IF (IUNITS .EQ. ' ') THEN
            IUNITS = 'A units mole-cm-sec-K'
         ELSE
            IUNITS(IND:) = ', A units mole-cm-sec-K'
         ENDIF
      ELSEIF (UPCASE(LINE(N+4:), 4) .EQ. 'MOLEC') THEN
         AUNITS = 'MOLEC'
         IF (IUNITS .EQ. ' ') THEN
            IUNITS = 'A units molecules'
         ELSE
            IUNITS(IND:) = ', A units molecules'
         ENDIF
      ELSEIF (UPCASE(LINE(N+4:), 4) .EQ. 'CULE') THEN
         AUNITS = 'CULE'
         IF (IUNITS .EQ. ' ') THEN
            IUNITS = 'A units molecules'
         ELSE
            IUNITS(IND:) = ', A units molecules'
         ENDIF
      ENDIF
   ELSE
      AUNITS = 'MOLE'
      IND = ILASCH(IUNITS) + 1
      IF (IND .GT. 1) THEN
         IUNITS(IND:) = ', A units mole-cm-sec-K'
      ELSE
         IUNITS(IND:) = ', A units mole-cm-sec-K'
      ENDIF
   ENDIF
ENDIF
ENDIF
ENDIF
85 CONTINUE
C
   IF (AUNITS .EQ. ' ') THEN
      AUNITS = 'MOLE'
      IND = ILASCH(IUNITS) + 1
      IF (IND .GT. 1) THEN
         IUNITS(IND:) = ', A units mole-cm-sec-K'
      ELSE
         IUNITS(IND:) = ', A units mole-cm-sec-K'
      ENDIF
   ENDIF
C
   IF (EUNITS .EQ. ' ') THEN
      EUNITS = 'CAL/
      IND = ILASCH(IUNITS) + 1
      IF (IND .GT. 1) THEN
         IUNITS(IND:) = ', E units cal/mole'
      ELSE

IUNITS(IND:) = ' E units cal/mole'
ENDIF
ENDIF
C
RETURN
END
C
FUNCTION IPPLEN (LINE)

BEGIN PROLOGUE

FUNCTION IPPLEN (LINE)

Returns the effective length of a character string, i.e., the index of the last character before an exclamation mark (!) indicating a comment.

INPUT

LINE - A character string.

OUTPUT

IPPLEN - The effective length of the character string.

END PROLOGUE

*****precision > double
IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
*****END precision > double

*****precision > single
IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
*****END precision > single

CHARACTER LINE(*)

IN = IFIRCH(LINE)
IF (IN.EQ.0 .OR. LINE(IN:IN).EQ.'!') THEN
  IPPLEN = 0
ELSE
  IN = INDEX(LINE,'!')
  IF (IN .EQ. 0) THEN
    IPPLEN = ILASCH(LINE)
  ELSE
    IPPLEN = ILASCH(LINE(:IN-1))
  ENDIF
ENDIF
RETURN
END

CHARACTER*(*) FUNCTION UPCASE(ISTR, ILEN)

CHARACTER ISTR(*), LCASE(26)*1, UCASE(26)*1
DATA LCASE /'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'/,

UPCASE = ' ' 
UPCASE = ISTR(:ILEN)
JJ = MIN (LEN(UPCASE), LEN(ISTR), ILEN)
DO 10 J = 1, JJ
  DO 10 N = 1, 26
    IF (ISTR(J:J) .EQ. LCASE(N)) UPCASE(J:J) = UCASE(N)
10 CONTINUE
RETURN
END
SUBROUTINE FILEIO_CVT

IMPLICIT DOUBLE PRECISION (A-H,O-Z)
PARAMETER (LIN=5, LOUT=6, LOG=15, LINC=25)
CHARACTER FILE_IN*20, FILE_OUT*20
COMMON/FILE_IO/FILE_IN, FILE_OUT

5 WRITE(*,10)
10 FORMAT(' Please enter the MASTER mechanism file name : ')
READ(*,'(A20)') FILE_IN
OPEN(UNIT=5, FILE=FILE_IN, STATUS='OLD', ERR=30)
GO TO 50
30 CONTINUE
WRITE(*,40) FILE_IN
40 FORMAT(' Can not find file : ',A20)
GO TO 5
50 CONTINUE
105 WRITE(*,110)
110 FORMAT(' Please enter the output mechanism file name : ')
READ(*,'(A20)') FILE_OUT
OPEN(UNIT=6, FILE=FILE_OUT, STATUS='UNKNOWN', FORM='FORMATTED',
2 ERR=130)
GO TO 150
130 CONTINUE
WRITE(*,140) FILE_OUT
140 FORMAT(' Can not find file : ',A20)
GO TO 105
150 CONTINUE
C OPEN(UNIT=6, FILE='CKINTERP.OUT', STATUS='UNKNOWN')
C OPEN(UNIT=25, FILE='CHEMKIN.LNK', STATUS='UNKNOWN',
C 1 FORM='BINARY')

RETURN
END
D.7 CHEBYSHEV POLY FITTINGS

SUBROUTINE file_IO(Error)
C
IMPLICIT NONE
CHARACTER FILE_INP*20, FILE_LOG*20, FILE_COL*20,
1 FILE_KIN*20, FILE_OUT*20, FILE_NAME*20, FILE_FIT*20
INTEGER INP, LOG, COL, KIN, OUT, Istart, Idot, Ifirch, FIT
logical Error
C
PARAMETER (MG_file=20, Max_Specy_Group=50,
1 Max_Element=4)
PARAMETER (INP=10, LOG=20, COL=30, KIN=55, FIT=95, OUT=99)
COMMON/FILE/FILE_INP, FILE_LOG, FILE_COL, FILE_KIN, FILE_OUT, FILE_NAME
1 , FILE_FIT
C
INCLUDE 'THM_PAR.FI'
C
CHARACTER FILE_NAME*20
INTEGER IDOT, ISTART, IFIRCH
C
ERROR = .FALSE.
5 WRITE(*,10)
10 FORMAT( ' enter the input file name : ' )
READ(*,'(A20)')FILE_INP
OPEN(INP,FILE=FILE_INP,STATUS='OLD',ERR=30)
GO TO 50
30 CONTINUE
WRITE(*,40)FILE_INP
40 FORMAT( ' Can not find file : ',A20)
GO TO 5
50 CONTINUE
ISTART = IFIRCH(FILE_INP)
IDOT = INDEX(FILE_INP,'.')
FILE_NAME = FILE_INP(ISTART:IDOT-1)
C
FILE_LOG = FILE_NAME
FILE_LOG(IDOT:IDOT) = '.
FILE_LOG(IDOT+1:IDOT+3) = 'LOG'
C
FILE_COL = FILE_NAME
FILE_COL(IDOT:IDOT) = '.
FILE_COL(IDOT+1:IDOT+3) = 'COL'
C
FILE_KIN = FILE_NAME
FILE_KIN(IDOT:IDOT) = '.
FILE_KIN(IDOT+1:IDOT+3) = 'KIN'
C
FILE_OUT = FILE_NAME
FILE_OUT(IDOT:IDOT) = '.
FILE_OUT(IDOT+1:IDOT+3) = 'OUT'
C
FILE_FIT = FILE_NAME
FILE_FIT(IDOT:IDOT) = '.
FILE_FIT(IDOT+1:IDOT+3) = 'FIT'
C
OPEN(LOG,FILE=FILE_LOG,STATUS='UNKNOWN',ERR=230)
CLOSE(LOG,STATUS='DELETE')
OPEN(LOG,FILE=FILE_LOG,STATUS='NEW')
GO TO 250
230 CONTINUE
WRITE(*,240)FILE_LOG
240 FORMAT( ' File I/O Error : ',A20)
ERROR = .TRUE.
GO TO 550
250 CONTINUE
C
OPEN (COL, FILE=FILE_COL, STATUS='UNKNOWN', ERR=330)
CLOSE (COL, STATUS='DELETE')
OPEN (COL, FILE=FILE_COL, STATUS='NEW')
GO TO 350
330 CONTINUE
WRITE(*,340)FILE_COL
340 FORMAT(' File I/O Error : ',A20)
ERROR = .TRUE.
GO TO 550
350 CONTINUE
C
OPEN (KIN, FILE=FILE_KIN, STATUS='UNKNOWN', ERR=430)
CLOSE (KIN, STATUS='DELETE')
OPEN (KIN, FILE=FILE_KIN, STATUS='NEW')
GO TO 450
430 CONTINUE
WRITE(*,440)FILE_KIN
440 FORMAT(' File I/O Error : ',A20)
ERROR = .TRUE.
GO TO 550
450 CONTINUE
C
OPEN (FIT, FILE=FILE_FIT, STATUS='UNKNOWN', ERR=530)
CLOSE (FIT, STATUS='DELETE')
OPEN (FIT, FILE=FILE_FIT, STATUS='NEW')
C
OPEN (OUT, FILE=FILE_OUT, STATUS='UNKNOWN', ERR=530)
CLOSE (OUT, STATUS='DELETE')
OPEN (OUT, FILE=FILE_OUT, STATUS='NEW')
GO TO 550
530 CONTINUE
WRITE(*,540)FILE_OUT
540 FORMAT(' File I/O Error : ',A20)
ERROR = .TRUE.
GO TO 550
550 CONTINUE
C
RETURN
END
C
FUNCTION IFIRCH(STRING)
C BEGIN PROLOGUE IFIRCH
C DATE WRITTEN 850626
C REVISION DATE 850626
C CATEGORY NO. M4.
C KEYWORDS CHARACTER STRINGS, SIGNIFICANT CHARACTERS
C AUTHOR CLARK, G.L., GROUP C-3 LOS ALAMOS NAT'L LAB
C PURPOSE Determines first significant (non-blank) character
C in character variable
C DESCRIPTION
C------------------------------------------------------------------------------------------------------------
C IFIRCH locates the first non-blank character in a string of
C arbitrary length. If no characters are found, IFIRCH is set = 0.
C When used with the companion routine ILASCH, the length of a string
C can be determined, and/or a concatenated substring containing the
C significant characters produced.
C------------------------------------------------------------------------------------------------------------
C REFERENCES (NONE)
C ROUTINES CALLED (NONE)
C** PROLOGUE IFIRCH
C**** precision > double
IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER (I-N)
C**** END precision > double
C
C**** precision > single
IMPLICIT REAL (A-H,O-Z), INTEGER (I-N)
C**** END precision > single
C
CHARACTER* (*)STRING
C
FIRST EXECUTABLE STATEMENT IFIRCH
NLOOP = LEN(STRING)
C
IF (NLOOP .EQ. 0) THEN
  IFIRCH = 0
  RETURN
ENDIF
C
DO 100 I = 1, NLOOP
  IF (STRING(I:I) .NE. '  ') GO TO 120
100 CONTINUE
C
IFIRCH = 0
RETURN
120 CONTINUE
IFIRCH = I
END

C*****FITF**************************************
subroutine FitF(lecho)
implicit none
include 'cdparams.fh'
include 'cdrange.fh'
include 'cdffunc.fh'
include 'cdffit.fh'
include 'cdtemp.fh'
c local variables
integer iter,1WK,INFO,iWK(mxFitP),M,J,it,ip,I
integer Iparam(7),lecho,mxIter,mxFEval,mxGEval
c real*8 Rparam(7),Xguess(mxFitP),Xscale(mxFitP),Fvalue,
 2 X(mxFitP),Fscale,TOL,FNORM,ENORM,
4 WK(mxFitP*mxTpts * mxPPts + 5*mxFitP + mxTpts * mxPPts + 20)
real*8 T_cheb,P_cheb,RK_tempLog,PhiCh
external FCN
c end declarations
C get initial guess for solution; scaling info
C
call initX(Nfit,Xguess,Xscale,Fscale,funstr,Fopt)
c
CHANGED 02/95
iTempWell = iWell
iTempProd = iProd
if (Fopt.eq.'CHEB') Nfit = N_cheb*M_cheb
C write(*,*)Nfit,N_cheb,M_cheb
C call initX(Nfit,X,Xscale,Fscale,funstr,Fopt)
c
c initialize minimizer
  c flag to reset parameters
  Iparam(1) = 1
  c call DU4INF(Iparam,Rparam)
  c change some parameters for minimizer
  call resetP(Iparam,Rparam,mxIter,mxFEval,mxGEval)
  c U2INF is the IMSL minimization fitting routine
  c NOTE U2INF is really for single precision but we
  c always compile with opt CRAY real*8 --> single precision
  c (changed to du2inf for SGI <ayc 3/94>)
  c
  c This block has been changed by W. Ing
  c sub hybridl from MINPACK was incorporated to replace
  c IMSL minimization routine U2INF.
  c call DU2INF(FCN,Nfit,Xguess,Xscale,Fscale,Iparam,Rparam,
  c 2 X, Fvalue, WK)
  c
  c 1WK = mxFitP*(3*mxFitP+13)
  TOL = 1.0D-10
  M = ntemps*npres
  1wk = mxFitP*M + 5*mxFitP + M + 20
  c Fvalue(1) = 1.d80
  c call HYBRD1(FCN,Nfit,X,Fvalue,TOL,INFO,WK,1WK)
  c call LMDIF1(FCN,M,Nfit,X,Fvalue,TOL,INFO,WK,1WK)
  c
  c test for convergence
  c if (Iparam(3).ge.mxIter) write(lecho,100) mxIter
  C100 format(/,'ERROR: No convergence in ',i5, ' iterations.')
  c if (Iparam(4).ge.mxFEval) write(lecho,110) mxFEval
  C110 format(/,'ERROR: No convergence in ',i5,
  c 2 ' function evaluations.')
  c if (Iparam(5).ge.mxGEval) write(lecho,120) mxGEval
  C120 format(/,'ERROR: No convergence in ',i5,
  c 2 ' gradient evaluations.')
  c if (INFO.eq.0) write(lecho,100)
  100 format(/,'ERROR: Improper input parameters ')
  if (INFO.eq.4) write(lecho,101)
  101 format(/,'WARNING: Fvalue is orthogonal to the columns of the Jac
  c 2obian to machine precision.')
  if (INFO.eq.5) write(lecho,102)
  102 format(/,'WARNING: Number of calls to FCN has reached or exceeded
  c 2 200*Fit.')
  if (INFO.eq.6) write(lecho,103)
  103 format(/,'WARNING: TOL is too small. No further reduction in t
  c he sum of squares in possible')
  if (INFO.eq.7) write(lecho,104)
  104 format(/,'WARNING: TOL is too small. No further improvment in
  c 2he approximate solution X is possible')
  if (INFO.eq.1) write(lecho,105)
  105 format(/,'F Fitting : ALGORITHM ESTIMATES THAT THE RELATIVE ERROR
  c 2 IN THE SUM OF SQUARES IS AT MOST TOL.')
  if (INFO.eq.2) write(lecho,106)
  106 format(/,'F Fitting : ALGORITHM ESTIMATES THAT THE RELATIVE ERROR
2 BETWEEN X AND THE SOLUTION IS AT MOST TOL.
) if (INFO.eq.3) write(lecho,107)

107

format(/,' Fitting : ALGORITHM ESTIMATES THAT THE RELATIVE ERROR
2 both IN THE SUM OF SQUARES IS AT MOST TOL AND BETWEEN X AND THE
3 SOLUTION IS AT MOST TOL.
) C

C write(*,*)'...before out...
C write(*,*)X(J),J=1,Nfit
C fill out fit parameters into common variables
C write(*,*)'...before FitVec filled out...
C write(*,*)'...Nfit =
C write(*,*)INFO
C do 150 iter = 1,Nfit
FitVec(iter) = X(iter)
150 continue
C write(*,*)'...after FitVec filled out...
C write(*,*)FitVec(1),FitVec(2),FitVec(3)
C write(*,*)'...lecho, iWell, iProd = 1
C write(*,*)lecho,iWell,iProd
return
C
C**************write k & fit cheb K compare output
C continue
C endif
C write(*,*)'....after FitVec filled out...
C write(*,*)'....Xl-3 =
C write(*,*)FitVec(1),FitVec(2),FitVec(3)
C write(*,*)'....lecho, iWell, iProd =
C write(*,*)lecho,iWell,iProd
return
c
C********RESET*****************************************************************************
C subroutine resetP(iparam,Rparam,mxI,mxF,mxG)
C sets up parameters for minimizer - see IMSL doc.
parameter (mxIter = 1000, mxFEval = 4000, mxGEval = 4000)
integer Iparam(7),mxI,mxF,mxG
real*8 Rparam(7)
C all these things have defaults, but some
C we reset
C iparam(1) was control flag which we already invoked
C iparam(2) is # of machine-dependent good digits
C #3 is max # of iterations (def: 100)
Iparam(3) = mxIter
mxI = mxIter
C #4 is max # of function evaluations (def: 400)
Iparam(4) = mxFEval
mxF = mxFEval
C #5 is max # of gradient evaluations (def: 400)
Iparam(5) = mxGEval
mxG = mxGEval
C iparam(6) def = 0 - init Hessian to Identity
C iparam(7) is not used
C Rparam defaults look ok - based on machine
C dependent properties - i don't think we can
C do better - so we leave alone this time
return
c subroutine to define objective function

c see IMSL documentation

02/95 changed to use HYBRD1

c subroutine FCN(N, X, F)

subroutine FCN(M, N, X, F, IFLAG)

implicit none
include 'cdparams.fh'
include 'cdrange.fh'
include 'cdffunc.fh'
include 'cdffit.fh'
include 'cdtemp.fh'
include 'cdrates.fh'

c local variables
integer N,IFLAG,M
integer it, ip, ii, ntt
real*8 X(N), error,cumerr,Ffit,F(N)
real*8 getFfit

c end declarations

c cumerr = 0.d0
write(*,*)1  ....xl..x2..x3..'
write(*,*)x(l),x(2),x(3)
do 200 it = l,ntemps
cumerr = 0.0d0
do 100 ip = 1,npres
Ffit = getFfit(N,X,temp(it),Pr(it, ip),Fopt,iTempWell,
iTempProd,it,ip)
if (Fopt.eq."CHEB") then
error = dabs((Ffit-dloglO(RK(iTempwell,iTempprod,it,ip))))
endif
ntt = (it-l)*npres + ip
Ffit = getFfit(N,X,temp(it),Pr(it,ip),Fopt,iTempWell,
iTempProd,it,ip)
if (Fopt.eq."CHEB") then
error = dabs((Ffit-dloglO(RK(iTempwell,iTempprod,it,ip))))
endif

c92 format(1,'I2', 'I2', 'f8.4', 'f8.4', 'f8.4')
F(ntt) = error
WRITE(*,*)temp(it),pr(it,ip),error,cumerr
WRITE(*,*)F(it)
200 continue

c test

do 200 ii = 1, N
F(ii) = 1.0d-6
200 continue
422

end

**FITOUT**

subroutine FitOut(option, lfout1, lfout2, iwell, iprod)

fill out output files with Fcalc vs T, P

implicit none

include 'cdparams.fh'
include 'cdwell0.fh'
include 'cdlabels.fh'
include 'cdrange.fh'
include 'cdfunc.fh'
include 'cdfit.fh'

character option*8,RCname*20

integer it, ip,iwell,iprod,iter

integer lfout1, lfout2, itermax, ipEmax, nFlt90, nEgt10

real*8 Ffit, error, errormx, totterr

real*8 getFfit

character tab

end declarations

tab = char(9)

RCname = PDname(inpwell, inpchan)

write(lfout1,50) option, iwell, iprod, iter

RCname, PDname(iwell, iprod)

50 format(/,1x,a,'(',i2,',') -> ',i2,':',i2,' ' ,a, ' = ',a)

write(lfout1,150) tab, tab, tab, tab, tab, tab, tab, tab, tab, tab

150 format(/ ,l x, ' T (K)' ,a, ' 1000/T' ,a, ' log P' ,a, ' log Pr' ,a, 2
2 ' Fcalc' ,a,'-lnFcalc' ,a,' F fit' ,a,'-lnFfit' ,a,' %error')

nFlt90 = 0

nEgt10 = 0

errormx = 0.d0

totterr = 0.d0

do 350 it = 1,ntemps

do 300 ip = 1,npres

Ffit = getFfit(Nfit,FitVec,temp(it),Pr(it,ip),Popt,

iWell,iprod,it,ip)

if (Ffit.lt.9.d-1) nFlt90 = nFlt90 + 1

error = (Fcalc(it,ip)-Ffit) / dmin1(Fcalc(it,ip),Ffit)

if (dabs(error).gt.1.d-1) nEgt10 = nEgt10 + 1

totterr = totterr + dabs(error)

if (dabs(error).gt.errormx) then

errormx = dabs(error)

itermax = it

ipEmax = ip
endif

c

if (Ffit.eq.0) Ffit=1.0d-1

if (Pr(it,ip).eq.0) Pr(it,ip)=1.0d-1

if (pres(it,ip).eq.0) pres(it,ip)=1.0d-1

if (Fcalc(it,ip).eq.0) Fcalc(it,ip)=1.0d-1

c

c

c

write(lfout1,280) temp(it), tab,1000.d0/temp(it),

tab, dlog10(pres(it,ip)), tab, dlog10(Pr(it,ip)),

tab, Fcalc(it,ip), tab, -dlog(Fcalc(it,ip)),

tab, Ffit, tab, -dlog(Ffit), tab, error*100.d0

c280 format( ' ',f7.2,a,f7.4,a,f7.3,a,f7.3,a,f7.3,a,f7.3,a,f7.4,a,f7.3,

a,f7.4,a,f7.3,a,f7.2)

c

write(lfout1,280) temp(it), tab,1000.d0/temp(it),

tab, dlog10(pres(it,ip)), tab, dlog10(Pr(it,ip)),

tab, Fcalc(it,ip), tab, (Fcalc(it,ip)),

tab, Ffit, tab, (Ffit), tab, error*100.d0

c280 format( ' ',f7.2,a,f7.4,a,f7.3,a,f7.3,a,f7.4,a,f7.3,

a,f7.4,a,f7.3,a,f10.2)
300    continue
350 continue
    now write summary output to lfout2
    write(lfout2,50) option,inpwell,iwell,iprod,
      2 RCname,PDname(iwell,iprod)
    write(lfout2,400) funstr
400    format(//,'Fit: ',a,/) do 430 iter = 1,NFit
          write(lfout2,410) iter,FitVec(iter)
430    continue
    write(lfout2,450) npres*ntemps,nFlt90,nEgtl0,
      2 100.0*toterr/(npres*ntemps)
450      format(//,'Total points: ',i3,' # F'\'s < 0.9: ',i3,
          2 ' # err > 10%: ',i3,' avg err%: ',f5.2)
    Ffit = getFfit(Nfit,FitVec,temp(itEmax)),
      2 Pr(itEmax,ipEmax),Popt,iwell,iprod,itEmax,ipEmax)
    C
    C
    if (pres(itEmax,ipEmax) .eq. 0) pres(itEmax,ipEmax)=1.0d-1
    C
    write(lfout2,460) errormx*100.0,temp(itEmax),
      2 dlog10(pres(itEmax,ipEmax)),Fcalt(itEmax,ipEmax),Ffit
460       format(//,'Worst error%: ',f5.1,' T: ',f6.1,' logP: ',f5.2,
           2 ' Fcalc: ',f7.4,' Ffit: ',f7.4)
    write(lfout2,470) errormx*100.0,temp(itEmax),
      2 (pres(itEmax,ipEmax)),Fcalt(itEmax,ipEmax),Ffit
700       format(//,'Worst error%: ',f8.1,' T: ',f6.1,' logP: ',f5.2,
           2 ' Fcalc: ',f10.4,' Ffit: ',f10.4)
    return
    end
C* **************************************************
C** FCHEBOUT*****************************************
C********************************************************************
  subroutine FChebOut(option,lfout1,lfout2,lfout3,iwell,iprod)
    fill out output files with Fcalc vs T, P
    implicit none
    include 'cdparams.fh'
    include 'cdwell10.fh'
    include 'cdlabels.fh'
    include 'cdrange.fh'
    include 'cdfunc.fh'
    include 'cdffit.fh'
    include 'cdrates.fh'
    include 'cdkfit.fh'
    include 'cdcoils.fh'
    local variables
    character option*8,RCname*20,Pname*20
    integer it,ip,iwell,iprod,iter,I,J,irest
    integer lfout1,lfout2,itEmax,ipEmax,nFlt90,nEgtl0,lfout3
    real*8 Ffit,error,errormx,toterr
    real*8 getFfit,RK_tempLog,RK_temp,PhiCh,a,rn,ea
    character tab
    integer*2 year,month,day
    CHARACTER FILE_INP*20, FILE_LOG*20, FILE_COL*20,
      1 FILE_KIN*20, FILE_OUT*20, FILE_NAME*20
    COMMON/FILE/FILE_INP,FILE_LOG,FILE_COL,FILE_KIN,FILE_OUT,FILE_NAME
    end declarations
    tab = char(9)
    RCname = PDname(inpwell,ipchann)
    write(lfout1,50) option,inpwell,iwell,iprod,
      2 RCname,PDname(iwell,iprod)
50      format(//,1x,a,'(',i2,') -> ',i2,'.',i2,' ',a,' = ',a)
    write(lfout1,150) tab,tab,tab,tab,tab,tab,tab,tab,tab,tab
form(T,K)',a,'1000/T',a,'log P',a,'log Pr',a,
2 'Kcal',a,'log Kcal',a,'K fit',a,'log K fit',a,'error')

nfIt90 = 0
nEgt10 = 0
errormx = 0.d0
toterr = 0.d0
do 350 it = 1,ntemps
do 300 ip = 1,npres
   RK_tempLog = getFit(Nfit,FitVec,temp(it),Pr(it,ip),Popt,iWell,iprod,it,ip)
   RK_temp = 10**(RK_tempLog)
   if (Ffit(it,9.d-1) nFlt90 = nFlt90 + 1
   error = (RK_temp-RK(iWell,iprod,it,ip)) / 2
   if (dabs(error).gt.1.d-1) nEgt10 = nEgt10 + 1
   totterr = totterr + dabs(error)
   if (dabs(error).gt.errormx) then
      errormx = dabs(error)
      itEmax = it
      ipEmax = ip
   endif

   if (Ffit.eq.0) Ffit=1.0d-1
   if (Pr(it,ip).eq.0) Pr(it,ip)=1.0d-1
   if (pres(it,ip).eq.0) pres(it,ip)=1.0d-1
   if (Fcalc(it,ip).eq.0) Fcalc(it,ip)=1.0d-1
280 format('1,f7.2,a,f7.4,a,f7.3,a,f7.3,a,f7.4,a,f7.3,a,f7.2)
   write(lfout1,280) temp(it),tab,1000.d0/temp(it),
   tab,dlog10(pres(it,ip)),tab,dlog10(Pr(it,ip)),
   tab,Fcalc(it,ip),tab,-dlog(Fcalc(it,ip)),
   tab,Ffit,tab,-dlog(Ffit),tab,errormx*100.d0
280 format('1,f7.2,a,f7.4,a,f7.3,a,f7.3,a,f7.4,a,f7.3,a,E12.4,a,E12.4,a,E12.4,a,E12.4,a)
300 continue
350 continue

now write summary output to lfout2
write(lfout2,50) option,inpwell,iWell,iprod,PDname(iWell,iprod)
write(lfout2,400) funstr
400 format('/','Fit:','a,/
do 430 iter = 1,Nfit
   write(lfout2,410) iter,FitVec(iter)
410 format('Param(',i2,'):',i1pe,m4.7)
430 continue
write(lfout2,450) npres*ntemps,nFlt90,nEgt10,
2 100.d0*toterr/(npres*ntemps)
450 format('/','Total points: ','i3, '# F"s < 0.9: ','i3,
2 '# err > 10%: ',i3, 'avg err%: ','f5.2)
   Ffit = getFit(Nfit,FitVec,temp(itEmax),
2  Pr(itEmax,ipEmax),Popt,iWell,iprod,itEmax,ipEmax)
C C
   if (pres(itEmax,ipEmax).eq.0) pres(itEmax,ipEmax)=1.0d-1
C C
   write(lfout2,460) errormx*100.d0,temp(itEmax),
2  dlog10(pres(itEmax,ipEmax)),Fcalc(itEmax,ipEmax),Ffit
c460 format('(/,'Worst error%: ',f5.1,'T: ',f6.1,',logP: ',f5.2,10p,'Fcalc: ',f7.4,'Pfit: ',f7.4)
write(lfout2,460) errormx*100.d0,temp(itEmax),
2 (pres(itEmax,ipEmax)),RK(iwell,iprod,itEmax,ipEmax),10**fFIT
460 format('(/,'Worst error%: ',f8.1,'T: ',f6.1,'P: ',f5.2,10p,'Kcalc: ',ell.4,'Kfit: ',ell.4)
c
now write summary output to lfout3

call concat(PDname(inpwell,inpchan),'+(M)',RCname)
call concat(PDname(iwell,iprod),'+(M)',Pname)
c
RCname = PDname(inpwell,inpchan)
a = 1.0
rn = 0.0
ea = 0.0
call getdat(year,month,day)
year = year - 1900
irest = MOD(Nfit-4,5)
date(lfout3,500) RCname,Pname,a,rn,ea,FILE_NAME,month,year
500 format(a,'<=>','a,lp9.2,','0p7.3,','7.0,1x,
2 '!1,A8,1x,12,'/',12)
date(lfout3,510)
510 format('LOW/1.0 0.0 0.0/')
date(lfout3,520)n_cheb,m_cheb,(FitVec(iter),iter=1,4)
520 format('CHEB/','11,3x,11,5x,lp4(1x,ell.4,').'/
write(lfout3,530) (FitVec(iter),iter=5,Nfit-irest)
530 format('CHEB/','lp5(1x,ell.4,').'/
if (irest.eq.1) then
write(lfout3,540) (FitVec(iter),iter=Nfit-irest+1,Nfit)
540 format('CHEB/','lp1x,ell.4,').'/
else if (irest.eq.2) then
write(lfout3,550) (FitVec(iter),iter=Nfit-irest+1,Nfit)
550 format('CHEB/','lp2(1x,ell.4,').'/
else if (irest.eq.3) then
write(lfout3,560) (FitVec(iter),iter=Nfit-irest+1,Nfit)
560 format('CHEB/','lp3(1x,ell.4,').'/
else if (irest.eq.4) then
write(lfout3,570) (FitVec(iter),iter=Nfit-irest+1,Nfit)
570 format('CHEB/','lp4(1x,ell.4,').'/
endif
do 430 iter = 1,Nfit
write(lfout3,600) iter,FitVec(iter)
600 format('Param(',i2,':',lp14.7)
c430 continue

CH3 + OH (+M) = CH2O + H2 (+M) 1.00E+00 0.00 0. 14.04E+12
-51 2839.1CHEM241
C LOW/1.0 0.0 0.0 /
C CHEB/7.3 1.0044196E+01 -7.3333131E-01 -2.9292355E-01/
C CHEB/5.48661E-01 7.0805E-01 2.3241E-01 5.8694E-02 9.35E-02/
C CHEB/8.3981941E-02 -2.2423983E-02 -3.0251131E-02 4.4687742E-04 -
1.7928843E-02/
C CHEB/-2.8764778E-02 -1.5347567E-02 -7.4310585E-03 -1.3008562E-02 -
9.7005805E-03/
C CHEB/-1.8067830E-03 -3.2013939E-03 -3.1585273E-03/
c
return
e

C***INITX*****************************************************************************
c
sub to return initial-guess solution and scaling parameters
subroutine initX(Npar,X0,Xscale,Fscale,IDstr,option)
implicit none
c
local variables
integer Npar
real*8 X0(*),Xscale(*),Fscale
character IDstr*, option*8

end declarations

IMPORTANT: these options are set in getparams

if (option.eq.'Tree') then
  call initTree(Npar,X0,Xscale,Fscale,IDstr)
else if (option.eq.'CHEB') then
  call initCheb(Npar,X0,Xscale,Fscale,IDstr)
else if (option.eq.'SRI') then
  call initSRI(Npar,X0,Xscale,Fscale,IDstr)
else if (option.eq.'FRENK') then
  call initFreK(Npar,X0,Xscale,Fscale,IDstr)
else
  call initDEF(Npar,X0,Xscale,Fscale,IDstr)
endif
return
end

***GETFFIT*******************************************************************

routine to return fitted F

function getFFit(Npar,X,T,P,option,iWell,iProd,it,ip)
  implicit none

  local variables
  integer Npar
  real*8 X(*),T,P,it,ip,iWell,iProd
  real*8 getFFit,getTree,getSRI,getDef,getFreK,getCheb
  character option*8

end declarations

if (option.eq.'Tree') then
  getFFit = getTree(Npar,X,T,P)
else if (option.eq.'CHEB') then
  getFFit = getCheb(Npar,X,T,P,iWell,iProd,it,ip)
else if (option.eq.'SRI') then
  getFFit = getSRI(Npar,X,T,P)
else if (option.eq.'FRENK') then
  getFFit = getFreK(Npar,X,T,P,iWell,iProd,it,ip)
else
  getFFit = getDef(Npar,X,T,P)
endif
return
end

***INITCHEB************************************************************************

c companion routine to getFreK for initializing variables

subroutine initCheb(Npar,X0,Xscale,Fscale,IDstr)
  implicit none

  local variables
  integer iter
  integer Npar,Mpar
  real*8 X0(*),Xscale(*),Fscale
  character IDstr*(*)

end declarations

Npar = 3
no better idea so set Xscale to 1 - see IML description;
set all x's to order 1 (we define T's as x's * 1000)
center FIT function ID string here
IDstr = 'M,N-Parameter Chebyshev Fit in T/1000 (w/atan)'
do 10 iter = 1,Npar
  Xscale(iter) = 1.d0
  X0(iter) = 5.d0
10 continue
Fscale = 1.d0
now we reset the a factor
X0(3) = 2.0
return
end

***GETCHEB************************************************************************

c function to return default "a" fitting function

function getCheb(Npar,X,T,P,iWell,iProd,it,ip)
implicit none
include 'cdparams.fh'
include 'cdrange.fh'
include 'cdrates.fh'
include 'cdlimfit.fh'
include 'cdffunc.fh'
c local variables
integer Npar, it, ip, iWell, iProd, i, J
real*8 X(*), T, P, a
real*8 getCheb, kFit, PhiCh, T_cheb, P_cheb, RK_tempLog
c end declarations
c a = X(1)*exp(-((dlog10(P)-X(2))/X(3))**2)-(1/nPr(iwell,iprod))
C kFit = (RK(iwell,iprod,it,0)**a + RK(iwell,iprod,it,nPres+1)**a)**2
C 2 = (1/a)
C T_cheb = (2.*((1./temp(it))-(1/Tmin)-(1/Tmax)) / 2
C P_cheb = (2.*dlog10(pres(1,ip))-dlog10(Pmin)-dlog10(Pmax))/ 2
C RK_tempLog = 0.0
DO 20 I = 1, N_Cheb
DO 10 J = 1, M_Cheb
C write(*,*)RK_tempLog,((I-1)*M_cheb+J),X((I-1)*M_cheb+J),
C PhiCh(I,T_cheb),PhiCh(J,P_cheb)
C RK_tempLog = RK_tempLog
2 + X((I-1)*M_cheb+J)*PhiCh(I,T_cheb)*PhiCh(J,P_cheb)
10 CONTINUE
20 CONTINUE
C RK_Cheb(iwell,iprod,it,ip)=10**RK_tempLog
C write(*,*)'it...ip....a...k...kfit...' write(*,*)it,ip,a,RK(iwell,iprod,it,ip),kFit
C write(*,*)'it...ip...' write(*,*)it,ip
C getCheb = RK_Cheb(iwell,iprod,it,ip)
getCheb = RK_tempLog
C write(*,*)'logk**a+kfit**a-logk**a........getCheb = ...' write(*,*)getCheb
C return
end
C***PHICH*******************************************************************************
C FUNCTION PhiCh(i,x)
C implicit none
c local variables
integer i
real*8 x,PhiCh
C PhiCh = DCOS((i-1)*dacos(x))
return
end
C***INITFREK*******************************************************************************
c companion routine to getFreK for initializing variables
 subroutine initFreK(Npar,X0,Xscale,Fscale,IDstr)
 implicit none
c local variables
integer itera
integer Npar
real*8 X0(*), Xscale(*), Fscale
character IDstr(*)
c end declarations
Npar = 3
c no better idea so set Xscale to 1 - see IMSL description;
c set all x's to order 1 (we define T's as x's * 1000)
c enter FIT function ID string here
IDstr = '3-Parameter Frenklach Fit in T/1000 (w/atan)' do 10 itera = 1,Npar
Xscale(iter) = 1.0d1
X0(iter) = 5.0d1

10 continue
Fscale = 1.0d0
c Now we reset the a factor
X0(3) = 2.0
return
end

C***GETFREK******************************
C function to return default "a" fitting function
function getFreK(Npar, X, T, P, iWell, iProd, it, ip)
implicit none
include 'cdparams.fh'
include 'cdrange.fh'
include 'cdrates.fh'
include 'cdlimfit.fh'
include 'cddffunc.fh'
c local variables
integer Npar, it, ip, iWell, iProd
real*8 X(*), T, P, a
real*8 getFreK, kFit
c end declarations

a = X(1)*dexp(-|dlog10(P)-X(2))/X(3)|**2)-(1/nPr(iwell,iprod))
kFit = (RK(iwell,iprod,it,0)**a + RK(iwell,iprod,it,npres+1)**a)**2
write(*,*)'...it...ip...a...k...kfit...
write(*,*)it,ip,a,RK(iwell,iprod,it,ip),kFit
write(*,*)...it...ip...
write(*,*)it,ip
getFreK = dlog10(RK(iwell,iprod,it,0)**a
2 + RK(iwell,iprod,it,npres+1)**a)
3 - dlog10(RK(iwell,iprod,it,ip)**a)
write(*,*)'...logk0**a+kinf**a-logk**a.......getFreK = ...
write(*,*)getFreK
return
end

C***INITDEF******************************
C companion routine to getDef for initializing variables
subroutine initDef(Npar, X0, Xscale, Fscale, IDstr)
implicit none
c local variables
integer iter
integer Npar, iamp, ioff, iwidth, iasym
real*8 X0(*), Xscale(*), Fscale
c character IDstr(*)
c end declarations
data iamp, iwidth, iasym, ioff/1,6,9,12/
Npar = 12
c no better idea so set Xscale to 1 - see IMSL description;
c also zero out all X0's before setting non-zero values
c ok - let's try not quite zeroing out (we want init vector
c all the same order of magnitude)
c enter FIT function ID string here
IDstr = 'G (12): 5 amp, 3 width, 3 asym, 1 shift, exp = 1/2'
do 10 iter = 1,Npar
Xscale(iter) = 1.0d0
X0(iter) = 1.0d1
10 continue
Fscale = 1.0d0
c Now we reset the amplitude and width
X0(iamp) = 1.0d0
X0(iwidth) = 2.0d0
return
end

C***GETDEF******************************
C function to return default F fitting function
function getDef(Npar,X,T,P)
  implicit none
  local variables
  integer Npar, iamp, ioff, iwidth, iasym
  real*8 X(*), T, P, FcentLn, FLShape, Plgoff, Width, asym, Plg,
  2      arg, Tnm
  real*8 getDef
  end declarations
  IMPORTANT: data statement must be consistent with initDef
  IDstr set in initDef
  we invent new scaled T
  Tnm = (1.d-3*T)**0.5
  use base 10 log for P
  Plg = dlog10(P)
  a: we use first 5 parameters for amplitude function
  and we try fitting to cubic in 1000./T
  FcentLn = X(iamp+ )X(iamp+1)/(Tnm) + X(iamp+2)/(Tnm*Tnm)
  2   + X(iamp+3)/(Tnm*Tnm*Tnm) + X(iamp+4)/(Tnm*Tnm*Tnm*Tnm)
  the rest are lineshape parameters:
  b: we use 3 parameters for Half-width
  Width = X(iwidth) + X(iwidth+1)/(Tnm) + X(iwidth+2)/(Tnm*Tnm)
  c: we use 3 parameters for asymmetry parameter
  asym = X(iasym) + X(iasym+1)/(Tnm) + X(iasym+2)/(Tnm*Tnm)
  d: we use 1 parameter for offset from 0
  Plgoff = X(ioff)
  now the lineshape function
  Lorentzian: we don't use it but we keep it just in case
  FLShape = 1.d0 / 
  2   ( 1.d0 + ((Plg + Plgoff)/(Width + asym*Plg))**2 )
  Gaussian:
  FLShape = dexp( -((Plg + Plgoff)/(Width + asym*Plg))**2 )
  finally return the solution; but constrain argument
  to exponential to avoid overflows
  arg = -FLShape*FcentLn
  if (arg.gt.100.d0) arg = 100.d0
  if (arg.lt.-100.d0) arg = -100.d0
  getDef = dexp(arg)
  return
end

***INITTROE****************************
companion routine to getTroe for initializing variables

subroutine initTroe(Npar,X0,Xscale,Fscale,IDstr)
  implicit none
  local variables
  integer iter
  integer Npar
  real*8 X0(*),Xscale(*),Fscale
  character IDstr(*)
  end declarations
  Npar = 4
  no better idea so set Xscale to 1 - see IMSL description;
  set all x's to order 1 (we define T's as x's * 1000)
  enter FIT function ID string here
  IDstr = '4-Parameter Troe Fit in T/1000 (w/atan)'
  do 10 iter = 1,Npar
     Xscale(iter) = 1.d0
     X0(iter) = 5.d-1
  10 continue
  Fscale = 1.d0
  Now we reset the a factor
  X0(1) = 4.d-1
  return
end

***GETTROE******************************
function to return default P fitting function
function getTroe(Npar,X,T,P)
implicit none
  c local variables
    integer Npar,icoef
    real*8 X(*),T,P,TStar(3),a,Fcent,
    2   FLSShape,Rn,Rc,Rd,Plg
    real*8 getTroe
  c end declarations
  data Rd /0.14/
  c we try atan function to constrain a between 0 and 1
  a = datan(X(1))/3.14159265359d0 + .5d0
  c we wish to constrain T's to be positive
  do 10 icoef = 1,3
    Tstar(icoef) = l.d3*dabs(X(icoef+1))
  10 continue
  c use base 10 log for P
  Plg = dlog10(P)
  c this is right out of the Chemkin II manual, pg 23
  Fcent = (1.d0 - a)*dexp(-T/TStar(3)) + a*dexp(-T/TStar(1))
  2 + dexp(-TStar(2)/T)
    Rc = -0.4d0 - 0.67d0*dlog10(Fcent)
    Rn = 0.75 - 1.27*dlog10(Fcent)
    FLShape = 1.d0 /
    2 ( 1.d0 + ( (Plg + Rc)/(Rn - Rd*(Plg + Rc)) )**2 )
  getTroe = Fcent**2(FLShape)
  return
end

C**INITSRI*******************************
C companion routine to getSRI for initializing variables
subroutine initSRI(Npar,X0,Xscale,Fscale,ID str)
implicit none
  c local variables
    integer iter
    integer Npar
    real*8 X0(*),Xscale(*),Fscale
    character IDstr(*)
  c end declarations
  Npar = 3
  c no better idea so set Xscale to 1 - see IMSL description;
  c enter FIT function ID string here
  IDstr = '3-Parameter SRI Fit in T/1000'
  do 10 iter = 1,Npar
    X0(iter) = 1.d0
    Xscale(iter) = 1.d0
  10 continue
  Fscale = 1.d0
  c Now we reset the a factor
  X0(1) = 4.d-1
  return
end

C**GETSRI*******************************
C function to return default F fitting function
function getSRI(Npar,X,T,P)
implicit none
  c local variables
    integer Npar
    real*8 X(Npar),T,P,a,b,c,Fcent,FLShape,Plg
    real*8 getSRI
  c end declarations
  c (tried an equiv statement here, but compiler didn't
  c like it - probably because of passed variable)
  a = dabs(X(1))
  b = 1.d3*dabs(X(2))
  c = 1.d3*dabs(X(3))
  c use base 10 log for P
  Plg = dlog10(P)
this is right out of the Chemkin II manual, pg 24 -
but we have not included the optional d and e factors,
because we don't see that kind of limiting behavior

\[
F_{\text{cent}} = a * \exp(-b/T) + \exp(-T/c)
\]
\[
F_{\text{LShape}} = \frac{l.dO}{l.dO + P_lg^2}
\]
\[
getSRI = F_{\text{cent}}^{l.dO}(F_{\text{LShape}})
\]
return
end

we tried these subs but we don't use them anymore

**BOUNCR***********************************
func to reflect fitting parameter to right of of boundary
function bouncR(xval,xlim,xzone)
implicit none
real*8 xval,xlim,xzone
real*8 bouncR
bouncR is the reflection of xval to the other side
of xlim but within the region defined by xzone
xzone should be small compared to expected magnitude of
xval (because legitimate xval's could be stuck in the
zone); still xzone should be large enough that things
put there don't simple vanish into the noise
the test condition (xval < xlim) is assumed
bouncR = xlim + xzone/(l.dO + (xlim - xval))
return
end

**BOUNCL***********************************
func to reflect fitting parameter to left of of boundary
function bouncL(xval,xlim,xzone)
implicit none
real*8 xval,xlim,xzone
real*8 bouncL
bouncL is the reflection of xval to the other side
of xlim but within the region defined by xzone
xzone should be small compared to expected magnitude of
xval (because legitimate xval's could be stuck in the
zone); still xzone should be large enough that things
put there don't simple vanish into the noise
the test condition (xval > xlim) is assumed
bouncL = xlim - xzone/(l.dO + (xval - xlim))
return
end

program chemAct
version 5.3 <ing 5/95>
<v5.2> changed file input style to prompting users keyin input file
name,
output to fort.20, fort.30.... no longer used, changed to
<input_put_file_name>.log,
<input_put_file_name>.kin,
<input_put_file_name>.col,
<input_put_file_name>.out <ing 10/94>
<v5.3> added CHEB keyboard for chebyshov polynomials,
implimented subroutines for SGI version only to pc, so now
fitglobal, tree...keyword can be used, ....hurray <ing 5/95>
*****start here
non-standard implicit, but almost universal;
if problems switch to character or logical
implicit none
grab parameter statments for commons
include 'cdparams.fh'
grab only those commons needed for main
include 'cdwells.fh'
include 'cdrange.fh'
include 'cdrates.fh'
include 'cdlimfit.fh'
include 'cdcontri.fh'
include 'cdTemp.fh'
include 'cdFFit.fh'

c delcare local variables
character option*8,fstamp*120
integer IWell,iProd,ip,n
integer*2 hr_i,min_i,sec_i,hun_i,hr,min,sec,hun
integer ip,n
integer Nluout
real*8 dE
logical proceed,error

c assign lu's; Nluout is in data statement
integer linput,lecho,lkout,ldiag,lfout,lcout,ltabou,ltplot
2 lffit,lfsum,lecon,luarr(11),lussave(11),lplot
equivalence (luarr(1),lecho),(luarr(2),lkout),(luarr(3),ldiag),
2 (luarr(4),lfout),(luarr(5),ltplot),(luarr(6),lkout),
3 (luarr(7),ltplot),(luarr(8),lffit),(luarr(9),lfsum),
4 (luarr(10),lecon),(luarr(11),lplot)
data linput,lecho,lkout,ldiag,lfout,lcout,lplot,ltplot,
2 lffit,lfsum,lecon,lplot,Nluout
3 /10,20,25,30,35,45,55,65,75,85,95,99,11/
data lussave/ 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1/
c common/io/linput,lecho,lkout,ldiag,lfout,lcout,lplot,ltplot,
2 lffit,lfsum,lecon, Nluout

******end declarations ***********
c default T and P range assigned through block data getrange
c these defaults can be overwritten if option specified
c open and read file header from input file
c open(luinput, file = 'fort.10', status = 'old')
call file_io(error)
call gettime(hr_i,min_i,sec_i,hun_i)
read(luinput,'(a)') fstamp
c open all output files and stamp header
call initfs(Nluout,luarr,lussave,fstamp)
c read all other input parameters from input file and close
call getparams(lininput,lecho,dE)
close(lininput, status = 'keep')
c echo generic parameters to file
call echoparams(lecho)
c loop through remainder of program doing chemact calculation first
c then do dissociation calculations if desired
write('*','')
write('*',*) 'program running...'
do 200 n = 0,nWells
c set switches
if ((n.eq.0).and.(chemact)) then
  option = 'Chemact'
  proceed = .true.
else if ((n.ge.1).and.(dissoc)) then
  option = 'Dissoc'
inpWell = n
inpchan = 0
  proceed = .true.
else
  proceed = .false.
endif
c if (proceed) then
c get array idpWell giving depth of complex on isomer chart;
c also compute Well depths in kcal; and Emax
C write(*,'')
C write(*,'') 'program running...'
c write(*,'') 'setcalc'
call setCalc(lecho,iso,omers,option)
c write(*,'') 'echo2'
**C**

to

now do actual calculation

if (option.eq.'Chemact') then

  write(*,*)'chemical activation...'

call compChem(lecho,ldiag,rkold,dE)

write(*,*)' done'

else

  write(*,*)'

format('unimolec dissociation: well # ',i2)

call compDiss(lecho,ldiag,rkold,dE)

write(*,*)'...done'

endif

finish calculation

call cleanup(option)

write(*,*)'fitting and writting output...'

call cleanup(option)

write(*,*)'output'

now we output results; first we print table

if (table) call tabout(option,ltabout)

now we loop over all reactions and output and or fit

remember iProd = 0 is stabilization channel for chemact

note common blocks Kfit and Pfit only store one reaction's

worth of parameters at a time

do 100 iWell = 1,nWells

do 80 iP = 1,nPres

if (iProd = 0) then

  the boolean addTerm indicates whether we must include

  secondary low p limit term

  if (((option.eq.'Dissoc').and.(noExit(iWell))

     .and.(iProd.eq.0)) then

    addTerm = .true.

  else

    addTerm = .false.

  endif

  don't output dissociation channels where dissociating isomer

  is Product - this is a useless rate and we integrated for

  an E > 0 so we didn't calculate this correctly anyway

  if (.not.((option.eq.'Dissoc').and.(iWell.eq.inpWell)

     .and.(iProd.eq.0))) then

  endif

fit limiting arrhenius factors

if ((fitglobal.or.fitrange).and.(nTemps.gt.2)) then

  if (addTerm)

    call Kfitout(option,lecho,lkout,iWell,
     iP,-1)

    call Kfitout(option,lecho,lkout,iWell,iProd,0)

    call Kfitout(option,lecho,lkout,iWell,iProd,
     npres+1)

  endif

fit arrhenius factors over p range

if (fitrange.and.(nTemps.gt.3)) then

  do 70 ip = 1,nPres

  continue

  endif

fit lineshape factors; make sure enough parameters

if (((fitglobal).and.(nTemps.gt.3)).and.
(nPres.gt.5)) then
write(*,'(f fitting f functions..')
if (Fopt.eq.'CHEB') then
write(*,120)iwell,iprod
format(' fitting chebyshev polynomials...','
iwel #',i2,' channel # ',i2)
call kLimFit(option,lecho,ifsum,iWell,iProd)
call calcF(option,iWell,iProd)
call FitF(lecho)
call FChebOut(option,lffit,lfsum,lcout,
call FChabOut(option,lffit,lecho,lcout,
iWell,iProd)
call reConst(option,irecon,iWell,iProd)
else
write(*,130)iwell,iprod
format(' fitting f functions...well #',
iwel #',i2)
call kLimFit(option,lecho,ifsum,iWell,iProd)
call calcF(option,iWell,iProd)
if (showline) then
call Foutput(option,lfout,iWell,iProd)
call Toutput(option,ltout,iWell,iProd)
endif
endif
continue
100 continue
endif
continue
200 continue
c close all output files
999 continue
c call closfs(Nluout,luarr,lu/save)
c call gettim(hr,min,sec,hun)
if (sec.lt.sec_i) then
 sec = sec + 60 - sec_i
 min = min - 1
else
 sec = sec - sec_i
endif
if (min.lt.min_i) then
 min = min + 60 - min_i
 hr = hr - 1
else
 min = min - min_i
endif
hr = hr - hr_i
write(lecho,300)hr,min,sec
300 format(1x,'execution time : ','i2,' hours ','i2,' minutes ','i2,
 ' seconds')
write(*,310)hr,min,sec
310 format(1x,'execution time : ','i2,' hours ','i2,' minutes ','i2,
 ' seconds')
c call closfs(Nluout,luarr,lu/save)
c stop
end

***********************************
subroutine initfs(Nlu,lu,lu/save,string)
c opens and clears out all output files and stamps header
implicit none
c local variables
 integer Nlu,lu(Nlu),lu/save(Nlu)
integer i
character string*(*)",num*2,fname*8

c end declarations

do 20 i = 1,Nlu
if (luseave(i).eq.0) then
  write(num,'(i2)') lu(i)
call concat('fort.',num,fname)
open(lu(i), file = fname, status = 'unknown')
close(lu(i), status = 'open')
open(lu(i), file = fname, status = 'new')
write(lu(i),'(lx,a)') string
endif
20 continue
return
end

***CLOSFS***************************

subroutine closfs (Nlu,lu,luseave)
implicit none

local variables
integer Nlu,lu(Nlu),luseave(Nlu)
in integer i
c end declarations

do 20 i = 1,Nlu
if (luseave(i).eq.1) then
  close(lu(i), status = 'keep')
else
  close(lu(i), status = 'delete')
endif
20 continue
return
end

***GETRANGE***************

block data getrange
implicit none
include 'cdparams.fh'
include 'cdrange.fh'

c local variables
integer i,j
c end declarations

data ntemps,npres/13,61/
data (temp(i),i=1,13) /300.,400.,500.,600.,800.,1000.,1200.,1500.,
2 2000.,2500.,3200.,4000.,4800./
data ((pres(j),i=1,13),j=1,21) /13*1.d-10,13*2.d-10,13*5.d-10,
3 13*2.d-7,13*5.d-7,13*1.d-6,13*2.d-6,13*5.d-6,13*1.d-5,13*2.d-5,
data ((pres(i),j=1,21),i=1,13) /22*1.d-10,22*2.d-10,22*5.d-10,
2 22*1.d-9,22*2.d-9,22*5.d-9,22*1.d-8,22*2.d-8,22*5.d-8,22*1.d-7,
3 22*2.d-7,22*5.d-7,22*1.d-6,22*2.d-6,22*5.d-6,22*1.d-5,22*2.d-5,
data (temp(i),i=1,13) /300.,400.,500.,600.,800.,1000.,1200.,1500.,
2 2000.,2500.,3200.,4000.,4800./
data (pres(j),j=1,21) /13*1.d-10,13*2.d-10,13*5.d-10,
3 13*2.d-7,13*5.d-7,13*1.d-6,13*2.d-6,13*5.d-6,13*1.d-5,13*2.d-5,
data (temp(i),i=1,13) /300.,400.,500.,600.,800.,1000.,1200.,1500.,
2 2000.,2500.,3200.,4000.,4800./
data (pres(j),j=1,21) /13*1.d-10,13*2.d-10,13*5.d-10,
3 13*2.d-7,13*5.d-7,13*1.d-6,13*2.d-6,13*5.d-6,13*1.d-5,13*2.d-5,
end

***LOOKUP******************

subroutine lookup(sname,rmass,sig,ek,deltaE,noinfo)
implicit none
integer madata
parameter (madata = 10)

local variables
logical noinfo
character sname*10
integer n
integer ndata
real*8 rmass, sig, ek, deltaE
character species(mxdata)*10
real*8 smass(mxdata), ssig(mxdata), sek(mxdata), sdeltaE(mxdata)
c do assignments here
data ndata/6/
data species(1) /'N2'/
data smass(1), ssig(1), sek(1), sdeltaE(1)/28.0, 3.621, 97.5, 830.0/
data species(2) /'AR'/
data smass(2), ssig(2), sek(2), sdeltaE(2)/40.0, 3.33, 136.5, 630.0/
data species(3) /'HE'/
data smass(3), ssig(3), sek(3), sdeltaE(3)/4.0, 2.6, 10.2, 431.0/
data species(4) /'CH4'/
data smass(4), ssig(4), sek(4), sdeltaE(4)/16.0, 3.746, 141.4, 2100.0/
data species(5) /'C3H8'/
data smass(5), ssig(5), sek(5), sdeltaE(5)/44.0, 4.98, 266.8, 4200.0/
data species(6) /'SF6'/
data smass(6), ssig(6), sek(6), sdeltaE(6)/146.0, 5.13, 222.1, 3400.0/
noinfo = .true.
do 10 n = 1,ndata
   if (sname.eq.species(n)) then
      rmass = smass(n)
      sig = ssig(n)
      ek = sek(n)
      deltaE = sdeltaE(n)
      noinfo = .false.
   endif
10 continue
return
end
REFERENCES

21 Sorek, H. *Combustion Science and Technology* 1984, 41, 203.
30 Grotheer, H.; Riekert, G.; Walter, D; Just, T. *Twenty-second Symposium (International) on Combustion* 1988, 963.


46 Chang, A.Y., Bozzelli, J.W., and Dean, A.M., (To be Published). Personnel communication, Chang, A.Y., and Dean, A.M., Exxon Research and Engineering, Annendale, NJ.


53 Westmoreland, P. R. Combust. Sci. and Tech. 1992, 82, 151.


Venkatesh, P.; Chang, A. Y.; Dean, A. M.; Cohen, M. H.; Carr, R. W. (To be Published), Personnel communication.


